

# TMS320C6742 DSP

## Technical Reference Manual



Literature Number: SPRUH81C  
April 2013–Revised September 2016

<b>Preface</b> .....	<b>50</b>
<b>1 Overview</b> .....	<b>51</b>
1.1 Introduction.....	52
1.2 DSP Subsystem .....	52
<b>2 DSP Subsystem</b> .....	<b>53</b>
2.1 Introduction.....	54
2.2 TMS320C674x Megamodule .....	55
2.2.1 Internal Memory Controllers .....	55
2.2.2 Internal Peripherals .....	55
2.3 Memory Map.....	59
2.3.1 DSP Internal Memory.....	59
2.3.2 External Memory .....	59
2.4 Advanced Event Triggering (AET) .....	59
<b>3 System Interconnect</b> .....	<b>60</b>
3.1 Introduction.....	61
3.2 System Interconnect Block Diagram.....	62
<b>4 System Memory</b> .....	<b>63</b>
4.1 Introduction.....	64
4.2 DSP Memories.....	64
4.3 Peripherals .....	64
<b>5 Memory Protection Unit (MPU)</b> .....	<b>65</b>
5.1 Introduction.....	66
5.1.1 Purpose of the MPU .....	66
5.1.2 Features .....	66
5.1.3 Block Diagram .....	66
5.1.4 MPU Default Configuration.....	67
5.2 Architecture .....	67
5.2.1 Privilege Levels.....	67
5.2.2 Memory Protection Ranges .....	68
5.2.3 Permission Structures .....	68
5.2.4 Protection Check .....	69
5.2.5 DSP L1/L2 Cache Controller Accesses .....	70
5.2.6 MPU Register Protection .....	70
5.2.7 Invalid Accesses and Exceptions .....	70
5.2.8 Reset Considerations .....	70
5.2.9 Interrupt Support .....	70
5.2.10 Emulation Considerations .....	71
5.3 MPU Registers.....	71
5.3.1 Revision Identification Register (REVID).....	72
5.3.2 Configuration Register (CONFIG).....	73
5.3.3 Interrupt Raw Status/Set Register (IRAWSTAT).....	74
5.3.4 Interrupt Enable Status/Clear Register (IENSTAT) .....	75
5.3.5 Interrupt Enable Set Register (IENSET) .....	76
5.3.6 Interrupt Enable Clear Register (IENCLR).....	76

5.3.7	Fixed Range Start Address Register (FXD_MPSAR) .....	77
5.3.8	Fixed Range End Address Register (FXD_MPEAR) .....	77
5.3.9	Fixed Range Memory Protection Page Attributes Register (FXD_MPPA).....	78
5.3.10	Programmable Range <i>n</i> Start Address Registers (PROG <sub><i>n</i></sub> _MPSAR) .....	79
5.3.11	Programmable Range <i>n</i> End Address Registers (PROG <sub><i>n</i></sub> _MPEAR) .....	80
5.3.12	Programmable Range <i>n</i> Memory Protection Page Attributes Register (PROG <sub><i>n</i></sub> _MPPA) .....	81
5.3.13	Fault Address Register (FLTADDR) .....	82
5.3.14	Fault Status Register (FLTSTAT) .....	83
5.3.15	Fault Clear Register (FLTCLR).....	84
<b>6</b>	<b>Device Clocking .....</b>	<b>85</b>
6.1	Overview.....	86
6.2	Frequency Flexibility .....	88
6.3	Peripheral Clocking .....	89
6.3.1	DDR2/mDDR Memory Controller Clocking .....	89
6.3.2	EMIFA Clocking .....	91
6.3.3	McASP Clocking.....	92
6.3.4	I/O Domains.....	93
<b>7</b>	<b>Phase-Locked Loop Controller (PLL) .....</b>	<b>94</b>
7.1	Introduction.....	95
7.2	PLL Controllers .....	95
7.2.1	Device Clock Generation .....	97
7.2.2	Steps for Programming the PLLs .....	98
7.3	PLL Registers .....	100
7.3.1	PLL0 Revision Identification Register (REVID) .....	101
7.3.2	PLL1 Revision Identification Register (REVID) .....	102
7.3.3	Reset Type Status Register (RSTYPE).....	102
7.3.4	PLL0 Reset Control Register (RCTRL) .....	103
7.3.5	PLL0 Control Register (PLLCTL) .....	104
7.3.6	PLL1 Control Register (PLLCTL) .....	105
7.3.7	PLL0 OBSCLK Select Register (OCSEL) .....	106
7.3.8	PLL1 OBSCLK Select Register (OCSEL) .....	107
7.3.9	PLL Multiplier Control Register (PLLM).....	108
7.3.10	PLL0 Pre-Divider Control Register (PREDIV).....	108
7.3.11	PLL0 Divider 1 Register (PLLDIV1) .....	109
7.3.12	PLL1 Divider 1 Register (PLLDIV1) .....	109
7.3.13	PLL0 Divider 2 Register (PLLDIV2) .....	110
7.3.14	PLL1 Divider 2 Register (PLLDIV2) .....	110
7.3.15	PLL0 Divider 3 Register (PLLDIV3) .....	111
7.3.16	PLL1 Divider 3 Register (PLLDIV3) .....	111
7.3.17	PLL0 Divider 4 Register (PLLDIV4) .....	112
7.3.18	PLL0 Divider 5 Register (PLLDIV5) .....	112
7.3.19	PLL0 Divider 6 Register (PLLDIV6) .....	113
7.3.20	PLL0 Divider 7 Register (PLLDIV7) .....	113
7.3.21	PLL0 Oscillator Divider 1 Register (OSCDIV).....	114
7.3.22	PLL1 Oscillator Divider 1 Register (OSCDIV).....	114
7.3.23	PLL Post-Divider Control Register (POSTDIV) .....	115
7.3.24	PLL Controller Command Register (PLLCMD) .....	115
7.3.25	PLL Controller Status Register (PLLSTAT) .....	116
7.3.26	PLL0 Clock Align Control Register (ALNCTL) .....	117
7.3.27	PLL1 Clock Align Control Register (ALNCTL) .....	118
7.3.28	PLL0 PLLDIV Ratio Change Status Register (DCHANGE) .....	119
7.3.29	PLL1 PLLDIV Ratio Change Status Register (DCHANGE) .....	120
7.3.30	PLL0 Clock Enable Control Register (CKEN).....	121

7.3.31	PLL1 Clock Enable Control Register (CKEN)	121
7.3.32	PLL0 Clock Status Register (CKSTAT)	122
7.3.33	PLL1 Clock Status Register (CKSTAT)	123
7.3.34	PLL0 SYSCLK Status Register (SYSTAT)	124
7.3.35	PLL1 SYSCLK Status Register (SYSTAT)	125
7.3.36	Emulation Performance Counter 0 Register (EMUCNT0)	126
7.3.37	Emulation Performance Counter 1 Register (EMUCNT1)	126
<b>8</b>	<b>Power and Sleep Controller (PSC)</b>	<b>127</b>
8.1	Introduction	128
8.2	Power Domain and Module Topology	128
8.2.1	Power Domain States	130
8.2.2	Module States	130
8.3	Executing State Transitions	132
8.3.1	Power Domain State Transitions	132
8.3.2	Module State Transitions	132
8.4	IcePick Emulation Support in the PSC	133
8.5	PSC Interrupts	133
8.5.1	Interrupt Events	133
8.5.2	Interrupt Registers	134
8.5.3	Interrupt Handling	135
8.6	PSC Registers	136
8.6.1	Revision Identification Register (REVID)	137
8.6.2	Interrupt Evaluation Register (INTEVAL)	137
8.6.3	PSC0 Module Error Pending Register 0 (modules 0-15) (MERRPR0)	138
8.6.4	PSC1 Module Error Pending Register 0 (modules 0-31) (MERRPR0)	138
8.6.5	PSC0 Module Error Clear Register 0 (modules 0-15) (MERRCR0)	139
8.6.6	PSC1 Module Error Clear Register 0 (modules 0-31) (MERRCR0)	139
8.6.7	Power Error Pending Register (PERRPR)	140
8.6.8	Power Error Clear Register (PERRCR)	140
8.6.9	Power Domain Transition Command Register (PTCMD)	141
8.6.10	Power Domain Transition Status Register (PTSTAT)	142
8.6.11	Power Domain 0 Status Register (PDSTAT0)	143
8.6.12	Power Domain 1 Status Register (PDSTAT1)	144
8.6.13	Power Domain 0 Control Register (PDCTL0)	145
8.6.14	Power Domain 1 Control Register (PDCTL1)	146
8.6.15	Power Domain 0 Configuration Register (PDCFG0)	147
8.6.16	Power Domain 1 Configuration Register (PDCFG1)	148
8.6.17	Module Status <i>n</i> Register (MDSTAT $n$ )	149
8.6.18	PSC0 Module Control <i>n</i> Register (modules 0-15) (MDCTL $n$ )	150
8.6.19	PSC1 Module Control <i>n</i> Register (modules 0-31) (MDCTL $n$ )	151
<b>9</b>	<b>Power Management</b>	<b>152</b>
9.1	Introduction	153
9.2	Power Consumption Overview	153
9.3	PSC and PLL Overview	153
9.4	Features	154
9.5	Clock Management	155
9.5.1	Module Clock ON/OFF	155
9.5.2	Module Clock Frequency Scaling	155
9.5.3	PLL Bypass and Power Down	155
9.6	DSP Sleep Mode Management	156
9.6.1	C674x DSP CPU Sleep Mode	156
9.6.2	C674x Megamodule Sleep Mode	156
9.7	RTC-Only Mode	156

9.8	Dynamic Voltage and Frequency Scaling (DVFS) .....	156
9.8.1	Frequency Scaling Considerations .....	157
9.8.2	Voltage Scaling Considerations.....	158
9.9	Deep Sleep Mode.....	158
9.9.1	Entering/Exiting Deep Sleep Mode Using Externally Controlled Wake-Up .....	158
9.9.2	Entering/Exiting Deep Sleep Mode Using RTC Controlled Wake-Up.....	159
9.9.3	Deep Sleep Sequence .....	160
9.9.4	Entering/Exiting Deep Sleep Mode Using Software Handshaking .....	161
9.10	Additional Peripheral Power Management Considerations.....	161
9.10.1	DDR2/mDDR Memory Controller Clock Gating and Self-Refresh Mode .....	161
9.10.2	LVC MOS I/O Buffer Receiver Disable .....	162
9.10.3	Pull-Up/Pull-Down Disable.....	162
<b>10</b>	<b>System Configuration (SYSCFG) Module .....</b>	<b>163</b>
10.1	Introduction .....	164
10.2	Protection .....	164
10.2.1	Privilege Mode Protection .....	164
10.2.2	Kicker Mechanism Protection .....	165
10.3	Master Priority Control .....	165
10.4	Interrupt Support .....	166
10.4.1	Interrupt Events and Requests.....	166
10.4.2	Interrupt Multiplexing .....	167
10.4.3	Host-DSP Communication Interrupts .....	167
10.5	SYSCFG Registers .....	167
10.5.1	Revision Identification Register (REVID) .....	168
10.5.2	Device Identification Register 0 (DEVIDR0).....	169
10.5.3	Boot Configuration Register (BOOTCFG) .....	169
10.5.4	Chip Revision Identification Register (CHIPREVIDR) .....	169
10.5.5	Kick Registers (KICK0R-KICK1R) .....	171
10.5.6	Host 1 Configuration Register (HOST1CFG) .....	172
10.5.7	Interrupt Registers .....	173
10.5.8	Fault Registers .....	176
10.5.9	Master Priority Registers (MSTPRI0-MSTPRI2).....	178
10.5.10	Pin Multiplexing Control Registers (PINMUX0-PINMUX19) .....	181
10.5.11	Suspend Source Register (SUSPSRC) .....	221
10.5.12	Chip Signal Register (CHIPSIG) .....	223
10.5.13	Chip Signal Clear Register (CHIPSIG_CLR) .....	224
10.5.14	Chip Configuration 0 Register (CFGCHIP0) .....	225
10.5.15	Chip Configuration 1 Register (CFGCHIP1) .....	226
10.5.16	Chip Configuration 3 Register (CFGCHIP3) .....	228
10.5.17	Chip Configuration 4 Register (CFGCHIP4) .....	229
10.5.18	VTP I/O Control Register (VTPIO_CTL) .....	229
10.5.19	DDR Slew Register (DDR_SLEW).....	231
10.5.20	Deep Sleep Register (DEEPSLEEP) .....	232
10.5.21	Pullup/Pulldown Enable Register (PUPD_ENA) .....	233
10.5.22	Pullup/Pulldown Select Register (PUPD_SEL).....	233
10.5.23	RXACTIVE Control Register (RXACTIVE).....	235
<b>11</b>	<b>Boot Considerations .....</b>	<b>236</b>
11.1	Introduction .....	237
<b>12</b>	<b>DDR2/mDDR Memory Controller .....</b>	<b>238</b>
12.1	Introduction .....	239
12.1.1	Purpose of the Peripheral .....	239
12.1.2	Features.....	239
12.1.3	Functional Block Diagram .....	240

12.1.4	Supported Use Case Statement .....	240
12.1.5	Industry Standard(s) Compliance Statement.....	240
12.2	Architecture .....	241
12.2.1	Clock Control .....	241
12.2.2	Signal Descriptions .....	242
12.2.3	Protocol Description(s) .....	243
12.2.4	Memory Width and Byte Alignment .....	251
12.2.5	Address Mapping .....	252
12.2.6	DDR2/mDDR Memory Controller Interface.....	257
12.2.7	Refresh Scheduling.....	260
12.2.8	Self-Refresh Mode.....	260
12.2.9	Partial Array Self Refresh for Mobile DDR .....	261
12.2.10	Power-Down Mode.....	261
12.2.11	Reset Considerations.....	262
12.2.12	VTP IO Buffer Calibration .....	263
12.2.13	Auto-Initialization Sequence .....	263
12.2.14	Interrupt Support .....	266
12.2.15	DMA Event Support.....	266
12.2.16	Power Management .....	267
12.2.17	Emulation Considerations .....	268
12.3	Supported Use Cases .....	269
12.4	Registers .....	274
12.4.1	SDRAM Status Register (SDRSTAT) .....	275
12.4.2	SDRAM Configuration Register (SDCR) .....	276
12.4.3	SDRAM Refresh Control Register (SDRCR).....	279
12.4.4	SDRAM Timing Register 1 (SDTIMR1) .....	280
12.4.5	SDRAM Timing Register 2 (SDTIMR2) .....	281
12.4.6	SDRAM Configuration Register 2 (SDCR2).....	282
12.4.7	Peripheral Bus Burst Priority Register (PBBPR).....	283
12.4.8	Performance Counter 1 Register (PC1) .....	284
12.4.9	Performance Counter 2 Register (PC2) .....	284
12.4.10	Performance Counter Configuration Register (PCC).....	285
12.4.11	Performance Counter Master Region Select Register (PCMRS) .....	287
12.4.12	DDR PHY Reset Control Register (DRPYRCR) .....	288
12.4.13	Interrupt Raw Register (IRR) .....	289
12.4.14	Interrupt Masked Register (IMR) .....	289
12.4.15	Interrupt Mask Set Register (IMSR) .....	290
12.4.16	Interrupt Mask Clear Register (IMCR).....	291
12.4.17	DDR PHY Control Register (DRPYC1R).....	292
<b>13</b>	<b>Enhanced Capture (eCAP) Module .....</b>	<b>293</b>
13.1	Introduction .....	294
13.1.1	Purpose of the Peripheral .....	294
13.1.2	Features.....	294
13.2	Architecture .....	295
13.2.1	Capture and APWM Operating Mode .....	296
13.2.2	Capture Mode Description.....	297
13.3	Applications .....	304
13.3.1	Absolute Time-Stamp Operation Rising Edge Trigger Example .....	305
13.3.2	Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example .....	307
13.3.3	Time Difference (Delta) Operation Rising Edge Trigger Example .....	309
13.3.4	Time Difference (Delta) Operation Rising and Falling Edge Trigger Example .....	311
13.3.5	Application of the APWM Mode.....	313
13.4	Registers.....	320

13.4.1	Time-Stamp Counter Register (TSCTR).....	320
13.4.2	Counter Phase Control Register (CTPHS) .....	321
13.4.3	Capture 1 Register (CAP1).....	321
13.4.4	Capture 2 Register (CAP2).....	322
13.4.5	Capture 3 Register (CAP3).....	322
13.4.6	Capture 4 Register (CAP4).....	323
13.4.7	ECAP Control Register 1 (ECCTL1) .....	323
13.4.8	ECAP Control Register 2 (ECCTL2) .....	325
13.4.9	ECAP Interrupt Enable Register (ECEINT) .....	326
13.4.10	ECAP Interrupt Flag Register (ECFLG) .....	328
13.4.11	ECAP Interrupt Clear Register (ECCLR).....	329
13.4.12	ECAP Interrupt Forcing Register (ECFRC).....	330
13.4.13	Revision ID Register (REVID) .....	331
<b>14</b>	<b>Enhanced High-Resolution Pulse-Width Modulator (eHRPWM).....</b>	<b>332</b>
14.1	Introduction .....	333
14.1.1	Introduction .....	333
14.1.2	Submodule Overview .....	333
14.1.3	Register Mapping .....	337
14.2	Architecture .....	338
14.2.1	Overview .....	338
14.2.2	Proper Interrupt Initialization Procedure .....	341
14.2.3	Time-Base (TB) Submodule.....	342
14.2.4	Counter-Compare (CC) Submodule.....	351
14.2.5	Action-Qualifier (AQ) Submodule.....	356
14.2.6	Dead-Band Generator (DB) Submodule.....	374
14.2.7	PWM-Chopper (PC) Submodule.....	378
14.2.8	Trip-Zone (TZ) Submodule .....	382
14.2.9	Event-Trigger (ET) Submodule .....	386
14.2.10	High-Resolution PWM (HRPWM) Submodule.....	390
14.3	Applications to Power Topologies .....	397
14.3.1	Overview of Multiple Modules .....	397
14.3.2	Key Configuration Capabilities .....	398
14.3.3	Controlling Multiple Buck Converters With Independent Frequencies.....	399
14.3.4	Controlling Multiple Buck Converters With Same Frequencies.....	402
14.3.5	Controlling Multiple Half H-Bridge (HHB) Converters.....	405
14.3.6	Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM) .....	408
14.3.7	Practical Applications Using Phase Control Between PWM Modules .....	412
14.3.8	Controlling a 3-Phase Interleaved DC/DC Converter .....	413
14.3.9	Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter.....	418
14.4	Registers.....	421
14.4.1	Time-Base Submodule Registers .....	421
14.4.2	Counter-Compare Submodule Registers .....	425
14.4.3	Action-Qualifier Submodule Registers .....	428
14.4.4	Dead-Band Generator Submodule Registers .....	432
14.4.5	PWM-Chopper Submodule Register .....	435
14.4.6	Trip-Zone Submodule Registers.....	436
14.4.7	Event-Trigger Submodule Registers .....	440
14.4.8	High-Resolution PWM Submodule Registers .....	443
<b>15</b>	<b>Enhanced Direct Memory Access (EDMA3) Controller.....</b>	<b>446</b>
15.1	Introduction .....	447
15.1.1	Overview .....	447
15.1.2	Features.....	447
15.1.3	Functional Block Diagram .....	450

15.1.4	Terminology Used in This Document .....	450
15.2	Architecture .....	452
15.2.1	Functional Overview.....	452
15.2.2	Types of EDMA3 Transfers .....	455
15.2.3	Parameter RAM (PaRAM) .....	458
15.2.4	Initiating a DMA Transfer .....	468
15.2.5	Completion of a DMA Transfer.....	471
15.2.6	Event, Channel, and PaRAM Mapping .....	472
15.2.7	EDMA3 Channel Controller Regions .....	475
15.2.8	Chaining EDMA3 Channels .....	477
15.2.9	EDMA3 Interrupts.....	478
15.2.10	Event Queue(s).....	485
15.2.11	EDMA3 Transfer Controller (EDMA3TC).....	487
15.2.12	Event Dataflow .....	490
15.2.13	EDMA3 Prioritization.....	491
15.2.14	EDMA3CC and EDMA3TC Performance and System Considerations .....	493
15.2.15	EDMA3 Operating Frequency (Clock Control) .....	494
15.2.16	Reset Considerations .....	494
15.2.17	Power Management .....	494
15.2.18	Emulation Considerations .....	495
15.3	Transfer Examples.....	495
15.3.1	Block Move Example .....	495
15.3.2	Subframe Extraction Example .....	497
15.3.3	Data Sorting Example .....	498
15.3.4	Peripheral Servicing Example.....	500
15.4	Registers.....	512
15.4.1	Parameter RAM (PaRAM) Entries.....	512
15.4.2	EDMA3 Channel Controller (EDMA3CC) Registers.....	519
15.4.3	EDMA3 Transfer Controller (EDMA3TC) Registers .....	558
15.5	Tips .....	579
15.5.1	Debug Checklist .....	579
15.5.2	Miscellaneous Programming/Debug Tips .....	580
15.6	Setting Up a Transfer .....	581
<b>16</b>	<b>External Memory Interface A (EMIFA).....</b>	<b>582</b>
16.1	Introduction .....	583
16.1.1	Purpose of the Peripheral .....	583
16.1.2	Features.....	583
16.1.3	Functional Block Diagram .....	583
16.2	Architecture .....	583
16.2.1	Clock Control .....	584
16.2.2	EMIFA Requests.....	584
16.2.3	Pin Descriptions.....	584
16.2.4	SDRAM Controller and Interface .....	586
16.2.5	Asynchronous Controller and Interface .....	598
16.2.6	Data Bus Parking .....	617
16.2.7	Reset and Initialization Considerations .....	617
16.2.8	Interrupt Support.....	618
16.2.9	EDMA Event Support .....	619
16.2.10	Pin Multiplexing.....	619
16.2.11	Memory Map .....	619
16.2.12	Priority and Arbitration .....	620
16.2.13	System Considerations .....	621
16.2.14	Power Management .....	622



16.2.15	Emulation Considerations .....	623
16.3	Example Configuration .....	624
16.3.1	Hardware Interface .....	624
16.3.2	Software Configuration.....	624
16.4	Registers.....	646
16.4.1	Module ID Register (MIDR) .....	647
16.4.2	Asynchronous Wait Cycle Configuration Register (AWCC).....	647
16.4.3	SDRAM Configuration Register (SDCR) .....	649
16.4.4	SDRAM Refresh Control Register (SDRCR).....	651
16.4.5	Asynchronous <i>n</i> Configuration Registers (CE2CFG-CE5CFG) .....	652
16.4.6	SDRAM Timing Register (SDTIMR).....	654
16.4.7	SDRAM Self Refresh Exit Timing Register (SDSRETR) .....	655
16.4.8	EMIFA Interrupt Raw Register (INTRAW).....	656
16.4.9	EMIFA Interrupt Masked Register (INTMSK) .....	657
16.4.10	EMIFA Interrupt Mask Set Register (INTMSKSET).....	658
16.4.11	EMIFA Interrupt Mask Clear Register (INTMSKCLR) .....	659
16.4.12	NAND Flash Control Register (NANDFCR) .....	660
16.4.13	NAND Flash Status Register (NANDFSR).....	662
16.4.14	NAND Flash <i>n</i> ECC Registers (NANDF1ECC-NANDF4ECC) .....	663
16.4.15	NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) .....	664
16.4.16	NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) .....	665
16.4.17	NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) .....	665
16.4.18	NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) .....	666
16.4.19	NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) .....	666
16.4.20	NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) .....	667
16.4.21	NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) .....	667
16.4.22	NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1).....	668
16.4.23	NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2).....	668
<b>17</b>	<b>General-Purpose Input/Output (GPIO) .....</b>	<b>669</b>
17.1	Introduction .....	670
17.1.1	Purpose of the Peripheral .....	670
17.1.2	Features.....	670
17.1.3	Functional Block Diagram .....	670
17.1.4	Industry Standard(s) Compliance Statement.....	670
17.2	Architecture .....	671
17.2.1	Clock Control .....	671
17.2.2	Signal Descriptions .....	671
17.2.3	Pin Multiplexing .....	671
17.2.4	Endianness Considerations .....	671
17.2.5	GPIO Register Structure.....	672
17.2.6	Using a GPIO Signal as an Output.....	675
17.2.7	Using a GPIO Signal as an Input.....	676
17.2.8	Reset Considerations .....	676
17.2.9	Initialization .....	677
17.2.10	Interrupt Support .....	677
17.2.11	EDMA Event Support .....	678
17.2.12	Power Management .....	678
17.2.13	Emulation Considerations .....	678
17.3	Registers.....	679
17.3.1	Revision ID Register (REVID).....	680
17.3.2	GPIO Interrupt Per-Bank Enable Register (BINTEN) .....	681
17.3.3	GPIO Direction Registers (DIR <i>n</i> ) .....	682
17.3.4	GPIO Output Data Registers (OUT_DATA <i>n</i> ) .....	684

17.3.5	GPIO Set Data Registers (SET_DATA $n$ ) .....	686
17.3.6	GPIO Clear Data Registers (CLR_DATA $n$ ) .....	688
17.3.7	GPIO Input Data Registers (IN_DATA $n$ ) .....	690
17.3.8	GPIO Set Rising Edge Interrupt Registers (SET_RIS_TRIG $n$ ) .....	692
17.3.9	GPIO Clear Rising Edge Interrupt Registers (CLR_RIS_TRIG $n$ ) .....	694
17.3.10	GPIO Set Falling Edge Interrupt Registers (SET_FAL_TRIG $n$ ) .....	696
17.3.11	GPIO Clear Falling Edge Interrupt Registers (CLR_FAL_TRIG $n$ ) .....	698
17.3.12	GPIO Interrupt Status Registers (INTSTAT $n$ ) .....	700
<b>18</b>	<b>Host Port Interface (HPI) .....</b>	<b>702</b>
18.1	Introduction .....	703
18.1.1	Purpose of the Peripheral .....	703
18.1.2	Features.....	703
18.1.3	Functional Block Diagram .....	704
18.1.4	Industry Standard(s) Compliance Statement.....	705
18.1.5	Terminology Used in This Document .....	705
18.2	Architecture .....	706
18.2.1	Clock Control .....	706
18.2.2	Memory Map .....	706
18.2.3	Signal Descriptions .....	706
18.2.4	Pin Multiplexing and General-Purpose I/O Control Blocks .....	707
18.2.5	Protocol Description .....	708
18.2.6	Operation .....	708
18.2.7	Reset Considerations .....	723
18.2.8	Initialization .....	723
18.2.9	Interrupt Support.....	724
18.2.10	EDMA Event Support.....	725
18.2.11	Power Management .....	725
18.2.12	Emulation Considerations .....	726
18.3	Registers.....	726
18.3.1	Revision Identification Register (REVID) .....	727
18.3.2	Power and Emulation Management Register (PWREMU_MGMT) .....	727
18.3.3	GPIO Enable Register (GPIO_EN) .....	728
18.3.4	GPIO Direction 1 Register (GPIO_DIR1).....	729
18.3.5	GPIO Data 1 Register (GPIO_DAT1) .....	729
18.3.6	GPIO Direction 2 Register (GPIO_DIR2).....	730
18.3.7	GPIO Data 2 Register (GPIO_DAT2) .....	731
18.3.8	Host Port Interface Control Register (HPIC) .....	732
18.3.9	Host Port Interface Write Address Register (HPIAW) .....	734
18.3.10	Host Port Interface Read Address Register (HPIAR) .....	734
<b>19</b>	<b>Inter-Integrated Circuit (I2C) Module .....</b>	<b>735</b>
19.1	Introduction .....	736
19.1.1	Purpose of the Peripheral .....	736
19.1.2	Features.....	736
19.1.3	Functional Block Diagram .....	737
19.1.4	Industry Standard(s) Compliance Statement.....	737
19.2	Architecture .....	738
19.2.1	Bus Structure .....	738
19.2.2	Clock Generation .....	739
19.2.3	Clock Synchronization .....	740
19.2.4	Signal Descriptions .....	740
19.2.5	START and STOP Conditions .....	741
19.2.6	Serial Data Formats .....	742
19.2.7	Operating Modes .....	744

19.2.8	NACK Bit Generation .....	745
19.2.9	Arbitration .....	746
19.2.10	Reset Considerations .....	747
19.2.11	Initialization .....	747
19.2.12	Interrupt Support .....	748
19.2.13	DMA Events Generated by the I2C Peripheral .....	749
19.2.14	Power Management .....	749
19.2.15	Emulation Considerations .....	749
19.3	Registers .....	750
19.3.1	I2C Own Address Register (ICOAR) .....	751
19.3.2	I2C Interrupt Mask Register (ICIMR) .....	752
19.3.3	I2C Interrupt Status Register (ICSTR) .....	753
19.3.4	I2C Clock Divider Registers (ICCLKL and ICCLKH) .....	756
19.3.5	I2C Data Count Register (ICCNT) .....	757
19.3.6	I2C Data Receive Register (ICDRR) .....	758
19.3.7	I2C Slave Address Register (ICSAR) .....	759
19.3.8	I2C Data Transmit Register (ICDXR) .....	760
19.3.9	I2C Mode Register (ICMDR) .....	761
19.3.10	I2C Interrupt Vector Register (ICIVR) .....	765
19.3.11	I2C Extended Mode Register (ICEMDR) .....	766
19.3.12	I2C Prescaler Register (ICPSC) .....	767
19.3.13	I2C Revision Identification Register (REVID1) .....	768
19.3.14	I2C Revision Identification Register (REVID2) .....	768
19.3.15	I2C DMA Control Register (ICDMAC) .....	769
19.3.16	I2C Pin Function Register (ICPFUNC) .....	770
19.3.17	I2C Pin Direction Register (ICPDIR) .....	771
19.3.18	I2C Pin Data In Register (ICPDIN) .....	772
19.3.19	I2C Pin Data Out Register (ICPDOUT) .....	773
19.3.20	I2C Pin Data Set Register (ICPDSET) .....	774
19.3.21	I2C Pin Data Clear Register (ICPDCLR) .....	775
<b>20</b>	<b>Multichannel Audio Serial Port (McASP) .....</b>	<b>776</b>
20.0.22	Features .....	777
20.0.23	Protocols Supported .....	778
20.0.24	Functional Block Diagram .....	779
20.0.25	Definition of Terms .....	787
20.0.26	Overview .....	790
20.0.27	Clock and Frame Sync Generators .....	790
20.0.28	Reset Considerations .....	831
20.0.29	EDMA Event Support .....	831
20.0.30	Power Management .....	831
20.1	Registers .....	832
20.1.1	Register Bit Restrictions .....	835
20.1.2	Revision Identification Register (REV) .....	836
20.1.3	Pin Function Register (PFUNC) .....	837
20.1.4	Pin Direction Register (PDIR) .....	839
20.1.5	Pin Data Output Register (PDOUT) .....	841
20.1.6	Pin Data Input Register (PDIN) .....	843
20.1.7	Pin Data Set Register (PDSET) .....	845
20.1.8	Pin Data Clear Register (PDCLR) .....	847
20.1.9	Global Control Register (GBLCTL) .....	849
20.1.10	Audio Mute Control Register (AMUTE) .....	851
20.1.11	Digital Loopback Control Register (DLBCTL) .....	853
20.1.12	Digital Mode Control Register (DITCTL) .....	854

20.1.13	Receiver Global Control Register (RGBLCTL) .....	855
20.1.14	Receive Format Unit Bit Mask Register (RMASK).....	856
20.1.15	Receive Bit Stream Format Register (RFMT) .....	857
20.1.16	Receive Frame Sync Control Register (AFSRCTL) .....	859
20.1.17	Receive Clock Control Register (ACLKRCTL) .....	860
20.1.18	Receive High-Frequency Clock Control Register (AHCLKRCTL) .....	861
20.1.19	Receive TDM Time Slot Register (RTDM).....	862
20.1.20	Receiver Interrupt Control Register (RINTCTL).....	863
20.1.21	Receiver Status Register (RSTAT) .....	864
20.1.22	Current Receive TDM Time Slot Registers (R SLOT) .....	865
20.1.23	Receive Clock Check Control Register (RCLKCHK) .....	866
20.1.24	Receiver DMA Event Control Register (REVTCTL) .....	867
20.1.25	Transmitter Global Control Register (XGBLCTL) .....	868
20.1.26	Transmit Format Unit Bit Mask Register (XMASK) .....	869
20.1.27	Transmit Bit Stream Format Register (XFMT).....	870
20.1.28	Transmit Frame Sync Control Register (AFSXCTL).....	872
20.1.29	Transmit Clock Control Register (ACLKXCTL).....	873
20.1.30	Transmit High-Frequency Clock Control Register (AHCLKXCTL).....	874
20.1.31	Transmit TDM Time Slot Register (XTDM) .....	875
20.1.32	Transmitter Interrupt Control Register (XINTCTL) .....	876
20.1.33	Transmitter Status Register (XSTAT) .....	877
20.1.34	Current Transmit TDM Time Slot Register (XSLOT) .....	878
20.1.35	Transmit Clock Check Control Register (XCLKCHK) .....	879
20.1.36	Transmitter DMA Event Control Register (XEVTCTL) .....	880
20.1.37	Serializer Control Registers (SRCTL <sub>n</sub> ).....	881
20.1.38	DIT Left Channel Status Registers (DITCSRA0-DITCSRA5) .....	882
20.1.39	DIT Right Channel Status Registers (DITCSRB0-DITCSRB5).....	882
20.1.40	DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5) .....	883
20.1.41	DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5).....	883
20.1.42	Transmit Buffer Registers (XBUF <sub>n</sub> ).....	884
20.1.43	Receive Buffer Registers (RBUF <sub>n</sub> ) .....	884
20.1.44	AFIFO Revision Identification Register (AFIFOREV).....	885
20.1.45	Write FIFO Control Register (WFIFOCTL).....	886
20.1.46	Write FIFO Status Register (WFIFOSTS).....	887
20.1.47	Read FIFO Control Register (RFIFOCTL) .....	888
20.1.48	Read FIFO Status Register (RFIFOSTS) .....	889
<b>21</b>	<b>Multichannel Buffered Serial Port (McBSP) .....</b>	<b>890</b>
21.1	Introduction .....	891
21.1.1	Purpose of the Peripheral .....	891
21.1.2	Features.....	891
21.1.3	Functional Block Diagram .....	892
21.1.4	Industry Standard Compliance Statement.....	892
21.2	Architecture .....	893
21.2.1	Clock Control .....	893
21.2.2	Signal Descriptions .....	893
21.2.3	Pin Multiplexing .....	893
21.2.4	Endianness Considerations .....	893
21.2.5	Clock, Frames, and Data .....	894
21.2.6	McBSP Buffer FIFO (BFIFO).....	908
21.2.7	McBSP Standard Operation.....	908
21.2.8	μ-Law/A-Law Companding Hardware Operation .....	922
21.2.9	Multichannel Selection Modes .....	924
21.2.10	SPI Operation Using the Clock Stop Mode .....	932

21.2.11	Resetting the Serial Port: RRST, XRST, GRST, and RESET .....	932
21.2.12	McBSP Initialization Procedure.....	933
21.2.13	Interrupt Support .....	937
21.2.14	EDMA Event Support .....	938
21.2.15	Power Management .....	939
21.2.16	Emulation Considerations .....	939
21.3	Registers .....	940
21.3.1	Data Receive Register (DRR).....	941
21.3.2	Data Transmit Register (DXR) .....	941
21.3.3	Serial Port Control Register (SPCR) .....	942
21.3.4	Receive Control Register (RCR).....	944
21.3.5	Transmit Control Register (XCR) .....	946
21.3.6	Sample Rate Generator Register (SRGR) .....	948
21.3.7	Multichannel Control Register (MCR) .....	949
21.3.8	Enhanced Receive Channel Enable Registers (RCERE0-RCERE3).....	953
21.3.9	Enhanced Transmit Channel Enable Registers (XCERE0-XCERE3) .....	955
21.3.10	Pin Control Register (PCR) .....	957
21.3.11	BFIFO Revision Identification Register (BFIFOREV).....	959
21.3.12	Write FIFO Control Register (WFIFOCTL).....	960
21.3.13	Write FIFO Status Register (WFIFOSTS).....	961
21.3.14	Read FIFO Control Register (RFIFOCTL) .....	962
21.3.15	Read FIFO Status Register (RFIFOSTS) .....	963
<b>22</b>	<b>Real-Time Clock (RTC) .....</b>	<b>964</b>
22.1	Introduction .....	965
22.1.1	Purpose of the Peripheral .....	965
22.1.2	Features.....	965
22.1.3	Block Diagram.....	965
22.2	Architecture .....	966
22.2.1	Clock Source .....	966
22.2.2	Signal Descriptions .....	966
22.2.3	Isolated Power Supply .....	966
22.2.4	Operation .....	967
22.2.5	Interrupt Requests .....	969
22.2.6	Register Protection Against Spurious Writes .....	970
22.2.7	General-Purpose Scratch Registers .....	971
22.2.8	Real-Time Clock Response to Low Power Modes (Idle Configurations).....	971
22.2.9	Emulation Modes of the Real-Time Clock.....	971
22.2.10	Reset Considerations.....	971
22.3	Registers .....	972
22.3.1	Second Register (SECOND) .....	973
22.3.2	Minute Register (MINUTE) .....	973
22.3.3	Hour Register (HOUR) .....	974
22.3.4	Day of the Month Register (DAY) .....	975
22.3.5	Month Register (MONTH) .....	975
22.3.6	Year Register (YEAR).....	976
22.3.7	Day of the Week Register (DOTW) .....	976
22.3.8	Alarm Second Register (ALARMSECOND).....	977
22.3.9	Alarm Minute Register (ALARMMINUTE) .....	977
22.3.10	Alarm Hour Register (ALARMHOUR) .....	978
22.3.11	Alarm Day of the Month Register (ALARMDAY).....	979
22.3.12	Alarm Month Register (ALARMMONTH).....	980
22.3.13	Alarm Year Register (ALARMYEAR) .....	980
22.3.14	Control Register (CTRL) .....	981

22.3.15	Status Register (STATUS) .....	982
22.3.16	Interrupt Register (INTERRUPT) .....	983
22.3.17	Compensation (LSB) Register (COMPLSB) .....	984
22.3.18	Compensation (MSB) Register (COMPMSB) .....	985
22.3.19	Oscillator Register (OSC) .....	986
22.3.20	Scratch Registers (SCRATCH0-SCRATCH2) .....	987
22.3.21	Kick Registers (KICK0R, KICK1R) .....	987
<b>23</b>	<b>Serial Peripheral Interface (SPI) .....</b>	<b>988</b>
23.1	Introduction .....	989
23.1.1	Purpose of the Peripheral .....	989
23.1.2	Features .....	989
23.1.3	Functional Block Diagram .....	990
23.1.4	Industry Standard(s) Compliance Statement .....	990
23.2	Architecture .....	991
23.2.1	Clock .....	991
23.2.2	Signal Descriptions .....	991
23.2.3	Operation Modes .....	991
23.2.4	Programmable Registers .....	992
23.2.5	Master Mode Settings .....	993
23.2.6	Slave Mode Settings .....	995
23.2.7	SPI Operation: 3-Pin Mode .....	996
23.2.8	SPI Operation: 4-Pin with Chip Select Mode .....	997
23.2.9	SPI Operation: 4-Pin with Enable Mode .....	999
23.2.10	SPI Operation: 5-Pin Mode .....	1001
23.2.11	Data Formats .....	1003
23.2.12	Interrupt Support .....	1006
23.2.13	DMA Events Support .....	1007
23.2.14	Robustness Features .....	1007
23.2.15	Reset Considerations .....	1009
23.2.16	Power Management .....	1009
23.2.17	General-Purpose I/O Pin .....	1010
23.2.18	Emulation Considerations .....	1010
23.2.19	Initialization .....	1010
23.2.20	Timing Diagrams .....	1011
23.3	Registers .....	1017
23.3.1	SPI Global Control Register 0 (SPIGCR0) .....	1017
23.3.2	SPI Global Control Register 1 (SPIGCR1) .....	1018
23.3.3	SPI Interrupt Register (SPIINT0) .....	1020
23.3.4	SPI Interrupt Level Register (SPILVL) .....	1022
23.3.5	SPI Flag Register (SPIFLG) .....	1023
23.3.6	SPI Pin Control Register 0 (SPIPC0) .....	1025
23.3.7	SPI Pin Control Register 1 (SPIPC1) .....	1026
23.3.8	SPI Pin Control Register 2 (SPIPC2) .....	1027
23.3.9	SPI Pin Control Register 3 (SPIPC3) .....	1028
23.3.10	SPI Pin Control Register 4 (SPIPC4) .....	1029
23.3.11	SPI Pin Control Register 5 (SPIPC5) .....	1030
23.3.12	SPI Transmit Data Register 0 (SPIDAT0) .....	1031
23.3.13	SPI Transmit Data Register 1 (SPIDAT1) .....	1032
23.3.14	SPI Receive Buffer Register (SPIBUF) .....	1033
23.3.15	SPI Emulation Register (SPIEMU) .....	1035
23.3.16	SPI Delay Register (SPIDELAY) .....	1036
23.3.17	SPI Default Chip Select Register (SPIDEF) .....	1039
23.3.18	SPI Data Format Registers (SPIFMT $n$ ) .....	1040



	23.3.19 SPI Interrupt Vector Register 1 (INTVEC1) .....	1042
<b>24</b>	<b>64-Bit Timer Plus</b> .....	<b>1043</b>
	24.1 Introduction .....	1044
	24.1.1 Purpose of the Peripheral .....	1044
	24.1.2 Features .....	1044
	24.1.3 Block Diagram .....	1045
	24.1.4 Industry Standard Compatibility Statement .....	1045
	24.1.5 Architecture – General-Purpose Timer Mode .....	1045
	24.1.6 Architecture – Watchdog Timer Mode .....	1057
	24.1.7 Reset Considerations .....	1059
	24.1.8 Interrupt Support .....	1059
	24.1.9 DMA Event Support .....	1059
	24.1.10 TM64P_OUT Event Support .....	1060
	24.1.11 Interrupt/DMA Event Generation Control and Status .....	1061
	24.1.12 Power Management .....	1061
	24.1.13 Emulation Considerations .....	1061
	24.2 Registers .....	1062
	24.2.1 Revision ID Register (REVID) .....	1064
	24.2.2 Emulation Management Register (EMUMGT) .....	1064
	24.2.3 GPIO Interrupt Control and Enable Register (GPINTGPEN) .....	1065
	24.2.4 GPIO Data and Direction Register (GPDATGPDIR) .....	1066
	24.2.5 Timer Counter Registers (TIM12 and TIM34) .....	1067
	24.2.6 Timer Period Registers (PRD12 and PRD34) .....	1068
	24.2.7 Timer Control Register (TCR) .....	1069
	24.2.8 Timer Global Control Register (TGCR) .....	1071
	24.2.9 Watchdog Timer Control Register (WDTCR) .....	1072
	24.2.10 Timer Reload Register 12 (REL12) .....	1073
	24.2.11 Timer Reload Register 34 (REL34) .....	1073
	24.2.12 Timer Capture Register 12 (CAP12) .....	1074
	24.2.13 Timer Capture Register 34 (CAP34) .....	1074
	24.2.14 Timer Interrupt Control and Status Register (INTCTLSTAT) .....	1075
<b>25</b>	<b>Universal Asynchronous Receiver/Transmitter (UART)</b> .....	<b>1077</b>
	25.1 Introduction .....	1078
	25.1.1 Purpose of the Peripheral .....	1078
	25.1.2 Features .....	1078
	25.1.3 Functional Block Diagram .....	1078
	25.1.4 Industry Standard(s) Compliance Statement .....	1078
	25.2 Peripheral Architecture .....	1080
	25.2.1 Clock Generation and Control .....	1080
	25.2.2 Signal Descriptions .....	1082
	25.2.3 Pin Multiplexing .....	1082
	25.2.4 Protocol Description .....	1082
	25.2.5 Operation .....	1084
	25.2.6 Reset Considerations .....	1088
	25.2.7 Initialization .....	1088
	25.2.8 Interrupt Support .....	1088
	25.2.9 DMA Event Support .....	1090
	25.2.10 Power Management .....	1090
	25.2.11 Emulation Considerations .....	1090
	25.2.12 Exception Processing .....	1090
	25.3 Registers .....	1091
	25.3.1 Receiver Buffer Register (RBR) .....	1092
	25.3.2 Transmitter Holding Register (THR) .....	1093

---

25.3.3	Interrupt Enable Register (IER) .....	1094
25.3.4	Interrupt Identification Register (IIR).....	1095
25.3.5	FIFO Control Register (FCR) .....	1096
25.3.6	Line Control Register (LCR) .....	1098
25.3.7	Modem Control Register (MCR).....	1100
25.3.8	Line Status Register (LSR) .....	1101
25.3.9	Modem Status Register (MSR).....	1104
25.3.10	Scratch Pad Register (SCR) .....	1105
25.3.11	Divisor Latches (DLL and DLH).....	1105
25.3.12	Revision Identification Registers (REVID1 and REVID2) .....	1107
25.3.13	Power and Emulation Management Register (PWREMU_MGMT) .....	1108
25.3.14	Mode Definition Register (MDR).....	1109
<b>Revision History</b>	.....	<b>1110</b>



## List of Figures

1-1.	TMS320C6742 DSP Block Diagram .....	52
2-1.	TMS320C674x Megamodule Block Diagram .....	54
3-1.	System Interconnect Block Diagram .....	62
5-1.	MPU Block Diagram .....	66
5-2.	Permission Fields .....	68
5-3.	Revision ID Register (REVID) .....	72
5-4.	Configuration Register (CONFIG) .....	73
5-5.	Interrupt Raw Status/Set Register (IRAWSTAT) .....	74
5-6.	Interrupt Enable Status/Clear Register (IENSTAT) .....	75
5-7.	Interrupt Enable Set Register (IENSET) .....	76
5-8.	Interrupt Enable Clear Register (IENCLR) .....	76
5-9.	Fixed Range Start Address Register (FXD_MPSAR) .....	77
5-10.	Fixed Range End Address Register (FXD_MPEAR) .....	77
5-11.	Fixed Range Memory Protection Page Attributes Register (FXD_MPPA) .....	78
5-12.	MPU2 Programmable Range <i>n</i> Start Address Register (PROG <sub><i>n</i></sub> _MPSAR) .....	79
5-13.	MPU2 Programmable Range <i>n</i> End Address Register (PROG <sub><i>n</i></sub> _MPEAR) .....	80
5-14.	Programmable Range Memory Protection Page Attributes Register (PROG <sub><i>n</i></sub> _MPPA) .....	81
5-15.	Fault Address Register (FLTADDRR) .....	82
5-16.	Fault Status Register (FLTSTAT) .....	83
5-17.	Fault Clear Register (FLTCLR) .....	84
6-1.	Overall Clocking Diagram .....	87
6-2.	DDR2/mDDR Memory Controller Clocking Diagram .....	90
6-3.	EMIFA Clocking Diagram .....	91
6-4.	McASP Clocking Diagram .....	92
7-1.	PLL Structure .....	96
7-2.	PLL0 Revision Identification Register (REVID) .....	101
7-3.	PLL1 Revision Identification Register (REVID) .....	102
7-4.	Reset Type Status Register (RSTYPE) .....	102
7-5.	Reset Control Register (RSCTRL) .....	103
7-6.	PLL0 Control Register (PLLCTL) .....	104
7-7.	PLL1 Control Register (PLLCTL) .....	105
7-8.	PLL0 OBSCLK Select Register (OCSEL) .....	106
7-9.	PLL1 OBSCLK Select Register (OCSEL) .....	107
7-10.	PLL Multiplier Control Register (PLLM) .....	108
7-11.	PLL0 Pre-Divider Control Register (PREDIV) .....	108
7-12.	PLL0 Divider 1 Register (PLLDIV1) .....	109
7-13.	PLL1 Divider 1 Register (PLLDIV1) .....	109
7-14.	PLL0 Divider 2 Register (PLLDIV2) .....	110
7-15.	PLL1 Divider 2 Register (PLLDIV2) .....	110
7-16.	PLL0 Divider 3 Register (PLLDIV3) .....	111
7-17.	PLL1 Divider 3 Register (PLLDIV3) .....	111
7-18.	PLL0 Divider 4 Register (PLLDIV4) .....	112
7-19.	PLL0 Divider 5 Register (PLLDIV5) .....	112
7-20.	PLL0 Divider 6 Register (PLLDIV6) .....	113
7-21.	PLL0 Divider 7 Register (PLLDIV7) .....	113
7-22.	PLL0 Oscillator Divider 1 Register (OSCDIV) .....	114
7-23.	PLL1 Oscillator Divider 1 Register (OSCDIV) .....	114

7-24.	PLL Post-Divider Control Register (POSTDIV) .....	115
7-25.	PLL Controller Command Register (PLLCMD) .....	115
7-26.	PLL Controller Status Register (PLLSTAT) .....	116
7-27.	PLLC0 Clock Align Control Register (ALNCTL) .....	117
7-28.	PLLC1 Clock Align Control Register (ALNCTL) .....	118
7-29.	PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) .....	119
7-30.	PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) .....	120
7-31.	PLLC0 Clock Enable Control Register (CKEN) .....	121
7-32.	PLLC1 Clock Enable Control Register (CKEN) .....	121
7-33.	PLLC0 Clock Status Register (CKSTAT) .....	122
7-34.	PLLC1 Clock Status Register (CKSTAT) .....	123
7-35.	PLLC0 SYSCLK Status Register (SYSTAT) .....	124
7-36.	PLLC1 SYSCLK Status Register (SYSTAT) .....	125
7-37.	Emulation Performance Counter 0 Register (EMUCNT0) .....	126
7-38.	Emulation Performance Counter 1 Register (EMUCNT1) .....	126
8-1.	Revision Identification Register (REVID) .....	137
8-2.	Interrupt Evaluation Register (INTEVAL) .....	137
8-3.	PSC0 Module Error Pending Register 0 (MERRPR0) .....	138
8-4.	PSC1 Module Error Pending Register 0 (MERRPR0) .....	138
8-5.	PSC0 Module Error Clear Register 0 (MERRCR0) .....	139
8-6.	PSC1 Module Error Clear Register 0 (MERRCR0) .....	139
8-7.	Power Error Pending Register (PERRPR) .....	140
8-8.	Power Error Clear Register (PERRCR) .....	140
8-9.	Power Domain Transition Command Register (PTCMD) .....	141
8-10.	Power Domain Transition Status Register (PTSTAT) .....	142
8-11.	Power Domain 0 Status Register (PDSTAT0) .....	143
8-12.	Power Domain 1 Status Register (PDSTAT1) .....	144
8-13.	Power Domain 0 Control Register (PDCTL0) .....	145
8-14.	Power Domain 1 Control Register (PDCTL1) .....	146
8-15.	Power Domain 0 Configuration Register (PDCFG0) .....	147
8-16.	Power Domain 1 Configuration Register (PDCFG1) .....	148
8-17.	Module Status <i>n</i> Register (MDSTAT <i>n</i> ) .....	149
8-18.	PSC0 Module Control <i>n</i> Register (MDCTL <i>n</i> ) .....	150
8-19.	PSC1 Module Control <i>n</i> Register (MDCTL <i>n</i> ) .....	151
9-1.	Deep Sleep Mode Sequence .....	160
10-1.	Revision Identification Register (REVID) .....	168
10-2.	Device Identification Register 0 (DEVIDR0) .....	169
10-3.	Boot Configuration Register (BOOTCFG) .....	169
10-4.	Chip Revision Identification Register (CHIPREVIDR) .....	169
10-5.	Kick 0 Register (KICK0R) .....	171
10-6.	Kick 1 Register (KICK1R) .....	171
10-7.	Host 1 Configuration Register (HOST1CFG) .....	172
10-8.	Interrupt Raw Status/Set Register (IRAWSTAT) .....	173
10-9.	Interrupt Enable Status/Clear Register (IENSTAT) .....	174
10-10.	Interrupt Enable Register (IENSET) .....	175
10-11.	Interrupt Enable Clear Register (IENCLR) .....	175
10-12.	End of Interrupt Register (EOI) .....	176
10-13.	Fault Address Register (FLTADDRR) .....	176
10-14.	Fault Status Register (FLTSTAT) .....	177

10-15. Master Priority 0 Register (MSTPRI0) .....	178
10-16. Master Priority 1 Register (MSTPRI1) .....	179
10-17. Master Priority 2 Register (MSTPRI2) .....	180
10-18. Pin Multiplexing Control 0 Register (PINMUX0) .....	181
10-19. Pin Multiplexing Control 1 Register (PINMUX1) .....	183
10-20. Pin Multiplexing Control 2 Register (PINMUX2) .....	185
10-21. Pin Multiplexing Control 3 Register (PINMUX3) .....	187
10-22. Pin Multiplexing Control 4 Register (PINMUX4) .....	189
10-23. Pin Multiplexing Control 5 Register (PINMUX5) .....	191
10-24. Pin Multiplexing Control 6 Register (PINMUX6) .....	193
10-25. Pin Multiplexing Control 7 Register (PINMUX7) .....	195
10-26. Pin Multiplexing Control 8 Register (PINMUX8) .....	197
10-27. Pin Multiplexing Control 9 Register (PINMUX9) .....	199
10-28. Pin Multiplexing Control 10 Register (PINMUX10) .....	201
10-29. Pin Multiplexing Control 11 Register (PINMUX11) .....	203
10-30. Pin Multiplexing Control 12 Register (PINMUX12) .....	205
10-31. Pin Multiplexing Control 13 Register (PINMUX13) .....	207
10-32. Pin Multiplexing Control 14 Register (PINMUX14) .....	209
10-33. Pin Multiplexing Control 15 Register (PINMUX15) .....	211
10-34. Pin Multiplexing Control 16 Register (PINMUX16) .....	213
10-35. Pin Multiplexing Control 17 Register (PINMUX17) .....	215
10-36. Pin Multiplexing Control 18 Register (PINMUX18) .....	217
10-37. Pin Multiplexing Control 19 Register (PINMUX19) .....	219
10-38. Suspend Source Register (SUSPSRC) .....	221
10-39. Chip Signal Register (CHIPSIG) .....	223
10-40. Chip Signal Clear Register (CHIPSIG_CLR) .....	224
10-41. Chip Configuration 0 Register (CFGCHIP0) .....	225
10-42. Chip Configuration 1 Register (CFGCHIP1) .....	226
10-43. Chip Configuration 3 Register (CFGCHIP3) .....	228
10-44. Chip Configuration 4 Register (CFGCHIP4) .....	229
10-45. VTP I/O Control Register (VTPIO_CTL) .....	229
10-46. DDR Slew Register (DDR_SLEW) .....	231
10-47. Deep Sleep Register (DEEPSLEEP) .....	232
10-48. Pullup/Pulldown Enable Register (PUPD_ENA) .....	233
10-49. Pullup/Pulldown Select Register (PUPD_SEL) .....	233
10-50. RXACTIVE Control Register (RXACTIVE) .....	235
12-1. Data Paths to DDR2/mDDR Memory Controller .....	240
12-2. DDR2/mDDR Memory Controller Clock Block Diagram .....	241
12-3. DDR2/mDDR Memory Controller Signals .....	242
12-4. Refresh Command .....	245
12-5. DCAB Command .....	246
12-6. DEAC Command .....	247
12-7. ACTV Command .....	248
12-8. DDR2/mDDR READ Command .....	249
12-9. DDR2/mDDR WRT Command .....	250
12-10. DDR2/mDDR MRS and EMRS Command .....	251
12-11. Byte Alignment .....	251
12-12. DDR2/mDDR SDRAM Column, Row, and Bank Access .....	255
12-13. Address Mapping Diagram (IBANKPOS = 1) .....	256

12-14. SDRAM Column, Row, Bank Access (IBANKPOS = 1) .....	257
12-15. DDR2/mDDR Memory Controller FIFO Block Diagram .....	258
12-16. DDR2/mDDR Memory Controller Reset Block Diagram .....	262
12-17. DDR2/mDDR Memory Controller Power Sleep Controller Diagram .....	267
12-18. Connecting DDR2/mDDR Memory Controller to a 16-Bit DDR2 Memory .....	269
12-19. Revision ID Register (REVID) .....	274
12-20. SDRAM Status Register (SDRSTAT) .....	275
12-21. SDRAM Configuration Register (SDCR) .....	276
12-22. SDRAM Refresh Control Register (SDRCR) .....	279
12-23. SDRAM Timing Register 1 (SDTIMR1) .....	280
12-24. SDRAM Timing Register 2 (SDTIMR2) .....	281
12-25. SDRAM Configuration Register 2 (SDCR2) .....	282
12-26. Peripheral Bus Burst Priority Register (PBBPR) .....	283
12-27. Performance Counter 1 Register (PC1) .....	284
12-28. Performance Counter 2 Register (PC2) .....	284
12-29. Performance Counter Configuration Register (PCC) .....	285
12-30. Performance Counter Master Region Select Register (PCMRS) .....	287
12-31. Performance Counter Time Register (PCT) .....	288
12-32. DDR PHY Reset Control Register (DRPYRCR) .....	288
12-33. Interrupt Raw Register (IRR) .....	289
12-34. Interrupt Masked Register (IMR) .....	289
12-35. Interrupt Mask Set Register (IMSR) .....	290
12-36. Interrupt Mask Clear Register (IMCR) .....	291
12-37. DDR PHY Control Register 1 (DRPYC1R) .....	292
13-1. Multiple eCAP Modules .....	295
13-2. Capture and APWM Modes of Operation .....	296
13-3. Capture Function Diagram .....	297
13-4. Event Prescale Control .....	298
13-5. Prescale Function Waveforms .....	298
13-6. Continuous/One-shot Block Diagram .....	299
13-7. Counter and Synchronization Block Diagram .....	300
13-8. Interrupts in eCAP Module .....	302
13-9. PWM Waveform Details Of APWM Mode Operation .....	303
13-10. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect .....	305
13-11. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect .....	307
13-12. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect .....	309
13-13. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect .....	311
13-14. PWM Waveform Details of APWM Mode Operation .....	313
13-15. Multichannel PWM Example Using 4 eCAP Modules .....	315
13-16. Multiphase (channel) Interleaved PWM Example Using 3 eCAP Modules .....	318
13-17. Time-Stamp Counter Register (TSCTR) .....	320
13-18. Counter Phase Control Register (CTRPHS) .....	321
13-19. Capture 1 Register (CAP1) .....	321
13-20. Capture 2 Register (CAP2) .....	322
13-21. Capture 3 Register (CAP3) .....	322
13-22. Capture 4 Register (CAP4) .....	323
13-23. ECAP Control Register 1 (ECCTL1) .....	323
13-24. ECAP Control Register 2 (ECCTL2) .....	325
13-25. ECAP Interrupt Enable Register (ECEINT) .....	327

13-26. ECAP Interrupt Flag Register (ECFLG).....	328
13-27. ECAP Interrupt Clear Register (ECCLR) .....	329
13-28. ECAP Interrupt Forcing Register (ECFRC).....	330
13-29. Revision ID Register (REVID).....	331
14-1. Multiple ePWM Modules .....	334
14-2. Submodules and Signal Connections for an ePWM Module.....	335
14-3. ePWM Submodules and Critical Internal Signal Interconnects .....	336
14-4. Time-Base Submodule Block Diagram .....	342
14-5. Time-Base Submodule Signals and Registers.....	343
14-6. Time-Base Frequency and Period.....	345
14-7. Time-Base Counter Synchronization Scheme 1 .....	346
14-8. Time-Base Up-Count Mode Waveforms .....	348
14-9. Time-Base Down-Count Mode Waveforms .....	349
14-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down on Synchronization Event.....	349
14-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up on Synchronization Event .....	350
14-12. Counter-Compare Submodule .....	351
14-13. Counter-Compare Submodule Signals and Registers .....	351
14-14. Counter-Compare Event Waveforms in Up-Count Mode.....	354
14-15. Counter-Compare Events in Down-Count Mode .....	354
14-16. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down on Synchronization Event .....	355
14-17. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up on Synchronization Event .....	355
14-18. Action-Qualifier Submodule .....	356
14-19. Action-Qualifier Submodule Inputs and Outputs.....	357
14-20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs .....	358
14-21. Up-Down-Count Mode Symmetrical Waveform.....	361
14-22. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High .....	362
14-23. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low .....	364
14-24. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA.....	366
14-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low .....	368
14-26. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary .....	370
14-27. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low .....	372
14-28. Dead-Band Generator Submodule .....	374
14-29. Configuration Options for the Dead-Band Generator Submodule.....	375
14-30. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%) .....	377
14-31. PWM-Chopper Submodule .....	378
14-32. PWM-Chopper Submodule Signals and Registers .....	379
14-33. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only .....	380
14-34. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses .....	380
14-35. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses .....	381
14-36. Trip-Zone Submodule .....	382
14-37. Trip-Zone Submodule Mode Control Logic.....	385
14-38. Trip-Zone Submodule Interrupt Logic .....	385
14-39. Event-Trigger Submodule .....	386

14-40. Event-Trigger Submodule Inter-Connectivity to Interrupt Controller .....	387
14-41. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs .....	387
14-42. Event-Trigger Interrupt Generator .....	389
14-43. HRPWM System Interface.....	390
14-44. Resolution Calculations for Conventionally Generated PWM.....	391
14-45. Operating Logic Using MEP .....	392
14-46. Required PWM Waveform for a Requested Duty = 40.5% .....	394
14-47. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz .....	396
14-48. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz .....	396
14-49. Simplified ePWM Module .....	397
14-50. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave .....	398
14-51. Control of Four Buck Stages. (Note: $F_{P_{WM1}} \neq F_{P_{WM2}} \neq F_{P_{WM3}} \neq F_{P_{WM4}}$ ) .....	399
14-52. Buck Waveforms for (Note: Only three bucks shown here) .....	400
14-53. Control of Four Buck Stages. (Note: $F_{P_{WM2}} = N \times F_{P_{WM1}}$ ) .....	402
14-54. Buck Waveforms for (Note: $F_{P_{WM2}} = F_{P_{WM1}}$ ).....	403
14-55. Control of Two Half-H Bridge Stages ( $F_{P_{WM2}} = N \times F_{P_{WM1}}$ ) .....	405
14-56. Half-H Bridge Waveforms for (Note: $F_{P_{WM2}} = F_{P_{WM1}}$ ) .....	406
14-57. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control.....	408
14-58. 3-Phase Inverter Waveforms for (Only One Inverter Shown).....	409
14-59. Configuring Two PWM Modules for Phase Control .....	412
14-60. Timing Waveforms Associated With Phase Control Between 2 Modules .....	413
14-61. Control of a 3-Phase Interleaved DC/DC Converter .....	414
14-62. 3-Phase Interleaved DC/DC Converter Waveforms for .....	415
14-63. Controlling a Full-H Bridge Stage ( $F_{P_{WM2}} = F_{P_{WM1}}$ ).....	418
14-64. ZVS Full-H Bridge Waveforms .....	419
14-65. Time-Base Control Register (TBCTL).....	421
14-66. Time-Base Status Register (TBSTS).....	423
14-67. Time-Base Phase Register (TBPHS) .....	424
14-68. Time-Base Counter Register (TBCNT) .....	424
14-69. Time-Base Period Register (TBPRD) .....	425
14-70. Counter-Compare Control Register (CMPCTL).....	426
14-71. Counter-Compare A Register (CMPA) .....	427
14-72. Counter-Compare B Register (CMPB).....	428
14-73. Action-Qualifier Output A Control Register (AQCTLA).....	429
14-74. Action-Qualifier Output B Control Register (AQCTLB).....	430
14-75. Action-Qualifier Software Force Register (AQSFRC) .....	431
14-76. Action-Qualifier Continuous Software Force Register (AQCSFRC).....	432
14-77. Dead-Band Generator Control Register (DBCTL).....	433
14-78. Dead-Band Generator Rising Edge Delay Register (DBRED).....	434
14-79. Dead-Band Generator Falling Edge Delay Register (DBFED) .....	434
14-80. PWM-Chopper Control Register (PCCTL).....	435
14-81. Trip-Zone Select Register (TZSEL) .....	436
14-82. Trip-Zone Control Register (TZCTL) .....	437
14-83. Trip-Zone Enable Interrupt Register (TZEINT).....	437
14-84. Trip-Zone Flag Register (TZFLG).....	438
14-85. Trip-Zone Clear Register (TZCLR) .....	439
14-86. Trip-Zone Force Register (TZFRC).....	439
14-87. Event-Trigger Selection Register (ETSEL) .....	440
14-88. Event-Trigger Prescale Register (ETPS) .....	441



14-89. Event-Trigger Flag Register (ETFLG).....	442
14-90. Event-Trigger Clear Register (ETCLR) .....	442
14-91. Event-Trigger Force Register (ETFRC).....	443
14-92. Time-Base Phase High-Resolution Register (TBPHSHR) .....	444
14-93. Counter-Compare A High-Resolution Register (CMPAHR) .....	444
14-94. HRPWM Configuration Register (HRCNFG) .....	445
15-1. EDMA3 Controller Block Diagram .....	450
15-2. EDMA3 Channel Controller (EDMA3CC) Block Diagram .....	453
15-3. EDMA3 Transfer Controller (EDMA3TC) Block Diagram.....	454
15-4. Definition of ACNT, BCNT, and CCNT .....	455
15-5. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3) .....	456
15-6. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3) .....	457
15-7. PaRAM Set .....	458
15-8. Linked Transfer Example .....	466
15-9. Link-to-Self Transfer Example .....	467
15-10. QDMA Channel to PaRAM Mapping .....	474
15-11. Shadow Region Registers .....	476
15-12. Interrupt Diagram .....	481
15-13. Error Interrupt Operation.....	484
15-14. EDMA3 Prioritization .....	491
15-15. Block Move Example .....	495
15-16. Block Move Example PaRAM Configuration .....	496
15-17. Subframe Extraction Example .....	497
15-18. Subframe Extraction Example PaRAM Configuration.....	497
15-19. Data Sorting Example .....	498
15-20. Data Sorting Example PaRAM Configuration .....	499
15-21. Servicing Incoming McBSP Data Example .....	500
15-22. Servicing Incoming McBSP Data Example PaRAM.....	501
15-23. Servicing Peripheral Burst Example.....	502
15-24. Servicing Peripheral Burst Example PaRAM.....	502
15-25. Servicing Continuous McBSP Data Example .....	503
15-26. Servicing Continuous McBSP Data Example PaRAM .....	504
15-27. Servicing Continuous McBSP Data Example Reload PaRAM.....	504
15-28. Ping-Pong Buffering for McBSP Data Example .....	507
15-29. Ping-Pong Buffering for McBSP Example PaRAM .....	508
15-30. Ping-Pong Buffering for McBSP Example Pong PaRAM.....	508
15-31. Ping-Pong Buffering for McBSP Example Ping PaRAM.....	509
15-32. Intermediate Transfer Completion Chaining Example .....	511
15-33. Single Large Block Transfer Example .....	511
15-34. Smaller Packet Data Transfers Example .....	512
15-35. Channel Options Parameter (OPT).....	513
15-36. Channel Source Address Parameter (SRC) .....	515
15-37. A Count/B Count Parameter (A_B_CNT).....	515
15-38. Channel Destination Address Parameter (DST).....	516
15-39. Source B Index/Destination B Index Parameter (SRC_DST_BIDX) .....	516
15-40. Link Address/B Count Reload Parameter (LINK_BCNTRLD) .....	517
15-41. Source C Index/Destination C Index Parameter (SRC_DST_CIDX).....	518
15-42. C Count Parameter (CCNT).....	518
15-43. Revision ID Register (REVID).....	522

15-44. EDMA3CC Configuration Register (CCCFG) .....	522
15-45. QDMA Channel <i>n</i> Mapping Register (QCHMAP <i>n</i> ) .....	524
15-46. DMA Channel Queue Number Register <i>n</i> (DMAQNUM <i>n</i> ).....	525
15-47. QDMA Channel Queue Number Register (QDMAQNUM) .....	526
15-48. Event Missed Register (EMR).....	527
15-49. Event Missed Clear Register (EMCR) .....	528
15-50. QDMA Event Missed Register (QEMR).....	529
15-51. QDMA Event Missed Clear Register (QEMCR) .....	530
15-52. EDMA3CC Error Register (CCERR) .....	531
15-53. EDMA3CC Error Clear Register (CCERRCLR).....	532
15-54. Error Evaluate Register (EEVAL).....	533
15-55. DMA Region Access Enable Register for Region <i>m</i> (DRAEm).....	534
15-56. QDMA Region Access Enable for Region <i>m</i> (QRAEm) .....	535
15-57. Event Queue Entry Registers (QxEy) .....	536
15-58. Queue <i>n</i> Status Register (QSTAT <i>n</i> ) .....	537
15-59. Queue Watermark Threshold A Register (QWMTHRA) .....	538
15-60. EDMA3CC Status Register (CCSTAT) .....	539
15-61. Event Register (ER) .....	541
15-62. Event Clear Register (ECR) .....	542
15-63. Event Set Register (ESR).....	543
15-64. Chained Event Register (CER) .....	544
15-65. Event Enable Register (EER) .....	545
15-66. Event Enable Clear Register (EECR) .....	546
15-67. Event Enable Set Register (EESR) .....	546
15-68. Secondary Event Register (SER).....	547
15-69. Secondary Event Clear Register (SECR) .....	547
15-70. Interrupt Enable Register (IER) .....	548
15-71. Interrupt Enable Clear Register (IECR).....	549
15-72. Interrupt Enable Set Register (IESR) .....	549
15-73. Interrupt Pending Register (IPR).....	550
15-74. Interrupt Clear Register (ICR).....	551
15-75. Interrupt Evaluate Register (IEVAL) .....	552
15-76. QDMA Event Register (QER) .....	553
15-77. QDMA Event Enable Register (QEER) .....	554
15-78. QDMA Event Enable Clear Register (QEECR) .....	555
15-79. QDMA Event Enable Set Register (QEESR) .....	555
15-80. QDMA Secondary Event Register (QSER).....	556
15-81. QDMA Secondary Event Clear Register (QSECR) .....	557
15-82. Revision ID Register (REVID).....	559
15-83. EDMA3TC Configuration Register (TCCFG).....	560
15-84. EDMA3TC Channel Status Register (TCSTAT) .....	561
15-85. Error Status Register (ERRSTAT).....	562
15-86. Error Enable Register (ERREN) .....	563
15-87. Error Clear Register (ERRCLR) .....	564
15-88. Error Details Register (ERRDET).....	565
15-89. Error Interrupt Command Register (ERRCMD).....	566
15-90. Read Command Rate Register (RDRATE).....	567
15-91. Source Active Options Register (SAOPT).....	568
15-92. Source Active Source Address Register (SASRC).....	569



15-93. Source Active Count Register (SACNT) .....	569
15-94. Source Active Destination Address Register (SADST) .....	570
15-95. Source Active B-Index Register (SABIDX) .....	570
15-96. Source Active Memory Protection Proxy Register (SAMPPTY) .....	571
15-97. Source Active Count Reload Register (SACNTRLD) .....	572
15-98. Source Active Source Address B-Reference Register (SASRCBREF).....	572
15-99. Source Active Destination Address B-Reference Register (SADSTBREF) .....	573
15-100. Destination FIFO Set Count Reload Register (DFCNTRLD) .....	573
15-101. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF).....	574
15-102. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) .....	574
15-103. Destination FIFO Options Register <i>n</i> (DFOPT <i>n</i> ) .....	575
15-104. Destination FIFO Source Address Register <i>n</i> (DFSRC <i>n</i> ) .....	576
15-105. Destination FIFO Count Register <i>n</i> (DFCNT <i>n</i> ) .....	576
15-106. Destination FIFO Destination Address Register <i>n</i> (DFDST <i>n</i> ) .....	577
15-107. Destination FIFO B-Index Register <i>n</i> (DFBIDX <i>n</i> ) .....	577
15-108. Destination FIFO Memory Protection Proxy Register <i>n</i> (DFMPPXY <i>n</i> ) .....	578
16-1. EMIFA Functional Block Diagram .....	583
16-2. Timing Waveform of SDRAM PRE Command .....	587
16-3. EMIFA to 2M × 16 × 4 bank SDRAM Interface .....	588
16-4. EMIFA to 512K × 16 × 2 bank SDRAM Interface .....	588
16-5. Timing Waveform for Basic SDRAM Read Operation .....	595
16-6. Timing Waveform for Basic SDRAM Write Operation .....	596
16-7. EMIFA Asynchronous Interface .....	598
16-8. EMIFA to 8-bit/16-bit Memory Interface.....	599
16-9. Common Asynchronous Interface .....	599
16-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode.....	604
16-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode.....	606
16-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode .....	608
16-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode .....	610
16-14. EMIFA to NAND Flash Interface .....	612
16-15. ECC Value for 8-Bit NAND Flash.....	614
16-16. EMIFA Reset Block Diagram .....	617
16-17. EMIFA PSC Block Diagram .....	622
16-18. Example Configuration Interface.....	625
16-19. SDRAM Timing Register (SDTIMR) .....	626
16-20. SDRAM Self Refresh Exit Timing Register (SDSRETR) .....	627
16-21. SDRAM Refresh Control Register (SDRCR).....	627
16-22. SDRAM Configuration Register (SDCR).....	628
16-23. Timing Waveform of an ASRAM Read .....	630
16-24. Timing Waveform of an ASRAM Write .....	631
16-25. Timing Waveform of an ASRAM Read with PCB Delays.....	633
16-26. Timing Waveform of an ASRAM Write with PCB Delays.....	634
16-27. Timing Waveform of a NAND Flash Read .....	639
16-28. Timing Waveform of a NAND Flash Command Write .....	641
16-29. Timing Waveform of a NAND Flash Address Write .....	641
16-30. Timing Waveform of a NAND Flash Data Write .....	642
16-31. Module ID Register (MIDR).....	647
16-32. Asynchronous Wait Cycle Configuration Register (AWCCR) .....	647
16-33. SDRAM Configuration Register (SDCR).....	649

16-34. SDRAM Refresh Control Register (SDRCR).....	651
16-35. Asynchronous <i>n</i> Configuration Register (CE <i>n</i> CFG).....	652
16-36. SDRAM Timing Register (SDTIMR).....	654
16-37. SDRAM Self Refresh Exit Timing Register (SDSRETR).....	655
16-38. EMIFA Interrupt Raw Register (INTRAW).....	656
16-39. EMIFA Interrupt Mask Register (INTMSK).....	657
16-40. EMIFA Interrupt Mask Set Register (INTMSKSET).....	658
16-41. EMIFA Interrupt Mask Clear Register (INTMSKCLR).....	659
16-42. NAND Flash Control Register (NANDFCR).....	660
16-43. NAND Flash Status Register (NANDFSR).....	662
16-44. NAND Flash <i>n</i> ECC Register (NANDF <i>n</i> ECC).....	663
16-45. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD).....	664
16-46. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1).....	665
16-47. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2).....	665
16-48. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3).....	666
16-49. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4).....	666
16-50. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1).....	667
16-51. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2).....	667
16-52. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1).....	668
16-53. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2).....	668
17-1. GPIO Block Diagram.....	671
17-2. Revision ID Register (REVID).....	680
17-3. GPIO Interrupt Per-Bank Enable Register (BINTEN).....	681
17-4. GPIO Banks 0 and 1 Direction Register (DIR01).....	682
17-5. GPIO Banks 2 and 3 Direction Register (DIR23).....	682
17-6. GPIO Banks 4 and 5 Direction Register (DIR45).....	682
17-7. GPIO Banks 6 and 7 Direction Register (DIR67).....	682
17-8. GPIO Bank 8 Direction Register (DIR8).....	683
17-9. GPIO Banks 0 and 1 Output Data Register (OUT_DATA01).....	684
17-10. GPIO Banks 2 and 3 Output Data Register (OUT_DATA23).....	684
17-11. GPIO Banks 4 and 5 Output Data Register (OUT_DATA45).....	684
17-12. GPIO Banks 6 and 7 Output Data Register (OUT_DATA67).....	684
17-13. GPIO Bank 8 Output Data Register (OUT_DATA8).....	685
17-14. GPIO Banks 0 and 1 Set Data Register (SET_DATA01).....	686
17-15. GPIO Banks 2 and 3 Set Data Register (SET_DATA23).....	686
17-16. GPIO Banks 4 and 5 Set Data Register (SET_DATA45).....	686
17-17. GPIO Banks 6 and 7 Set Data Register (SET_DATA67).....	686
17-18. GPIO Bank 8 Set Data Register (SET_DATA8).....	687
17-19. GPIO Banks 0 and 1 Clear Data Register (CLR_DATA01).....	688
17-20. GPIO Banks 2 and 3 Clear Data Register (CLR_DATA23).....	688
17-21. GPIO Banks 4 and 5 Clear Data Register (CLR_DATA45).....	688
17-22. GPIO Banks 6 and 7 Clear Data Register (CLR_DATA67).....	688
17-23. GPIO Bank 8 Clear Data Register (CLR_DATA8).....	689
17-24. GPIO Banks 0 and 1 Input Data Register (IN_DATA01).....	690
17-25. GPIO Banks 2 and 3 Input Data Register (IN_DATA23).....	690
17-26. GPIO Banks 4 and 5 Input Data Register (IN_DATA45).....	690
17-27. GPIO Banks 6 and 7 Input Data Register (IN_DATA67).....	690
17-28. GPIO Bank 8 Input Data Register (IN_DATA8).....	691
17-29. GPIO Banks 0 and 1 Set Rise Trigger Register (SET_RIS_TRIG01).....	692

17-30. GPIO Banks 2 and 3 Set Rise Trigger Register (SET_RIS_TRIG23) .....	692
17-31. GPIO Banks 4 and 5 Set Rise Trigger Register (SET_RIS_TRIG45) .....	692
17-32. GPIO Banks 6 and 7 Set Rise Trigger Register (SET_RIS_TRIG67) .....	692
17-33. GPIO Bank 8 Set Rise Trigger Register (SET_RIS_TRIG8).....	693
17-34. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR_RIS_TRIG01).....	694
17-35. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR_RIS_TRIG23).....	694
17-36. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR_RIS_TRIG45).....	694
17-37. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR_RIS_TRIG67).....	694
17-38. GPIO Bank 8 Clear Rise Trigger Register (CLR_RIS_TRIG8) .....	695
17-39. GPIO Banks 0 and 1 Set Rise Trigger Register (SET_FAL_TRIG01).....	696
17-40. GPIO Banks 2 and 3 Set Rise Trigger Register (SET_FAL_TRIG23).....	696
17-41. GPIO Banks 4 and 5 Set Rise Trigger Register (SET_FAL_TRIG45).....	696
17-42. GPIO Banks 6 and 7 Set Rise Trigger Register (SET_FAL_TRIG67).....	696
17-43. GPIO Bank 8 Set Rise Trigger Register (SET_FAL_TRIG8) .....	697
17-44. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR_FAL_TRIG01) .....	698
17-45. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR_FAL_TRIG23) .....	698
17-46. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR_FAL_TRIG45) .....	698
17-47. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR_FAL_TRIG67) .....	698
17-48. GPIO Bank 8 Clear Rise Trigger Register (CLR_FAL_TRIG8).....	699
17-49. GPIO Banks 0 and 1 Interrupt Status Register (INTSTAT01) .....	700
17-50. GPIO Banks 2 and 3 Interrupt Status Register (INTSTAT23) .....	700
17-51. GPIO Banks 4 and 5 Interrupt Status Register (INTSTAT45) .....	700
17-52. GPIO Banks 6 and 7 Interrupt Status Register (INTSTAT67) .....	700
17-53. GPIO Bank 8 Interrupt Status Register (INTSTAT8).....	701
18-1. HPI Block Diagram .....	704
18-2. Example of Host-Processor Signal Connections .....	709
18-3. HPI Strobe and Select Logic .....	711
18-4. Multiplexed-Mode Host Read Cycle .....	713
18-5. Multiplexed-Mode Host Write Cycle .....	714
18-6. Multiplexed-Mode Single-Halfword HPIC Cycle (Read or Write).....	715
18-7. $\overline{UHPI\_HRDY}$ Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode.....	716
18-8. $\overline{UHPI\_HRDY}$ Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle) .....	716
18-9. $\overline{UHPI\_HRDY}$ Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles).....	716
18-10. $\overline{UHPI\_HRDY}$ Behavior During an HPIC Write Cycle in the Multiplexed Mode .....	717
18-11. $\overline{UHPI\_HRDY}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing) .....	717
18-12. $\overline{UHPI\_HRDY}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write) .....	717
18-13. $\overline{UHPI\_HRDY}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write) .....	718
18-14. FIFOs in the HPI .....	719
18-15. Host-to-CPU Interrupt State Diagram .....	724
18-16. CPU-to-Host Interrupt State Diagram .....	725
18-17. Revision Identification Register (REVID) .....	727
18-18. Power and Emulation Management Register (PWREMU_MGMT) .....	727
18-19. GPIO Enable Register (GPIO_EN).....	728
18-20. GPIO Direction 1 Register (GPIO_DIR1).....	729
18-21. GPIO Data 1 Register (GPIO_DAT1) .....	729

18-22. GPIO Direction 2 Register (GPIO_DIR2) .....	730
18-23. GPIO Data 2 Register (GPIO_DAT2) .....	731
18-24. Host Port Interface Control Register (HPIC)—Host Access Permissions .....	732
18-25. Host Port Interface Control Register (HPIC)—CPU Access Permissions .....	732
18-26. Host Port Interface Write Address Register (HPIAW).....	734
18-27. Host Port Interface Read Address Register (HPIAR) .....	734
19-1. I2C Peripheral Block Diagram.....	737
19-2. Multiple I2C Modules Connected .....	738
19-3. Clocking Diagram for the I2C Peripheral .....	739
19-4. Synchronization of Two I2C Clock Generators During Arbitration .....	740
19-5. Bit Transfer on the I2C-Bus .....	741
19-6. I2C Peripheral START and STOP Conditions .....	741
19-7. I2C Peripheral Data Transfer.....	742
19-8. I2C Peripheral 7-Bit Addressing Format (FDF = 0, XA = 0 in ICMR) .....	742
19-9. I2C Peripheral 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in ICMR).....	743
19-10. I2C Peripheral Free Data Format (FDF = 1 in ICMR).....	743
19-11. I2C Peripheral 7-Bit Addressing Format With Repeated START Condition (FDF = 0, XA = 0 in ICMR) ...	743
19-12. Arbitration Procedure Between Two Master-Transmitters.....	746
19-13. I2C Own Address Register (ICOAR) .....	751
19-14. I2C Interrupt Mask Register (ICMR) .....	752
19-15. I2C Interrupt Status Register (ICSTR) .....	753
19-16. I2C Clock Low-Time Divider Register (ICLKL).....	756
19-17. I2C Clock High-Time Divider Register (ICLKH).....	756
19-18. I2C Data Count Register (ICNT) .....	757
19-19. I2C Data Receive Register (ICDRR) .....	758
19-20. I2C Slave Address Register (ICSAR) .....	759
19-21. I2C Data Transmit Register (ICDXR) .....	760
19-22. I2C Mode Register (ICMR) .....	761
19-23. Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit .....	764
19-24. I2C Interrupt Vector Register (ICIVR) .....	765
19-25. I2C Extended Mode Register (ICEMDR) .....	766
19-26. I2C Prescaler Register (ICPSC) .....	767
19-27. I2C Revision Identification Register 1 (REVID1) .....	768
19-28. I2C Revision Identification Register 2 (REVID2) .....	768
19-29. I2C DMA Control Register (ICDMAC).....	769
19-30. I2C Pin Function Register (ICPFUNC).....	770
19-31. I2C Pin Direction Register (ICPDIR) .....	771
19-32. I2C Pin Data In Register (ICPDIN) .....	772
19-33. I2C Pin Data Out Register (ICPDOUT) .....	773
19-34. I2C Pin Data Set Register (ICPDSET) .....	774
19-35. I2C Pin Data Clear Register (ICPDCLR) .....	775
20-1. McASP Block Diagram .....	779
20-2. McASP to Parallel 2-Channel DACs .....	780
20-3. McASP to 6-Channel DAC and 2-Channel DAC .....	780
20-4. McASP to Digital Amplifier.....	781
20-5. McASP as Digital Audio Encoder .....	781
20-6. TDM Format—6 Channel TDM Example .....	782
20-7. TDM Format Bit Delays from Frame Sync .....	783

20-8. Inter-IC Sound (I2S) Format.....	783
20-9. Biphase-Mark Code (BMC).....	784
20-10. S/PDIF Subframe Format.....	785
20-11. S/PDIF Frame Format.....	786
20-12. Definition of Bit, Word, and Slot .....	787
20-13. Bit Order and Word Alignment Within a Slot Examples .....	788
20-14. Definition of Frame and Frame Sync Width .....	789
20-15. Transmit Clock Generator Block Diagram .....	791
20-16. Receive Clock Generator Block Diagram.....	792
20-17. Frame Sync Generator Block Diagram .....	793
20-18. Individual Serializer and Connections Within McASP .....	794
20-19. Receive Format Unit.....	795
20-20. Transmit Format Unit.....	796
20-21. McASP I/O Pin Control Block Diagram .....	798
20-22. McASP I/O Pin to Control Register Mapping.....	799
20-23. Burst Frame Sync Mode .....	804
20-24. Transmit DMA Event (AXEVT) Generation in TDM Time Slots .....	807
20-25. DSP Service Time Upon Transmit DMA Event (AXEVT).....	812
20-26. DSP Service Time Upon Receive DMA Event (AREVT) .....	814
20-27. DMA Events in an Audio Example—Two Events .....	816
20-28. McASP Audio FIFO (AFIFO) Block Diagram.....	817
20-29. Data Flow Through Transmit Format Unit .....	820
20-30. Data Flow Through Receive Format Unit.....	822
20-31. Audio Mute (AMUTE) Block Diagram .....	824
20-32. Transmit Clock Failure Detection Circuit Block Diagram .....	828
20-33. Receive Clock Failure Detection Circuit Block Diagram .....	829
20-34. Serializers in Loopback Mode.....	830
20-35. Revision Identification Register (REV) .....	836
20-36. Pin Function Register (PFUNC) .....	837
20-37. Pin Direction Register (PDIR) .....	839
20-38. Pin Data Output Register (PDOUT) .....	841
20-39. Pin Data Input Register (PDIN) .....	843
20-40. Pin Data Set Register (PDSET).....	845
20-41. Pin Data Clear Register (PDCLR) .....	847
20-42. Global Control Register (GBLCTL) .....	849
20-43. Audio Mute Control Register (AMUTE) .....	851
20-44. Digital Loopback Control Register (DLBCTL) .....	853
20-45. Digital Mode Control Register (DITCTL) .....	854
20-46. Receiver Global Control Register (RGBLCTL).....	855
20-47. Receive Format Unit Bit Mask Register (RMASK) .....	856
20-48. Receive Bit Stream Format Register (RFMT).....	857
20-49. Receive Frame Sync Control Register (AFSRCTL).....	859
20-50. Receive Clock Control Register (ACLKRCTL) .....	860
20-51. Receive High-Frequency Clock Control Register (AHCLKRCTL) .....	861
20-52. Receive TDM Time Slot Register (RTDM) .....	862
20-53. Receiver Interrupt Control Register (RINTCTL) .....	863
20-54. Receiver Status Register (RSTAT) .....	864
20-55. Current Receive TDM Time Slot Registers (RSLOT) .....	865
20-56. Receive Clock Check Control Register (RCLKCHK).....	866



20-57. Receiver DMA Event Control Register (REVTCTL) .....	867
20-58. Transmitter Global Control Register (XGBLCTL).....	868
20-59. Transmit Format Unit Bit Mask Register (XMASK).....	869
20-60. Transmit Bit Stream Format Register (XFMT) .....	870
20-61. Transmit Frame Sync Control Register (AFSXCTL) .....	872
20-62. Transmit Clock Control Register (ACLKXCTL) .....	873
20-63. Transmit High-Frequency Clock Control Register (AHCLKXCTL) .....	874
20-64. Transmit TDM Time Slot Register (XTDM).....	875
20-65. Transmitter Interrupt Control Register (XINTCTL).....	876
20-66. Transmitter Status Register (XSTAT) .....	877
20-67. Current Transmit TDM Time Slot Register (XSLOT).....	878
20-68. Transmit Clock Check Control Register (XCLKCHK) .....	879
20-69. Transmitter DMA Event Control Register (XEVTCTL) .....	880
20-70. Serializer Control Registers (SRCTL <sub>n</sub> ) .....	881
20-71. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5) .....	882
20-72. DIT Right Channel Status Registers (DITCSRB0-DITCSRB5) .....	882
20-73. DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5) .....	883
20-74. DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5) .....	883
20-75. Transmit Buffer Registers (XBUF <sub>n</sub> ) .....	884
20-76. Receive Buffer Registers (RBUF <sub>n</sub> ).....	884
20-77. AFIFO Revision Identification Register (AFIFOREV) .....	885
20-78. Write FIFO Control Register (WFIFOCTL) .....	886
20-79. Write FIFO Status Register (WFIFOSTS) .....	887
20-80. Read FIFO Control Register (RFIFOCTL).....	888
20-81. Read FIFO Status Register (RFIFOSTS).....	889
21-1. McBSP Block Diagram .....	892
21-2. Clock and Frame Generation .....	894
21-3. Transmit Data Clocking .....	895
21-4. Receive Data Clocking .....	895
21-5. Sample Rate Generator Block Diagram .....	896
21-6. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1 .....	899
21-7. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3 .....	899
21-8. Digital Loopback Mode.....	900
21-9. Programmable Frame Period and Width .....	902
21-10. Dual-Phase Frame Example .....	904
21-11. Single-Phase Frame of Four 8-Bit Elements .....	905
21-12. Single-Phase Frame of One 32-Bit Element .....	906
21-13. Data Delay .....	906
21-14. 2-Bit Data Delay Used to Discard Framing Bit.....	907
21-15. McBSP Standard Operation .....	908
21-16. Receive Operation .....	909
21-17. Transmit Operation .....	909
21-18. Maximum Frame Frequency for Transmit and Receive .....	910
21-19. Unexpected Frame Synchronization With (R/X)FIG = 0.....	911
21-20. Unexpected Frame Synchronization With (R/X)FIG = 1 .....	912
21-21. Maximum Frame Frequency Operation With 8-Bit Data.....	912
21-22. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1 .....	913
21-23. Serial Port Receive Overrun.....	914
21-24. Serial Port Receive Overrun Avoided .....	914

21-25. Decision Tree Response to Receive Frame Synchronization Pulse .....	915
21-26. Unexpected Receive Frame Synchronization Pulse .....	916
21-27. Transmit With Data Overwrite .....	916
21-28. Transmit Empty .....	917
21-29. Transmit Empty Avoided.....	917
21-30. Decision Tree Response to Transmit Frame Synchronization Pulse .....	919
21-31. Unexpected Transmit Frame Synchronization Pulse .....	919
21-32. McBSP Buffer FIFO (BFIFO) Block Diagram .....	920
21-33. Companding Flow .....	922
21-34. Companding Data Formats.....	922
21-35. Transmit Data Companding Format in DXR .....	922
21-36. Companding of Internal Data.....	923
21-37. DX Timing for Multichannel Operation .....	925
21-38. Alternating Between the Channels of Partition A and the Channels of Partition B .....	927
21-39. Reassigning Channel Blocks Throughout a McBSP Data Transfer.....	927
21-40. McBSP Data Transfer in the 8-Partition Mode.....	928
21-41. Activity on McBSP Pins for the Possible Values of XMCM .....	931
21-42. Data Receive Register (DRR).....	941
21-43. Data Transmit Register (DXR) .....	941
21-44. Serial Port Control Register (SPCR) .....	942
21-45. Receive Control Register (RCR).....	944
21-46. Transmit Control Register (XCR) .....	946
21-47. Sample Rate Generator Register (SRGR) .....	948
21-48. Multichannel Control Registers (MCR).....	949
21-49. Enhanced Receive Channel Enable Register $n$ (RCEREN).....	953
21-50. Enhanced Transmit Channel Enable Register $n$ (XCEREN) .....	955
21-51. Pin Control Register (PCR).....	957
21-52. BFIFO Revision Identification Register (BFIFOREV) .....	959
21-53. Write FIFO Control Register (WFIFOCTL) .....	960
21-54. Write FIFO Status Register (WFIFOSTS) .....	961
21-55. Read FIFO Control Register (RFIFOCTL).....	962
21-56. Read FIFO Status Register (RFIFOSTS).....	963
22-1. Real-Time Clock Block Diagram .....	965
22-2. 32-kHz Oscillator Counter Compensation .....	969
22-3. Kick State Machine .....	970
22-4. Second Register (SECOND) .....	973
22-5. Minute Register (MINUTE) .....	973
22-6. Hour Register (HOUR) .....	974
22-7. Days Register (DAY) .....	975
22-8. Month Register (MONTH) .....	975
22-9. Year Register (YEAR).....	976
22-10. Day of the Week Register (DOTW) .....	976
22-11. Alarm Second Register (ALARMSECOND) .....	977
22-12. Alarm Minute Register (ALARMMINUTE) .....	977
22-13. Alarm Hour Register (ALARMHOUR) .....	978
22-14. Alarm Day Register (ALARMDAY) .....	979
22-15. Alarm Month Register (ALARMMONTH) .....	980
22-16. Alarm Year Register (ALARMYEAR).....	980
22-17. Control Register (CTRL) .....	981

22-18. Status Register (STATUS).....	982
22-19. Interrupt Register (INTERRUPT) .....	983
22-20. Compensation (LSB) Register (COMPLSB).....	984
22-21. Compensation (MSB) Register (COMPMSB) .....	985
22-22. Oscillator Register (OSC).....	986
22-23. Scratch Registers (SCRATCH $n$ ).....	987
22-24. Kick Registers (KICK $n$ R).....	987
23-1. SPI Block Diagram.....	990
23-2. SPI 3-Pin Option .....	996
23-3. SPI 4-Pin Option with $\overline{\text{SPIx\_SCS}}[n]$ .....	998
23-4. SPI 4-Pin Option with $\overline{\text{SPIx\_ENA}}$ .....	1000
23-5. SPI 5-Pin Option with $\overline{\text{SPIx\_ENA}}$ and $\overline{\text{SPIx\_SCS}}[n]$ .....	1002
23-6. Format for Transmitting 12-Bit Word.....	1003
23-7. Format for 10-Bit Received Word .....	1003
23-8. Clock Mode with POLARITY = 0 and PHASE = 0 .....	1004
23-9. Clock Mode with POLARITY = 0 and PHASE = 1 .....	1005
23-10. Clock Mode with POLARITY = 1 and PHASE = 0 .....	1005
23-11. Clock Mode with POLARITY = 1 and PHASE = 1 .....	1005
23-12. Five Bits per Character (5-Pin Option) .....	1006
23-13. SPI 3-Pin Master Mode with WDELAY .....	1011
23-14. SPI 4-Pin with $\overline{\text{SPIx\_SCS}}[n]$ Mode with T2CDELAY, WDELAY, and C2TDELAY .....	1012
23-15. SPI 4-Pin with $\overline{\text{SPIx\_ENA}}$ Mode Demonstrating T2EDELAY and WDELAY .....	1013
23-16. SPI 5-Pin Mode Demonstrating T2CDELAY, T2EDELAY, and WDELAY .....	1015
23-17. SPI 5-Pin Mode Demonstrating C2TDELAY and C2EDELAY .....	1016
23-18. SPI Global Control Register 0 (SPIGCR0).....	1017
23-19. SPI Global Control Register 1 (SPIGCR1).....	1018
23-20. SPI Interrupt Register (SPIINT0) .....	1020
23-21. SPI Interrupt Level Register (SPILVL).....	1022
23-22. SPI Flag Register (SPIFLG) .....	1023
23-23. SPI Pin Control Register 0 (SPIPC0) .....	1025
23-24. SPI Pin Control Register 1 (SPIPC1) .....	1026
23-25. SPI Pin Control Register 2 (SPIPC2) .....	1027
23-26. SPI Pin Control Register 3 (SPIPC3) .....	1028
23-27. SPI Pin Control Register 4 (SPIPC4) .....	1029
23-28. SPI Pin Control Register 5 (SPIPC5) .....	1030
23-29. SPI Data Register 0 (SPIDAT0) .....	1031
23-30. SPI Data Register 1 (SPIDAT1) .....	1032
23-31. SPI Buffer Register (SPIBUF) .....	1033
23-32. SPI Emulation Register (SPIEMU) .....	1035
23-33. SPI Delay Register (SPIDELAY) .....	1036
23-34. Example: $t_{\text{C2TDELAY}} = 8$ SPI Module Clock Cycles .....	1037
23-35. Example: $t_{\text{T2CDELAY}} = 4$ SPI Module Clock Cycles .....	1038
23-36. Transmit-Data-Finished-to- $\overline{\text{SPIx\_ENA}}$ -Inactive-Timeout .....	1038
23-37. Chip-Select-Active-to- $\overline{\text{SPIx\_ENA}}$ -Signal-Active-Timeout.....	1038
23-38. SPI Default Chip Select Register (SPIDEF) .....	1039
23-39. SPI Data Format Register (SPIFMT $n$ ).....	1040
23-40. SPI Interrupt Vector Register 1 (INTVEC1).....	1042
24-1. Timer Block Diagram .....	1045
24-2. Timer Clock Source Block Diagram.....	1046



24-3.	64-Bit Timer Mode Block Diagram .....	1047
24-4.	Dual 32-Bit Timers Chained Mode Block Diagram .....	1050
24-5.	Dual 32-Bit Timers Chained Mode Example.....	1050
24-6.	Dual 32-Bit Timers Unchained Mode Block Diagram.....	1052
24-7.	Dual 32-Bit Timers Unchained Mode Example.....	1053
24-8.	32-Bit Timer Counter Overflow Example .....	1056
24-9.	Watchdog Timer Mode Block Diagram .....	1058
24-10.	Watchdog Timer Operation State Diagram .....	1058
24-11.	Timer Operation in Pulse Mode (CPn = 0).....	1060
24-12.	Timer Operation in Clock Mode (CPn = 1).....	1060
24-13.	Revision ID Register (REVID) .....	1064
24-14.	Emulation Management Register (EMUMGT).....	1064
24-15.	GPIO Interrupt Control and Enable Register (GPINTGPEN).....	1065
24-16.	GPIO Data and Direction Register (GPDATGPDIR) .....	1066
24-17.	Timer Counter Register 12 (TIM12).....	1067
24-18.	Timer Counter Register 34 (TIM34).....	1067
24-19.	Timer Period Register 12 (PRD12) .....	1068
24-20.	Timer Period Register 34 (PRD34) .....	1068
24-21.	Timer Control Register (TCR) .....	1069
24-22.	Timer Global Control Register (TGCR).....	1071
24-23.	Watchdog Timer Control Register (WDTCR) .....	1072
24-24.	Timer Reload Register 12 (REL12) .....	1073
24-25.	Timer Reload Register 34 (REL34) .....	1073
24-26.	Timer Capture Register 12 (CAP12).....	1074
24-27.	Timer Capture Register 34 (CAP34).....	1074
24-28.	Timer Interrupt Control and Status Register (INTCTLSTAT) .....	1075
24-29.	Timer Compare Register (CMPn) .....	1076
25-1.	UART Block Diagram .....	1079
25-2.	UART Clock Generation Diagram.....	1080
25-3.	Relationships Between Data Bit, BCLK, and UART Input Clock.....	1081
25-4.	UART Protocol Formats .....	1083
25-5.	UART Interface Using Autoflow Diagram .....	1086
25-6.	Autoflow Functional Timing Waveforms for UARTn_RTS .....	1087
25-7.	Autoflow Functional Timing Waveforms for UARTn_CTS .....	1087
25-8.	UART Interrupt Request Enable Paths.....	1089
25-9.	Receiver Buffer Register (RBR) .....	1092
25-10.	Transmitter Holding Register (THR) .....	1093
25-11.	Interrupt Enable Register (IER).....	1094
25-12.	Interrupt Identification Register (IIR).....	1095
25-13.	FIFO Control Register (FCR) .....	1097
25-14.	Line Control Register (LCR) .....	1098
25-15.	Modem Control Register (MCR).....	1100
25-16.	Line Status Register (LSR).....	1101
25-17.	Modem Status Register (MSR) .....	1104
25-18.	Scratch Pad Register (SCR) .....	1105
25-19.	Divisor LSB Latch (DLL).....	1106
25-20.	Divisor MSB Latch (DLH) .....	1106
25-21.	Revision Identification Register 1 (REVID1) .....	1107
25-22.	Revision Identification Register 2 (REVID2) .....	1107

---

25-23. Power and Emulation Management Register (PWREMU_MGMT) .....	1108
25-24. Mode Definition Register (MDR) .....	1109

## List of Tables

2-1.	DSP Interrupt Map .....	55
3-1.	TMS320C6742 DSP System Interconnect Matrix.....	61
5-1.	MPU Memory Regions.....	67
5-2.	MPU2 Default Configuration .....	67
5-3.	Device Master Settings .....	67
5-4.	Request Type Access Controls.....	69
5-5.	MPU_BOOTCFG_ERR Interrupt Sources .....	71
5-6.	Memory Protection Unit 2 (MPU2) Registers .....	71
5-7.	Revision ID Register (REVID) Field Descriptions .....	72
5-8.	Configuration Register (CONFIG) Field Descriptions .....	73
5-9.	Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions.....	74
5-10.	Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions.....	75
5-11.	Interrupt Enable Set Register (IENSET) Field Descriptions .....	76
5-12.	Interrupt Enable Clear Register (IENCLR) Field Descriptions.....	76
5-13.	Fixed Range Memory Protection Page Attributes Register (FXD_MPPA) Field Descriptions .....	78
5-14.	MPU2 Programmable Range <i>n</i> Start Address Register (PROG <sub><i>n</i></sub> _MPSAR) Field Descriptions .....	79
5-15.	MPU2 Programmable Range <i>n</i> End Address Register (PROG <sub><i>n</i></sub> _MPEAR) Field Descriptions .....	80
5-16.	Programmable Range Memory Protection Page Attributes Register (PROG <sub><i>n</i></sub> _MPPA) Field Descriptions ....	81
5-17.	Fault Address Register (FLTADDR) Field Descriptions .....	82
5-18.	Fault Status Register (FLTSTAT) Field Descriptions .....	83
5-19.	Fault Clear Register (FLTCLR) Field Descriptions.....	84
6-1.	Device Clock Inputs.....	86
6-2.	System Clock Domains.....	86
6-3.	Example PLL Frequencies .....	89
6-4.	DDR2/mDDR Memory Controller MCLK Frequencies.....	90
6-5.	EMIFA Frequencies .....	91
6-6.	Peripherals .....	93
7-1.	System PLLC Output Clocks .....	97
7-2.	PLL Controller 0 (PLL0) Registers.....	100
7-3.	PLL Controller 1 (PLL1) Registers.....	101
7-4.	PLL0 Revision Identification Register (REVID) Field Descriptions .....	101
7-5.	PLL1 Revision Identification Register (REVID) Field Descriptions .....	102
7-6.	Reset Type Status Register (RSTYPE) Field Descriptions .....	102
7-7.	Reset Control Register (RSCTRL) Field Descriptions .....	103
7-8.	PLL0 Control Register (PLLCTL) Field Descriptions.....	104
7-9.	PLL1 Control Register (PLLCTL) Field Descriptions.....	105
7-10.	PLL0 OBSCLK Select Register (OCSEL) Field Descriptions .....	106
7-11.	PLL1 OBSCLK Select Register (OCSEL) Field Descriptions .....	107
7-12.	PLL Multiplier Control Register (PLLM) Field Descriptions.....	108
7-13.	PLL0 Pre-Divider Control Register (PREDIV) Field Descriptions .....	108
7-14.	PLL0 Divider 1 Register (PLLDIV1) Field Descriptions .....	109
7-15.	PLL1 Divider 1 Register (PLLDIV1) Field Descriptions .....	109
7-16.	PLL0 Divider 2 Register (PLLDIV2) Field Descriptions .....	110
7-17.	PLL1 Divider 2 Register (PLLDIV2) Field Descriptions .....	110
7-18.	PLL0 Divider 3 Register (PLLDIV3) Field Descriptions .....	111
7-19.	PLL1 Divider 3 Register (PLLDIV3) Field Descriptions .....	111
7-20.	PLL0 Divider 4 Register (PLLDIV4) Field Descriptions .....	112

7-21.	PLLC0 Divider 5 Register (PLLDIV5) Field Descriptions .....	112
7-22.	PLLC0 Divider 6 Register (PLLDIV6) Field Descriptions .....	113
7-23.	PLLC0 Divider 7 Register (PLLDIV7) Field Descriptions .....	113
7-24.	PLLC0 Oscillator Divider 1 Register (OSCDIV) Field Descriptions .....	114
7-25.	PLLC1 Oscillator Divider 1 Register (OSCDIV) Field Descriptions .....	114
7-26.	PLL Post-Divider Control Register (POSTDIV) Field Descriptions .....	115
7-27.	PLL Controller Command Register (PLLCMD) Field Descriptions .....	115
7-28.	PLL Controller Status Register (PLLSTAT) Field Descriptions .....	116
7-29.	PLLC0 Clock Align Control Register (ALNCTL) Field Descriptions .....	117
7-30.	PLLC1 Clock Align Control Register (ALNCTL) Field Descriptions .....	118
7-31.	PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions .....	119
7-32.	PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions .....	120
7-33.	PLLC0 Clock Enable Control Register (CKEN) Field Descriptions .....	121
7-34.	PLLC1 Clock Enable Control Register (CKEN) Field Descriptions .....	121
7-35.	PLLC0 Clock Status Register (CKSTAT) Field Descriptions.....	122
7-36.	PLLC1 Clock Status Register (CKSTAT) Field Descriptions.....	123
7-37.	PLLC0 SYSCLK Status Register (SYSTAT) Field Descriptions.....	124
7-38.	PLLC1 SYSCLK Status Register (SYSTAT) Field Descriptions.....	125
7-39.	Emulation Performance Counter 0 Register (EMUCNT0) Field Descriptions .....	126
7-40.	Emulation Performance Counter 1 Register (EMUCNT1) Field Descriptions .....	126
8-1.	PSC0 Default Module Configuration .....	129
8-2.	PSC1 Default Module Configuration .....	129
8-3.	Module States .....	131
8-4.	IcePick Emulation Commands .....	133
8-5.	PSC Interrupt Events .....	133
8-6.	Power and Sleep Controller 0 (PSC0) Registers .....	136
8-7.	Power and Sleep Controller 1 (PSC1) Registers .....	136
8-8.	Revision Identification Register (REVID) Field Descriptions .....	137
8-9.	Interrupt Evaluation Register (INTEVAL) Field Descriptions .....	137
8-10.	PSC0 Module Error Pending Register 0 (MERRPR0) Field Descriptions.....	138
8-11.	PSC0 Module Error Clear Register 0 (MERRCR0) Field Descriptions .....	139
8-12.	Power Error Pending Register (PERRPR) Field Descriptions .....	140
8-13.	Power Error Clear Register (PERRCR) Field Descriptions .....	140
8-14.	Power Domain Transition Command Register (PTCMD) Field Descriptions .....	141
8-15.	Power Domain Transition Status Register (PTSTAT) Field Descriptions .....	142
8-16.	Power Domain 0 Status Register (PDSTAT0) Field Descriptions.....	143
8-17.	Power Domain 1 Status Register (PDSTAT1) Field Descriptions.....	144
8-18.	Power Domain 0 Control Register (PDCTL0) Field Descriptions.....	145
8-19.	Power Domain 1 Control Register (PDCTL1) Field Descriptions.....	146
8-20.	Power Domain 0 Configuration Register (PDCFG0) Field Descriptions.....	147
8-21.	Power Domain 1 Configuration Register (PDCFG1) Field Descriptions.....	148
8-22.	Module Status <i>n</i> Register (MDSTAT <i>n</i> ) Field Descriptions .....	149
8-23.	PSC0 Module Control <i>n</i> Register (MDCTL <i>n</i> ) Field Descriptions .....	150
8-24.	PSC1 Module Control <i>n</i> Register (MDCTL <i>n</i> ) Field Descriptions .....	151
9-1.	Power Management Features.....	154
10-1.	Master IDs .....	166
10-2.	Default Master Priority .....	166
10-3.	System Configuration Module 0 (SYSCFG0) Registers .....	167
10-4.	System Configuration Module 1 (SYSCFG1) Registers .....	168

10-5.	Revision Identification Register (REVID) Field Descriptions .....	168
10-6.	Device Identification Register 0 (DEVIDR0) Field Descriptions .....	169
10-7.	Boot Configuration Register (BOOTCFG) Field Descriptions .....	169
10-8.	Chip Revision Identification Register (CHIPREVIDR) Field Descriptions .....	170
10-9.	Kick 0 Register (KICK0R) Field Descriptions .....	171
10-10.	Kick 1 Register (KICK1R) Field Descriptions .....	171
10-11.	Host 1 Configuration Register (HOST1CFG) Field Descriptions .....	172
10-12.	Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions .....	173
10-13.	Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions .....	174
10-14.	Interrupt Enable Register (IENSET) Field Descriptions.....	175
10-15.	Interrupt Enable Clear Register (IENCLR) Field Descriptions .....	175
10-16.	End of Interrupt Register (EOI) Field Descriptions .....	176
10-17.	Fault Address Register (FLTADDRR) Field Descriptions .....	176
10-18.	Fault Status Register (FLTSTAT) Field Descriptions .....	177
10-19.	Master Priority 0 Register (MSTPRI0) Field Descriptions .....	178
10-20.	Master Priority 1 Register (MSTPRI1) Field Descriptions .....	179
10-21.	Master Priority 2 Register (MSTPRI2) Field Descriptions .....	180
10-22.	Pin Multiplexing Control 0 Register (PINMUX0) Field Descriptions.....	181
10-23.	Pin Multiplexing Control 1 Register (PINMUX1) Field Descriptions.....	183
10-24.	Pin Multiplexing Control 2 Register (PINMUX2) Field Descriptions.....	185
10-25.	Pin Multiplexing Control 3 Register (PINMUX3) Field Descriptions.....	187
10-26.	Pin Multiplexing Control 4 Register (PINMUX4) Field Descriptions.....	189
10-27.	Pin Multiplexing Control 5 Register (PINMUX5) Field Descriptions.....	191
10-28.	Pin Multiplexing Control 6 Register (PINMUX6) Field Descriptions.....	193
10-29.	Pin Multiplexing Control 7 Register (PINMUX7) Field Descriptions.....	195
10-30.	Pin Multiplexing Control 8 Register (PINMUX8) Field Descriptions.....	197
10-31.	Pin Multiplexing Control 9 Register (PINMUX9) Field Descriptions.....	199
10-32.	Pin Multiplexing Control 10 Register (PINMUX10) Field Descriptions.....	201
10-33.	Pin Multiplexing Control 11 Register (PINMUX11) Field Descriptions.....	203
10-34.	Pin Multiplexing Control 12 Register (PINMUX12) Field Descriptions.....	205
10-35.	Pin Multiplexing Control 13 Register (PINMUX13) Field Descriptions.....	207
10-36.	Pin Multiplexing Control 14 Register (PINMUX14) Field Descriptions.....	209
10-37.	Pin Multiplexing Control 15 Register (PINMUX15) Field Descriptions.....	211
10-38.	Pin Multiplexing Control 16 Register (PINMUX16) Field Descriptions.....	213
10-39.	Pin Multiplexing Control 17 Register (PINMUX17) Field Descriptions.....	215
10-40.	Pin Multiplexing Control 18 Register (PINMUX18) Field Descriptions.....	217
10-41.	Pin Multiplexing Control 19 Register (PINMUX19) Field Descriptions.....	219
10-42.	Suspend Source Register (SUSPSRC) Field Descriptions .....	221
10-43.	Chip Signal Register (CHIPSIG) Field Descriptions.....	223
10-44.	Chip Signal Clear Register (CHIPSIG_CLR) Field Descriptions .....	224
10-45.	Chip Configuration 0 Register (CFGCHIP0) Field Descriptions .....	225
10-46.	Chip Configuration 1 Register (CFGCHIP1) Field Descriptions .....	226
10-47.	Chip Configuration 3 Register (CFGCHIP3) Field Descriptions .....	228
10-48.	Chip Configuration 4 Register (CFGCHIP4) Field Descriptions .....	229
10-49.	VTP I/O Control Register (VTPIO_CTL) Field Descriptions.....	230
10-50.	DDR Slew Register (DDR_SLEW) Field Descriptions .....	231
10-51.	Deep Sleep Register (DEEPSLEEP) Field Descriptions .....	232
10-52.	Pullup/Pulldown Enable Register (PUPD_ENA) Field Descriptions.....	233
10-53.	Pullup/Pulldown Select Register (PUPD_SEL) Field Descriptions .....	233

10-54. Pullup/Pulldown Select Register (PUPD_SEL) Default Values .....	234
10-55. RXACTIVE Control Register (RXACTIVE) Field Descriptions .....	235
12-1. DDR2/mDDR SDRAM Commands .....	243
12-2. Truth Table for DDR2/mDDR SDRAM Commands .....	244
12-3. Addressable Memory Ranges.....	251
12-4. Configuration Register Fields for Address Mapping.....	252
12-5. Logical Address-to-DDR2/mDDR SDRAM Address Map for 16-bit SDRAM .....	253
12-6. Address Mapping Diagram for 16-Bit SDRAM (IBANKPOS = 1).....	255
12-7. DDR2/mDDR Memory Controller FIFO Description .....	257
12-8. Refresh Urgency Levels .....	260
12-9. Configuration Bit Field for Partial Array Self-refresh .....	261
12-10. Reset Sources.....	262
12-11. DDR2 SDRAM Configuration by MRS Command.....	264
12-12. DDR2 SDRAM Configuration by EMRS(1) Command.....	264
12-13. Mobile DDR SDRAM Configuration by MRS Command.....	264
12-14. Mobile DDR SDRAM Configuration by EMRS(1) Command .....	265
12-15. SDCR Configuration.....	270
12-16. DDR2 Memory Refresh Specification .....	271
12-17. SDRCR Configuration .....	271
12-18. SDTMR1 Configuration.....	272
12-19. SDTMR2 Configuration.....	272
12-20. DRPYC1R Configuration.....	273
12-21. DDR2/mDDR Memory Controller Registers.....	274
12-22. Revision ID Register (REVID) Field Descriptions .....	274
12-23. SDRAM Status Register (SDRSTAT) Field Descriptions.....	275
12-24. SDRAM Configuration Register (SDCR) Field Descriptions .....	276
12-25. SDRAM Refresh Control Register (SDRCR) Field Descriptions .....	279
12-26. SDRAM Timing Register 1 (SDTMR1) Field Descriptions.....	280
12-27. SDRAM Timing Register 2 (SDTMR2) Field Descriptions.....	281
12-28. SDRAM Configuration Register 2 (SDCR2) Field Descriptions .....	282
12-29. Peripheral Bus Burst Priority Register (PBBPR) Field Descriptions .....	283
12-30. Performance Counter 1 Register (PC1) Field Descriptions .....	284
12-31. Performance Counter 2 Register (PC2) Field Descriptions .....	284
12-32. Performance Counter Configuration Register (PCC) Field Descriptions .....	285
12-33. Performance Counter Filter Configuration.....	286
12-34. Performance Counter Master Region Select Register (PCMRS) Field Descriptions .....	287
12-35. Performance Counter Time Register (PCT) Field Description .....	288
12-36. DDR PHY Reset Control Register (DRPYRCR) .....	288
12-37. Interrupt Raw Register (IRR) Field Descriptions.....	289
12-38. Interrupt Masked Register (IMR) Field Descriptions .....	289
12-39. Interrupt Mask Set Register (IMSR) Field Descriptions.....	290
12-40. Interrupt Mask Clear Register (IMCR) Field Descriptions .....	291
12-41. DDR PHY Control Register 1 (DRPYC1R) Field Descriptions.....	292
13-1. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger.....	306
13-2. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger .....	308
13-3. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger.....	310
13-4. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers .....	312
13-5. ECAP Initialization for APWM Mode .....	314
13-6. ECAP1 Initialization for Multichannel PWM Generation with Synchronization .....	316



13-7.	ECAP2 Initialization for Multichannel PWM Generation with Synchronization .....	316
13-8.	ECAP3 Initialization for Multichannel PWM Generation with Synchronization .....	316
13-9.	ECAP4 Initialization for Multichannel PWM Generation with Synchronization .....	316
13-10.	ECAP1 Initialization for Multichannel PWM Generation with Phase Control .....	319
13-11.	ECAP2 Initialization for Multichannel PWM Generation with Phase Control .....	319
13-12.	ECAP3 Initialization for Multichannel PWM Generation with Phase Control .....	319
13-13.	Control and Status Register Set .....	320
13-14.	Time-Stamp Counter Register (TSCTR) Field Descriptions .....	320
13-15.	Counter Phase Control Register (CTRPHS) Field Descriptions .....	321
13-16.	Capture 1 Register (CAP1) Field Descriptions.....	321
13-17.	Capture 2 Register (CAP2) Field Descriptions.....	322
13-18.	Capture 3 Register (CAP3) Field Descriptions.....	322
13-19.	Capture 4 Register (CAP4) Field Descriptions.....	323
13-20.	ECAP Control Register 1 (ECCTL1) Field Descriptions .....	323
13-21.	ECAP Control Register 2 (ECCTL2) Field Descriptions .....	325
13-22.	ECAP Interrupt Enable Register (ECEINT) Field Descriptions .....	327
13-23.	ECAP Interrupt Flag Register (ECFLG) Field Descriptions .....	328
13-24.	ECAP Interrupt Clear Register (ECCLR) Field Descriptions .....	329
13-25.	ECAP Interrupt Forcing Register (ECFRC) Field Descriptions .....	330
13-26.	Revision ID Register (REVID) Field Descriptions .....	331
14-1.	ePWM Module Control and Status Registers Grouped by Submodule.....	337
14-2.	Submodule Configuration Parameters .....	338
14-3.	Time-Base Submodule Registers.....	343
14-4.	Key Time-Base Signals .....	344
14-5.	Counter-Compare Submodule Registers .....	352
14-6.	Counter-Compare Submodule Key Signals .....	352
14-7.	Action-Qualifier Submodule Registers .....	356
14-8.	Action-Qualifier Submodule Possible Input Events .....	357
14-9.	Action-Qualifier Event Priority for Up-Down-Count Mode .....	359
14-10.	Action-Qualifier Event Priority for Up-Count Mode .....	359
14-11.	Action-Qualifier Event Priority for Down-Count Mode.....	359
14-12.	Behavior if CMPA/CMPB is Greater than the Period .....	360
14-13.	EPWMx Initialization for .....	363
14-14.	EPWMx Run Time Changes for .....	363
14-15.	EPWMx Initialization for .....	365
14-16.	EPWMx Run Time Changes for .....	365
14-17.	EPWMx Initialization for .....	367
14-18.	EPWMx Run Time Changes for .....	367
14-19.	EPWMx Initialization for .....	369
14-20.	EPWMx Run Time Changes for .....	369
14-21.	EPWMx Initialization for .....	371
14-22.	EPWMx Run Time Changes for .....	371
14-23.	EPWMx Initialization for .....	373
14-24.	EPWMx Run Time Changes for .....	373
14-25.	Dead-Band Generator Submodule Registers .....	374
14-26.	Classical Dead-Band Operating Modes .....	376
14-27.	PWM-Chopper Submodule Registers.....	378
14-28.	Trip-Zone Submodule Registers.....	383
14-29.	Possible Actions On a Trip Event.....	384

14-30. Event-Trigger Submodule Registers .....	386
14-31. Resolution for PWM and HRPWM.....	391
14-32. HRPWM Submodule Registers .....	392
14-33. Relationship Between MEP Steps, PWM Frequency and Resolution .....	393
14-34. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right) .....	394
14-35. EPWM1 Initialization for .....	401
14-36. EPWM2 Initialization for .....	401
14-37. EPWM3 Initialization for .....	401
14-38. EPWM1 Initialization for .....	404
14-39. EPWM2 Initialization for .....	404
14-40. EPWM1 Initialization for .....	407
14-41. EPWM2 Initialization for .....	407
14-42. EPWM1 Initialization for .....	410
14-43. EPWM2 Initialization for .....	410
14-44. EPWM3 Initialization for .....	411
14-45. EPWM1 Initialization for .....	416
14-46. EPWM2 Initialization for .....	416
14-47. EPWM3 Initialization for .....	417
14-48. EPWM1 Initialization for .....	420
14-49. EPWM2 Initialization for .....	420
14-50. Submodule Registers .....	421
14-51. Time-Base Submodule Registers.....	421
14-52. Time-Base Control Register (TBCTL) Field Descriptions .....	422
14-53. Time-Base Status Register (TBSTS) Field Descriptions .....	423
14-54. Time-Base Phase Register (TBPHS) Field Descriptions .....	424
14-55. Time-Base Counter Register (TBCNT) Field Descriptions .....	424
14-56. Time-Base Period Register (TBPRD) Field Descriptions.....	425
14-57. Counter-Compare Submodule Registers .....	425
14-58. Counter-Compare Control Register (CMPCTL) Field Descriptions .....	426
14-59. Counter-Compare A Register (CMPA) Field Descriptions.....	427
14-60. Counter-Compare B Register (CMPB) Field Descriptions.....	428
14-61. Action-Qualifier Submodule Registers .....	428
14-62. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions .....	429
14-63. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions .....	430
14-64. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions.....	431
14-65. Action-Qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions .....	432
14-66. Dead-Band Generator Submodule Registers .....	432
14-67. Dead-Band Generator Control Register (DBCTL) Field Descriptions.....	433
14-68. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions .....	434
14-69. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions .....	434
14-70. PWM-Chopper Control Register (PCCTL) Bit Descriptions .....	435
14-71. Trip-Zone Submodule Registers .....	436
14-72. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions .....	436
14-73. Trip-Zone Control Register (TZCTL) Field Descriptions .....	437
14-74. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions .....	437
14-75. Trip-Zone Flag Register (TZFLG) Field Descriptions .....	438
14-76. Trip-Zone Clear Register (TZCLR) Field Descriptions .....	439
14-77. Trip-Zone Force Register (TZFRC) Field Descriptions .....	439
14-78. Event-Trigger Submodule Registers .....	440



14-79. Event-Trigger Selection Register (ETSEL) Field Descriptions .....	440
14-80. Event-Trigger Prescale Register (ETPS) Field Descriptions .....	441
14-81. Event-Trigger Flag Register (ETFLG) Field Descriptions .....	442
14-82. Event-Trigger Clear Register (ETCLR) Field Descriptions .....	442
14-83. Event-Trigger Force Register (ETFRC) Field Descriptions .....	443
14-84. High-Resolution PWM Submodule Registers .....	443
14-85. Time-Base Phase High-Resolution Register (TBPHSHR) Field Descriptions .....	444
14-86. Counter-Compare A High-Resolution Register (CMPAHR) Field Descriptions.....	444
14-87. HRPWM Configuration Register (HRCNFG) Field Descriptions.....	445
15-1. EDMA3 Channel Parameter Description .....	459
15-2. Dummy and Null Transfer Request .....	462
15-3. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set) .....	463
15-4. Expected Number of Transfers for Non-Null Transfer .....	471
15-5. EDMA3 DMA Channel to PaRAM Mapping .....	473
15-6. Shadow Region Registers .....	475
15-7. Chain Event Triggers .....	478
15-8. EDMA3 Transfer Completion Interrupts .....	478
15-9. EDMA3 Error Interrupts .....	478
15-10. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping .....	479
15-11. Number of Interrupts .....	480
15-12. EDMA3 Transfer Controller Configurations .....	487
15-13. Read/Write Command Optimization Rules .....	493
15-14. EDMA3 Channel Controller (EDMA3CC) Parameter RAM (PaRAM) Entries.....	512
15-15. Channel Options Parameters (OPT) Field Descriptions .....	513
15-16. Channel Source Address Parameter (SRC) Field Descriptions .....	515
15-17. A Count/B Count Parameter (A_B_CNT) Field Descriptions .....	515
15-18. Channel Destination Address Parameter (DST) Field Descriptions .....	516
15-19. Source B Index/Destination B Index Parameter (SRC_DST_BIDX) Field Descriptions.....	516
15-20. Link Address/B Count Reload Parameter (LINK_BCNTRLD) Field Descriptions .....	517
15-21. Source C Index/Destination C Index Parameter (SRC_DST_CIDX) Field Descriptions .....	518
15-22. C Count Parameter (CCNT) Field Descriptions.....	518
15-23. EDMA3 Channel Controller (EDMA3CC) Registers.....	519
15-24. Revision ID Register (REVID) Field Descriptions .....	522
15-25. EDMA3CC Configuration Register (CCCFG) Field Descriptions .....	523
15-26. QDMA Channel <i>n</i> Mapping Register (QCHMAP <i>n</i> ) Field Descriptions.....	524
15-27. DMA Channel Queue Number Register <i>n</i> (DMAQNUM <i>n</i> ) Field Descriptions .....	525
15-28. Bits in DMAQNUM <i>n</i> .....	525
15-29. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions .....	526
15-30. Event Missed Register (EMR) Field Descriptions .....	527
15-31. Event Missed Clear Register (EMCR) Field Descriptions .....	528
15-32. QDMA Event Missed Register (QEMR) Field Descriptions .....	529
15-33. QDMA Event Missed Clear Register (QEMCR) Field Descriptions .....	530
15-34. EDMA3CC Error Register (CCERR) Field Descriptions .....	531
15-35. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions .....	532
15-36. Error Evaluate Register (EEVAL) Field Descriptions.....	533
15-37. DMA Region Access Enable Register for Region <i>m</i> (DRAE <i>m</i> ) Field Descriptions .....	534
15-38. QDMA Region Access Enable for Region <i>m</i> (QRAE <i>m</i> ) Field Descriptions .....	535
15-39. Event Queue Entry Registers (QxEy) Field Descriptions.....	536
15-40. Queue <i>n</i> Status Register (QSTAT <i>n</i> ) Field Descriptions .....	537

15-41. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions .....	538
15-42. EDMA3CC Status Register (CCSTAT) Field Descriptions .....	539
15-43. Event Register (ER) Field Descriptions .....	541
15-44. Event Clear Register (ECR) Field Descriptions .....	542
15-45. Event Set Register (ESR) Field Descriptions .....	543
15-46. Chained Event Register (CER) Field Descriptions .....	544
15-47. Event Enable Register (EER) Field Descriptions .....	545
15-48. Event Enable Clear Register (EECR) Field Descriptions .....	546
15-49. Event Enable Set Register (EESR) Field Descriptions .....	546
15-50. Secondary Event Register (SER) Field Descriptions .....	547
15-51. Secondary Event Clear Register (SECR) Field Descriptions .....	547
15-52. Interrupt Enable Register (IER) Field Descriptions .....	548
15-53. Interrupt Enable Clear Register (IECR) Field Descriptions .....	549
15-54. Interrupt Enable Set Register (IESR) Field Descriptions .....	549
15-55. Interrupt Pending Register (IPR) Field Descriptions .....	550
15-56. Interrupt Clear Register (ICR) Field Descriptions .....	551
15-57. Interrupt Evaluate Register (IEVAL) Field Descriptions .....	552
15-58. QDMA Event Register (QER) Field Descriptions .....	553
15-59. QDMA Event Enable Register (QEER) Field Descriptions .....	554
15-60. QDMA Event Enable Clear Register (QEECR) Field Descriptions .....	555
15-61. QDMA Event Enable Set Register (QEESR) Field Descriptions .....	555
15-62. QDMA Secondary Event Register (QSER) Field Descriptions .....	556
15-63. QDMA Secondary Event Clear Register (QSECR) Field Descriptions .....	557
15-64. EDMA3 Transfer Controller (EDMA3TC) Registers .....	558
15-65. Revision ID Register (REVID) Field Descriptions .....	559
15-66. EDMA3TC Configuration Register (TCCFG) Field Descriptions .....	560
15-67. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions .....	561
15-68. Error Status Register (ERRSTAT) Field Descriptions .....	562
15-69. Error Enable Register (ERREN) Field Descriptions .....	563
15-70. Error Clear Register (ERRCLR) Field Descriptions .....	564
15-71. Error Details Register (ERRDET) Field Descriptions .....	565
15-72. Error Interrupt Command Register (ERRCMD) Field Descriptions .....	566
15-73. Read Command Rate Register (RDRATE) Field Descriptions .....	567
15-74. Source Active Options Register (SAOPT) Field Descriptions .....	568
15-75. Source Active Source Address Register (SASRC) Field Descriptions .....	569
15-76. Source Active Count Register (SACNT) Field Descriptions .....	569
15-77. Source Active Destination Address Register (SADST) Field Descriptions .....	570
15-78. Source Active B-Index Register (SABIDX) Field Descriptions .....	570
15-79. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions .....	571
15-80. Source Active Count Reload Register (SACNTRLD) Field Descriptions .....	572
15-81. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions .....	572
15-82. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions .....	573
15-83. Destination FIFO Set Count Reload Register (DFCNTRLD) Field Descriptions .....	573
15-84. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF) Field Descriptions .....	574
15-85. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) Field Descriptions .....	574
15-86. Destination FIFO Options Register <i>n</i> (DFOPT <i>n</i> ) Field Descriptions .....	575
15-87. Destination FIFO Source Address Register <i>n</i> (DFSRC <i>n</i> ) Field Descriptions .....	576
15-88. Destination FIFO Count Register <i>n</i> (DFCNT <i>n</i> ) Field Descriptions .....	576
15-89. Destination FIFO Destination Address Register <i>n</i> (DFDST <i>n</i> ) Field Descriptions .....	577

15-90. Destination FIFO B-Index Register $n$ (DFBIDX $n$ ) Field Descriptions .....	577
15-91. Destination FIFO Memory Protection Proxy Register $n$ (DFMPPRXY $n$ ) Field Descriptions .....	578
15-92. Debug List .....	579
16-1. EMIFA Pins Used to Access Both SDRAM and Asynchronous Memories.....	584
16-2. EMIFA Pins Specific to SDRAM .....	585
16-3. EMIFA Pins Specific to Asynchronous Memory .....	585
16-4. EMIFA SDRAM Commands .....	586
16-5. Truth Table for SDRAM Commands .....	586
16-6. 16-bit EMIFA Address Pin Connections .....	588
16-7. Description of the SDRAM Configuration Register (SDCR).....	589
16-8. Description of the SDRAM Refresh Control Register (SDRCR).....	589
16-9. Description of the SDRAM Timing Register (SDTIMR) .....	590
16-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR) .....	590
16-11. SDRAM LOAD MODE REGISTER Command.....	591
16-12. Refresh Urgency Levels .....	592
16-13. Mapping from Logical Address to EMIFA Pins for 16-bit SDRAM .....	597
16-14. Normal Mode vs. Select Strobe Mode .....	598
16-15. Description of the Asynchronous $m$ Configuration Register (CE $n$ CFG) .....	600
16-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC) .....	601
16-17. Description of the EMIFA Interrupt Mask Set Register (INTMSKSET) .....	603
16-18. Description of the EMIFA Interrupt Mast Clear Register (INTMSKCLR) .....	603
16-19. Asynchronous Read Operation in Normal Mode .....	603
16-20. Asynchronous Write Operation in Normal Mode .....	605
16-21. Asynchronous Read Operation in Select Strobe Mode.....	607
16-22. Asynchronous Write Operation in Select Strobe Mode .....	609
16-23. Description of the NAND Flash Control Register (NANDFCR) .....	611
16-24. Reset Sources.....	617
16-25. Interrupt Monitor and Control Bit Fields.....	619
16-26. SR Field Value For the EMIFA to K4S641632H-TC(L)70 Interface.....	624
16-27. SDTIMR Field Calculations for the EMIFA to K4S641632H-TC(L)70 Interface .....	626
16-28. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface.....	627
16-29. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface.....	627
16-30. SDCR Field Values For the EMIFA to K4S641632H-TC(L)70 Interface .....	628
16-31. EMIFA Input Timing Requirements.....	629
16-32. ASRAM Output Timing Characteristics .....	629
16-33. ASRAM Input Timing Requirement for a Read .....	629
16-34. ASRAM Input Timing Requirements for a Write .....	630
16-35. ASRAM Timing Requirements With PCB Delays.....	632
16-36. EMIFA Timing Requirements for TC5516100FT-12 Example .....	635
16-37. ASRAM Timing Requirements for TC5516100FT-12 Example .....	635
16-38. Measured PCB Delays for TC5516100FT-12 Example .....	635
16-39. Configuring CE3CFG for TC5516100FT-12 Example .....	637
16-40. Recommended Margins.....	637
16-41. EMIFA Read Timing Requirements .....	638
16-42. NAND Flash Read Timing Requirements .....	638
16-43. NAND Flash Write Timing Requirements .....	640
16-44. EMIFA Timing Requirements for HY27UA081G1M Example.....	643
16-45. NAND Flash Timing Requirements for HY27UA081G1M Example.....	643
16-46. Configuring CE2CFG for HY27UA081G1M Example.....	645

16-47. Configuring NANDFCR for HY27UA081G1M Example.....	645
16-48. External Memory Interface (EMIFA) Registers .....	646
16-49. Module ID Register (MIDR) Field Descriptions .....	647
16-50. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions.....	648
16-51. SDRAM Configuration Register (SDCR) Field Descriptions .....	649
16-52. SDRAM Refresh Control Register (SDRCR) Field Descriptions .....	651
16-53. Asynchronous <i>n</i> Configuration Register (CE <sub><i>n</i></sub> CFG) Field Descriptions .....	652
16-54. SDRAM Timing Register (SDTIMR) Field Descriptions.....	654
16-55. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions .....	655
16-56. EMIFA Interrupt Raw Register (INTRAW) Field Descriptions.....	656
16-57. EMIFA Interrupt Mask Register (INTMSK) Field Descriptions .....	657
16-58. EMIFA Interrupt Mask Set Register (INTMSKSET) Field Descriptions .....	658
16-59. EMIFA Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions .....	659
16-60. NAND Flash Control Register (NANDFCR) Field Descriptions.....	660
16-61. NAND Flash Status Register (NANDFSR) Field Descriptions .....	662
16-62. NAND Flash <i>n</i> ECC Register (NANDF <sub><i>n</i></sub> ECC) Field Descriptions .....	663
16-63. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) Field Descriptions .....	664
16-64. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) Field Descriptions.....	665
16-65. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) Field Descriptions.....	665
16-66. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) Field Descriptions.....	666
16-67. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) Field Descriptions.....	666
16-68. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) Field Descriptions .....	667
16-69. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) Field Descriptions .....	667
16-70. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1) Field Descriptions.....	668
16-71. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2) Field Descriptions.....	668
17-1. GPIO Register Bits and Banks Associated With GPIO Signals .....	672
17-2. GPIO Registers .....	679
17-3. Revision ID Register (REVID) Field Descriptions .....	680
17-4. GPIO Interrupt Per-Bank Enable Register (BINTEN) Field Descriptions.....	681
17-5. GPIO Direction Register (DIR <sub><i>n</i></sub> ) Field Descriptions .....	683
17-6. GPIO Output Data Register (OUT_DATA <sub><i>n</i></sub> ) Field Descriptions .....	685
17-7. GPIO Set Data Register (SET_DATA <sub><i>n</i></sub> ) Field Descriptions .....	687
17-8. GPIO Clear Data Register (CLR_DATA <sub><i>n</i></sub> ) Field Descriptions .....	689
17-9. GPIO Input Data Register (IN_DATA <sub><i>n</i></sub> ) Field Descriptions.....	691
17-10. GPIO Set Rising Edge Trigger Interrupt Register (SET_RIS_TRIG <sub><i>n</i></sub> ) Field Descriptions .....	693
17-11. GPIO Clear Rising Edge Interrupt Register (CLR_RIS_TRIG <sub><i>n</i></sub> ) Field Descriptions .....	695
17-12. GPIO Set Falling Edge Trigger Interrupt Register (SET_FAL_TRIG <sub><i>n</i></sub> ) Field Descriptions .....	697
17-13. GPIO Clear Falling Edge Interrupt Register (CLR_FAL_TRIG <sub><i>n</i></sub> ) Field Descriptions .....	699
17-14. GPIO Interrupt Status Register (INTSTAT <sub><i>n</i></sub> ) Field Descriptions.....	701
18-1. HPI Pins.....	706
18-2. Value on Optional Pins when Configured as General-Purpose I/O .....	707
18-3. Options for Connecting Host and HPI Data Strobe Pins .....	711
18-4. Access Types Selectable With the UHPI_HCNTL Signals.....	712
18-5. Cycle Types Selectable With the UHPI_HCNTL and UHPI_HR/ $\overline{W}$ Signals.....	712
18-6. HPI Registers.....	726
18-7. Revision Identification Register (REVID) Field Descriptions .....	727
18-8. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions .....	727
18-9. GPIO Enable Register (GPIO_EN) Field Descriptions.....	728
18-10. GPIO Direction 1 Register (GPIO_DIR1) Field Descriptions.....	729

18-11. GPIO Data 1 Register (GPIO_DAT1) Field Descriptions .....	729
18-12. GPIO Direction 2 Register (GPIO_DIR2) Field Descriptions.....	730
18-13. GPIO Data 2 Register (GPIO_DAT2) Field Descriptions.....	731
18-14. Host Port Interface Control Register (HPIC) Field Descriptions .....	733
18-15. Host Port Interface Write Address Register (HPIAW) Field Descriptions .....	734
18-16. Host Port Interface Read Address Register (HPIAR) Field Descriptions.....	734
19-1. Operating Modes of the I2C Peripheral.....	744
19-2. Ways to Generate a NACK Bit.....	745
19-3. Descriptions of the I2C Interrupt Events .....	749
19-4. Inter-Integrated Circuit (I2C) Registers .....	750
19-5. I2C Own Address Register (ICOAR) Field Descriptions.....	751
19-6. I2C Interrupt Mask Register (ICIMR) Field Descriptions.....	752
19-7. I2C Interrupt Status Register (ICSTR) Field Descriptions .....	753
19-8. I2C Clock Low-Time Divider Register (ICCLKL) Field Descriptions .....	756
19-9. I2C Clock High-Time Divider Register (ICCLKH) Field Descriptions .....	756
19-10. I2C Data Count Register (ICCNT) Field Descriptions.....	757
19-11. I2C Data Receive Register (ICDRR) Field Descriptions.....	758
19-12. I2C Slave Address Register (ICSAR) Field Descriptions.....	759
19-13. I2C Data Transmit Register (ICDXR) Field Descriptions .....	760
19-14. I2C Mode Register (ICMDR) Field Descriptions .....	761
19-15. Master-Transmitter/Receiver Bus Activity Defined by RM, STT, and STP Bits .....	763
19-16. How the MST and FDF Bits Affect the Role of TRX Bit .....	763
19-17. I2C Interrupt Vector Register (ICIVR) Field Descriptions.....	765
19-18. I2C Extended Mode Register (ICEMDR) Field Descriptions .....	766
19-19. I2C Prescaler Register (ICPSC) Field Descriptions .....	767
19-20. I2C Revision Identification Register 1 (REVID1) Field Descriptions .....	768
19-21. I2C Revision Identification Register 2 (REVID2) Field Descriptions .....	768
19-22. I2C DMA Control Register (ICDMAC) Field Descriptions .....	769
19-23. I2C Pin Function Register (ICPFUNC) Field Descriptions .....	770
19-24. I2C Pin Direction Register (ICPDIR) Field Descriptions .....	771
19-25. I2C Pin Data In Register (ICPDIN) Field Descriptions .....	772
19-26. I2C Pin Data Out Register (ICPDOUT) Field Descriptions.....	773
19-27. I2C Pin Data Set Register (ICPDSET) Field Descriptions.....	774
19-28. I2C Pin Data Clear Register (ICPDCLR) Field Descriptions .....	775
20-1. Biphase-Mark Encoder.....	784
20-2. Preamble Codes .....	785
20-3. Channel Status and User Data for Each DIT Block .....	811
20-4. Transmit Bitstream Data Alignment .....	819
20-5. Receive Bitstream Data Alignment .....	821
20-6. EDMA Events - McASP .....	831
20-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port .....	832
20-8. McASP Registers Accessed by CPU/EDMA Through DMA Port .....	835
20-9. McASP AFIFO Registers Accessed Through Peripheral Configuration Port .....	835
20-10. Revision Identification Register (REV) Field Descriptions.....	836
20-11. Pin Function Register (PFUNC) Field Descriptions .....	838
20-12. Pin Direction Register (PDIR) Field Descriptions.....	840
20-13. Pin Data Output Register (PDOUT) Field Descriptions.....	842
20-14. Pin Data Input Register (PDIN) Field Descriptions.....	844
20-15. Pin Data Set Register (PDSET) Field Descriptions .....	846



20-16. Pin Data Clear Register (PDCLR) Field Descriptions .....	848
20-17. Global Control Register (GBLCTL) Field Descriptions.....	849
20-18. Audio Mute Control Register (AMUTE) Field Descriptions .....	851
20-19. Digital Loopback Control Register (DLBCTL) Field Descriptions.....	853
20-20. Digital Mode Control Register (DITCTL) Field Descriptions.....	854
20-21. Receiver Global Control Register (RGBLCTL) Field Descriptions .....	855
20-22. Receive Format Unit Bit Mask Register (RMASK) Field Descriptions .....	856
20-23. Receive Bit Stream Format Register (RFMT) Field Descriptions .....	857
20-24. Receive Frame Sync Control Register (AFSRCTL) Field Descriptions.....	859
20-25. Receive Clock Control Register (ACLKRCTL) Field Descriptions.....	860
20-26. Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions.....	861
20-27. Receive TDM Time Slot Register (RTDM) Field Descriptions .....	862
20-28. Receiver Interrupt Control Register (RINTCTL) Field Descriptions .....	863
20-29. Receiver Status Register (RSTAT) Field Descriptions.....	864
20-30. Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions.....	865
20-31. Receive Clock Check Control Register (RCLKCHK) Field Descriptions .....	866
20-32. Receiver DMA Event Control Register (REVTCTL) Field Descriptions.....	867
20-33. Transmitter Global Control Register (XGBLCTL) Field Descriptions .....	868
20-34. Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions .....	869
20-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions.....	870
20-36. Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions .....	872
20-37. Transmit Clock Control Register (ACLKXCTL) Field Descriptions .....	873
20-38. Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions .....	874
20-39. Transmit TDM Time Slot Register (XTDM) Field Descriptions .....	875
20-40. Transmitter Interrupt Control Register (XINTCTL) Field Descriptions .....	876
20-41. Transmitter Status Register (XSTAT) Field Descriptions.....	877
20-42. Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions .....	878
20-43. Transmit Clock Check Control Register (XCLKCHK) Field Descriptions.....	879
20-44. Transmitter DMA Event Control Register (XEVTCTL) Field Descriptions.....	880
20-45. Serializer Control Registers (SRCTL <sub>n</sub> ) Field Descriptions .....	881
20-46. AFIFO Revision Identification Register (AFIFOREV) Field Descriptions .....	885
20-47. Write FIFO Control Register (WFIFOCTL) Field Descriptions .....	886
20-48. Write FIFO Status Register (WFIFOSTS) Field Descriptions .....	887
20-49. Read FIFO Control Register (RFIFOCTL) Field Descriptions .....	888
20-50. Read FIFO Status Register (RFIFOSTS) Field Descriptions .....	889
21-1. McBSP Interface Signals .....	893
21-2. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits .....	897
21-3. Receive Clock Selection .....	900
21-4. Transmit Clock Selection .....	901
21-5. Receive Frame Synchronization Selection .....	902
21-6. Transmit Frame Synchronization Selection .....	903
21-7. RCR/XCR Fields Controlling Elements per Frame and Bits per Element.....	904
21-8. Receive/Transmit Frame Length Configuration.....	904
21-9. Receive/Transmit Element Length Configuration.....	905
21-10. Effect of RJUST Bit Values With 12-Bit Example Data ABCh .....	907
21-11. Effect of RJUST Bit Values With 20-Bit Example Data ABCDEh .....	907
21-12. Justification of Expanded Data in DRR .....	923
21-13. Receive Channel Assignment and Control When Two Receive Partitions are Used.....	926
21-14. Transmit Channel Assignment and Control When Two Transmit Partitions are Used .....	926

21-15. Receive Channel Assignment and Control When Eight Receive Partitions are Used .....	928
21-16. Transmit Channel Assignment and Control When Eight Transmit Partitions are Used .....	928
21-17. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits .....	929
21-18. Reset State of McBSP Pins .....	932
21-19. Receiver Clock and Frame Configurations .....	933
21-20. Transmitter Clock and Frame Configurations .....	933
21-21. McBSP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR .....	939
21-22. McBSP Registers .....	940
21-23. Data Receive Register (DRR) Field Descriptions .....	941
21-24. Data Transmit Register (DXR) Field Descriptions .....	941
21-25. Serial Port Control Register (SPCR) Field Descriptions .....	942
21-26. Receive Control Register (RCR) Field Descriptions .....	944
21-27. Transmit Control Register (XCR) Field Descriptions .....	946
21-28. Sample Rate Generator Register (SRGR) Field Descriptions .....	948
21-29. Multichannel Control Register (MCR) Field Descriptions .....	949
21-30. Enhanced Receive Channel Enable Register <i>n</i> (RCEREN) Field Descriptions .....	953
21-31. Use of the Receive Channel Enable Registers .....	954
21-32. Enhanced Transmit Channel Enable Register <i>n</i> (XCEREN) Field Descriptions .....	955
21-33. Use of the Transmit Channel Enable Registers .....	956
21-34. Pin Control Register (PCR) Field Descriptions .....	957
21-35. BFIFO Revision Identification Register (BFIFOREV) Field Descriptions .....	959
21-36. Write FIFO Control Register (WFIFOCTL) Field Descriptions .....	960
21-37. Write FIFO Status Register (WFIFOSTS) Field Descriptions .....	961
21-38. Read FIFO Control Register (RFIFOCTL) Field Descriptions .....	962
21-39. Read FIFO Status Register (RFIFOSTS) Field Descriptions .....	963
22-1. Real-Time Clock Signals .....	966
22-2. Real-Time Clock (RTC) Registers .....	972
22-3. Second Register (SECOND) Field Descriptions .....	973
22-4. Minute Register (MINUTE) Field Descriptions .....	973
22-5. Hour Register (HOUR) Field Descriptions .....	974
22-6. Day Register (DAY) Field Descriptions .....	975
22-7. Month Register (MONTH) Field Descriptions .....	975
22-8. Year Register (YEAR) Field Descriptions .....	976
22-9. Day of the Week Register (DOTW) Field Descriptions .....	976
22-10. Alarm Second Register (ALARMSECOND) Field Descriptions .....	977
22-11. Alarm Minute Register (ALARMMINUTE) Field Descriptions .....	977
22-12. Alarm Hour Register (ALARMHOUR) Field Descriptions .....	978
22-13. Alarm Day Register (ALARMDAY) Field Descriptions .....	979
22-14. Alarm Month Register (ALARMMONTH) Field Descriptions .....	980
22-15. Alarm Years Register (ALARMYEARS) Field Descriptions .....	980
22-16. Control Register (CTRL) Field Descriptions .....	981
22-17. Status Register (STATUS) Field Descriptions .....	982
22-18. Interrupt Register (INTERRUPT) Field Descriptions .....	983
22-19. Compensations Register (COMPLSB) Field Descriptions .....	984
22-20. Compensations Register (COMPMSB) Field Descriptions .....	985
22-21. Oscillator Register (OSC) Field Descriptions .....	986
22-22. Scratch Registers (SCRATCH <i>n</i> ) Field Descriptions .....	987
22-23. Kick Registers (KICK <i>n</i> R) Field Descriptions .....	987
23-1. SPI Pins .....	991



23-2.	SPI Registers .....	992
23-3.	SPI Register Settings Defining Master Modes .....	993
23-4.	Allowed SPI Register Settings in Master Modes .....	993
23-5.	SPI Register Settings Defining Slave Modes .....	995
23-6.	Allowed SPI Register Settings in Slave Modes .....	995
23-7.	Clocking Modes.....	1004
23-8.	SPI Registers .....	1017
23-9.	SPI Global Control Register 0 (SPIGCR0) Field Descriptions .....	1017
23-10.	SPI Global Control Register 1 (SPIGCR1) Field Descriptions .....	1018
23-11.	SPI Interrupt Register (SPIINT0) Field Descriptions.....	1020
23-12.	SPI Interrupt Level Register (SPILVL) Field Descriptions .....	1022
23-13.	SPI Flag Register (SPIFLG) Field Descriptions .....	1023
23-14.	SPI Pin Control Register 0 (SPIPC0) Field Descriptions.....	1025
23-15.	SPI Pin Control Register 1 (SPIPC1) Field Descriptions.....	1026
23-16.	SPI Pin Control Register 2 (SPIPC2) Field Descriptions.....	1027
23-17.	SPI Pin Control Register 3 (SPIPC3) Field Descriptions.....	1028
23-18.	SPI Pin Control Register 4 (SPIPC4) Field Descriptions.....	1029
23-19.	SPI Pin Control Register 5 (SPIPC5) Field Descriptions.....	1030
23-20.	SPI Data Register 0 (SPIDAT0) Field Descriptions.....	1031
23-21.	SPI Data Register 1 (SPIDAT1) Field Descriptions.....	1032
23-22.	SPI Buffer Register (SPIBUF) Field Descriptions .....	1033
23-23.	SPI Emulation Register (SPIEMU) Field Descriptions.....	1035
23-24.	SPI Delay Register (SPIDELAY) Field Descriptions .....	1036
23-25.	SPI Default Chip Select Register (SPIDEF) Field Descriptions .....	1039
23-26.	SPI Data Format Register (SPIFMT <sub>n</sub> ) Field Descriptions .....	1040
23-27.	SPI Interrupt Vector Register 1 (INTVEC1) Field Descriptions .....	1042
24-1.	Timer Clock Source Selection .....	1046
24-2.	64-Bit Timer Configurations .....	1048
24-3.	32-Bit Timer Chained Mode Configurations .....	1051
24-4.	32-Bit Timer Unchained Mode Configurations.....	1054
24-5.	Counter and Period Registers Used in GP Timer Modes.....	1056
24-6.	TSTAT Parameters in Pulse and Clock Modes .....	1060
24-7.	Timer Emulation Modes Selection .....	1062
24-8.	Timer Registers.....	1062
24-9.	Revision ID Register (REVID) Field Descriptions .....	1064
24-10.	Emulation Management Register (EMUMGT) Field Descriptions .....	1064
24-11.	GPIO Interrupt Control and Enable Register (GPINTGPEN) Field Descriptions .....	1065
24-12.	GPIO Data and Direction Register (GPDATGPDIR) Field Descriptions .....	1066
24-13.	Timer Counter Register 12 (TIM12) Field Descriptions .....	1067
24-14.	Timer Counter Register 34 (TIM34) Field Descriptions .....	1067
24-15.	Timer Period Register (PRD12) Field Descriptions.....	1068
24-16.	Timer Period Register (PRD34) Field Descriptions .....	1068
24-17.	Timer Control Register (TCR) Field Descriptions .....	1069
24-18.	Timer Global Control Register (TGCR) Field Descriptions .....	1071
24-19.	Watchdog Timer Control Register (WDTCR) Field Descriptions.....	1072
24-20.	Timer Reload Register 12 (REL12) Field Descriptions.....	1073
24-21.	Timer Reload Register 34 (REL34) Field Descriptions.....	1073
24-22.	Timer Capture Register 12 (CAP12) Field Descriptions .....	1074
24-23.	Timer Capture Register 34 (CAP34) Field Descriptions .....	1074

24-24. Timer Interrupt Control and Status Register (INTCTLSTAT) Field Descriptions.....	1075
24-25. Timer Compare Register (CMP <sub>n</sub> ) Field Descriptions.....	1076
25-1. Baud Rate Examples for 150-MHZ UART Input Clock and 16x Over-sampling Mode .....	1081
25-2. Baud Rate Examples for 150-MHZ UART Input Clock and 13x Over-sampling Mode .....	1081
25-3. UART Signal Descriptions .....	1082
25-4. Character Time for Word Lengths .....	1085
25-5. UART Interrupt Requests Descriptions.....	1089
25-6. UART Registers .....	1091
25-7. Receiver Buffer Register (RBR) Field Descriptions.....	1092
25-8. Transmitter Holding Register (THR) Field Descriptions .....	1093
25-9. Interrupt Enable Register (IER) Field Descriptions .....	1094
25-10. Interrupt Identification Register (IIR) Field Descriptions.....	1095
25-11. Interrupt Identification and Interrupt Clearing Information .....	1096
25-12. FIFO Control Register (FCR) Field Descriptions.....	1097
25-13. Line Control Register (LCR) Field Descriptions .....	1098
25-14. Relationship Between ST, EPS, and PEN Bits in LCR.....	1099
25-15. Number of STOP Bits Generated .....	1099
25-16. Modem Control Register (MCR) Field Descriptions .....	1100
25-17. Line Status Register (LSR) Field Descriptions .....	1101
25-18. Modem Status Register (MSR) Field Descriptions.....	1104
25-19. Scratch Pad Register (MSR) Field Descriptions .....	1105
25-20. Divisor LSB Latch (DLL) Field Descriptions .....	1106
25-21. Divisor MSB Latch (DLH) Field Descriptions .....	1106
25-22. Revision Identification Register 1 (REVID1) Field Descriptions.....	1107
25-23. Revision Identification Register 2 (REVID2) Field Descriptions.....	1107
25-24. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions.....	1108
25-25. Mode Definition Register (MDR) Field Descriptions .....	1109

## Read This First

---

---

---

### About This Manual

This Technical Reference Manual (TRM) describes the System-on-Chip (SoC) and each peripheral in the device. The SoC consists of the following primary components

- DSP subsystem and associated memories
- A set of I/O peripherals

### Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

### Related Documentation From Texas Instruments

Copies of these documents are available on the Internet at [www.ti.com](http://www.ti.com). *Tip:* Enter the literature number in the search box provided at [www.ti.com](http://www.ti.com).

The current documentation that describes related peripherals and other technical collateral, is available in the C6000 DSP product folder at: [www.ti.com/c6000](http://www.ti.com/c6000).

**SPRUFK5— TMS320C674x DSP Megamodule Reference Guide.** Describes the TMS320C674x digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

**SPRUF8— TMS320C674x DSP CPU and Instruction Set Reference Guide.** Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C674x digital signal processors (DSPs). The C674x DSP is an enhancement of the C64x+ and C67x+ DSPs with added functionality and an expanded instruction set.

**SPRUG82— TMS320C674x DSP Cache User's Guide.** Explains the fundamentals of memory caches and describes how the two-level cache-based internal memory architecture in the TMS320C674x digital signal processor (DSP) can be efficiently used in DSP applications. Shows how to maintain coherence with external memory, how to use DMA to reduce memory latencies, and how to optimize your code to improve cache efficiency. The internal memory architecture in the C674x DSP is organized in a two-level hierarchy consisting of a dedicated program cache (L1P) and a dedicated data cache (L1D) on the first level. Accesses by the CPU to these first level caches can complete without CPU pipeline stalls. If the data requested by the CPU is not contained in cache, it is fetched from the next lower memory level, L2 or external memory.

## Overview

---

---

---

Topic	Page
1.1 Introduction .....	52
1.2 DSP Subsystem .....	52

## 1.1 Introduction

The C6742 DSP efficiently handles communication and audio processing tasks. The C6742 DSP consists of the following primary components:

- A set of I/O peripherals
- A powerful DMA subsystem and SDRAM EMIF interface

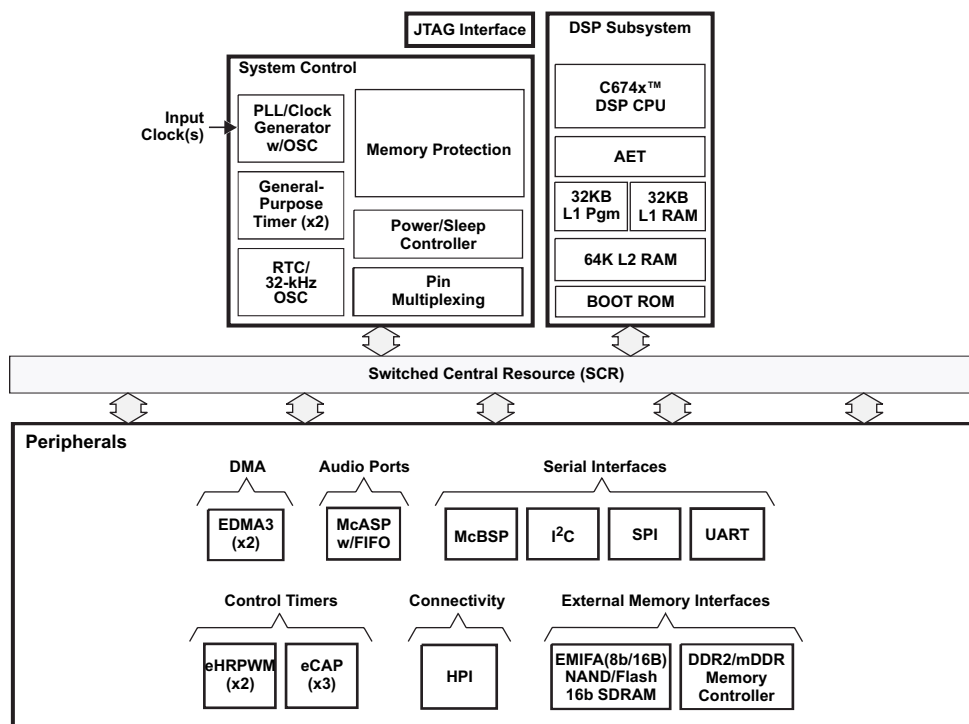
## Block Diagram

A block diagram for the 6742 DSP is shown in [Figure 1-1](#).

## 1.2 DSP Subsystem

The DSP subsystem (DSPSS) includes TI's standard TMS320C674x megamodule and several blocks of internal memory (L1P, L1D, and L2). The *DSP Subsystem* chapter describes the DSPSS components.

**Figure 1-1. TMS320C6742 DSP Block Diagram**



Note: Not all peripherals are available at the same time due to multiplexing.

## DMA Subsystem

The DMA subsystem includes two instances of the enhanced DMA controller (EDMA3). For more information, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.

## DSP Subsystem

---

---

---

Topic	Page
2.1 Introduction .....	54
2.2 TMS320C674x Megamodule .....	55
2.3 Memory Map .....	59
2.4 Advanced Event Triggering (AET) .....	59



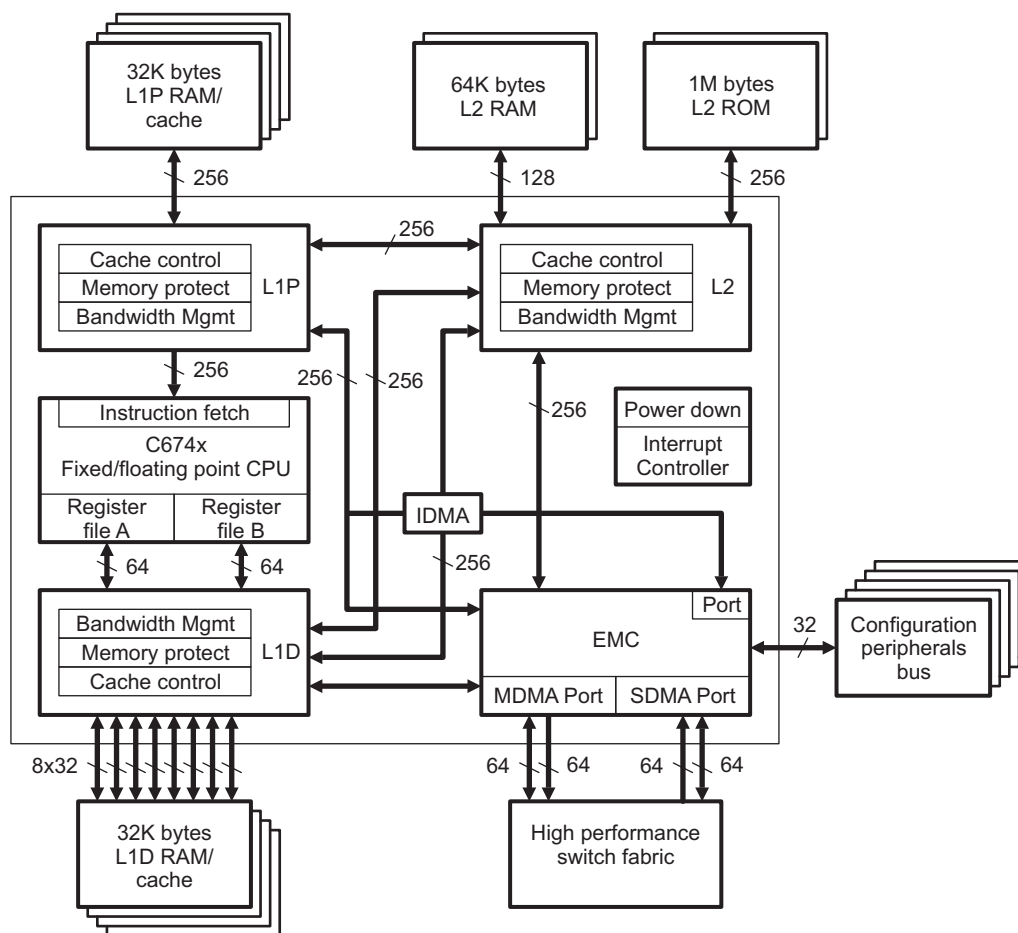
## 2.1 Introduction

The DSP subsystem (Figure 2-1) includes TI's standard TMS320C674x megamodule and several blocks of internal memory (L1P, L1D, and L2). This chapter provides an overview of the DSP subsystem and the following considerations associated with it:

- Memory mapping
- Interrupts
- Power management

For more information on the TMS320C674x megamodule, see the *TMS320C674x DSP Megamodule Reference Guide (SPRUFK5)*, the *TMS320C674x DSP CPU and Instruction Set Reference Guide (SPRUFE8)*, and the *TMS320C674x DSP Cache User's Guide (SPRUG82)*.

**Figure 2-1. TMS320C674x Megamodule Block Diagram**



## 2.2 TMS320C674x Megamodule

The C674x megamodule ([Figure 2-1](#)) consists of the following components:

- TMS320C674x CPU
- Internal memory controllers:
  - Level 1 program memory controller (PMC)
  - Level 1 data memory controller (DMC)
  - Level 2 unified memory controller (UMC)
  - Extended memory controller (EMC)
  - Internal direct memory access (IDMA) controller
- Internal peripherals:
  - Interrupt controller (INTC)
  - Power-down controller (PDC)
  - Bandwidth manager (BWM)
- Advanced event triggering (AET)

For more information about each of these controllers, see the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

### 2.2.1 Internal Memory Controllers

The C674x megamodule implements a two-level internal cache-based memory architecture with external memory support. Level 1 memory (L1) is split into separate program memory (L1P memory) and data memory (L1D memory). L1 memory is accessible to the CPU without stalls. Level 2 memory (L2) can also be split into L2 RAM (normal addressable on-chip memory) and L2 cache for caching external memory locations. The internal direct memory access controller (IDMA) manages DMA among the L1P, L1D, and L2 memories.

### 2.2.2 Internal Peripherals

The C674x megamodule includes the following internal peripherals:

- DSP interrupt controller (INTC)
- DSP power-down controller (PDC)
- Bandwidth manager (BWM)
- Internal DMA (IDMA) controller

This section briefly describes the INTC, PDC, BWM, and IDMA controller. For more information on these internal peripherals, see the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

#### 2.2.2.1 Interrupt Controller (INTC)

The C674x megamodule includes an interrupt controller (INTC) to manage CPU interrupts. The INTC maps DSP device events to 12 CPU interrupts. All DSP device events are listed in [Table 2-1](#). The INTC is fully described in the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

**Table 2-1. DSP Interrupt Map**

Event	Interrupt Name	Source
0	EVT0	C674x Interrupt Control 0
1	EVT1	C674x Interrupt Control 1
2	EVT2	C674x Interrupt Control 2
3	EVT3	C674x Interrupt Control 3
4	T64P0_TINT12	Timer64P0 Interrupt (TINT12)
5	SYSCFG_CHIPINT2	SYSCFG CHIPSIG Register

**Table 2-1. DSP Interrupt Map (continued)**

Event	Interrupt Name	Source
6	—	Reserved
7	EHRPWM0	HiResTimer/PWM0 Interrupt
8	EDMA3_0_CC0_INT1	EDMA3_0 Channel Controller 0 Shadow Region 1 Transfer Completion Interrupt
9	EMU-DTDMA	C674x-ECM
10	EHRPWM0TZ	HiResTimer/PWM0 Trip Zone Interrupt
11	EMU-RTDXRX	C674x-RTDX
12	EMU-RTDXTX	C674x-RTDX
13	IDMAINT0	C674x-EMC
14	IDMAINT1	C674x-EMC
15-17	—	Reserved
18	EHRPWM1	HiResTimer/PWM1 Interrupt
19-22	—	Reserved
23	EHRPWM1TZ	HiResTimer/PWM1 Trip Zone Interrupt
24-33	—	Reserved
34	UHPI_DSPINT	HPI DSP Interrupt
35	—	Reserved
36	IIC0_INT	I2C0 Interrupt
37	—	Reserved
38	UART0_INT	UART0 Interrupt
39	—	Reserved
40	T64P1_TINT12	Timer64P1 Interrupt (TINT12)
41	GPIO_B1INT	GPIO Bank 1 Interrupt
42	—	Reserved
43	SPI1_INT	SPI1 Interrupt
44	—	Reserved
45	ECAP0	ECAP0 Interrupt
46	—	Reserved
47	ECAP1	ECAP1 Interrupt
48	T64P1_TINT34	Timer64P1 Interrupt (TINT34)
49	GPIO_B2INT	GPIO Bank 2 Interrupt
50	—	Reserved
51	ECAP2	ECAP2 Interrupt
52	GPIO_B3INT	GPIO Bank 3 Interrupt
53	—	Reserved
54	GPIO_B4INT	GPIO Bank 4 Interrupt
55	EMIFA_INT	EMIFA Interrupt
56	EDMA3_0_CC0_ERRINT	EDMA3_0 Channel Controller 0 Error Interrupt
57	EDMA3_0_TC0_ERRINT	EDMA3_0 Transfer Controller 0 Error Interrupt
58	EDMA3_0_TC1_ERRINT	EDMA3_0 Transfer Controller 1 Error Interrupt
59	GPIO_B5INT	GPIO Bank 5 Interrupt
60	DDR2_MEMERR	DDR2 Memory Error Interrupt
61	MCASP0_INT	McASP0 Combined RX/TX Interrupt
62	GPIO_B6INT	GPIO Bank 6 Interrupt
63	RTC_IRQS	RTC Combined Interrupt
64	T64P0_TINT34	Timer64P0 Interrupt (TINT34)
65	GPIO_B0INT	GPIO Bank 0 Interrupt
66	—	Reserved

**Table 2-1. DSP Interrupt Map (continued)**

Event	Interrupt Name	Source
67	SYSCFG_CHIPINT3	SYSCFG CHIPSIG Register
68-69	—	Reserved
70	PSC0_ALLINT	PSC0
71	PSC1_ALLINT	PSC1
72	GPIO_B7INT	GPIO Bank 7 Interrupt
73	—	Reserved
74	PROTERR	SYSCFG Protection Shared Interrupt
75	GPIO_B8INT	GPIO Bank 8 Interrupt
76-88	—	Reserved
89	MCBSP1_RINT	McBSP1 Receive Interrupt
90	MCBSP1_XINT	McBSP1 Transmit Interrupt
91	EDMA3_1_CC0_INT1	EDMA3_1 Channel Controller 0 Shadow Region 1 Transfer Completion Interrupt
92	EDMA3_1_CC0_ERRINT	EDMA3_1 Channel Controller 0 Error Interrupt
93	EDMA3_1_TC0_ERRINT	EDMA3_1 Transfer Controller 0 Error Interrupt
94-95	—	Reserved
96	INTERR	C674x-Interrupt Control
97	EMC_IDMAERR	C674x-EMC
98-112	—	Reserved
113	PMC_ED	C674x-PMC
114-115	—	Reserved
116	UMC_ED1	C674x-UMC
117	UMC_ED2	C674x-UMC
118	PDC_INT	C674x-PDC
119	SYS_CMPA	C674x-SYS
120	PMC_CMPA	C674x-PMC
121	PMC_CMPA	C674x-PMC
122	DMC_CMPA	C674x-DMC
123	DMC_CMPA	C674x-DMC
124	UMC_CMPA	C674x-UMC
125	UMC_CMPA	C674x-UMC
126	EMC_CMPA	C674x-EMC
127	EMC_BUSERR	C674x-EMC

### 2.2.2.1.1 Interrupt Controller Registers

For more information on the DSP interrupt controller (INTC) registers, see the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

### 2.2.2.1.2 NMI Interrupt

In addition to the interrupts listed in [Table 2-1](#), the DSP also supports a special interrupt that behaves more like an exception, non-maskable interrupt (NMI). The NMI interrupt is controlled by two registers in the System Configuration Module, the chip signal register (CHIPSIG) and the chip signal clear register (CHIPSIG\_CLR).

The NMI interrupt is asserted by writing a 1 to the CHIPSIG4 bit in CHIPSIG. The NMI interrupt is cleared by writing a 1 to the CHIPSIG4 bit in CHIPSIG\_CLR. For more information on CHIPSIG and CHIPSIG\_CLR, see the *System Configuration (SYSCFG) Module* chapter.

### 2.2.2.2 Power-Down Controller (PDC)

The C674x megamodule includes a power-down controller (PDC). The PDC can power-down all of the following components of the C674x megamodule and internal memories of the DSP subsystem:

- C674x CPU
- Level 1 program memory controller (PMC)
- Level 1 data memory controller (DMC)
- Level 2 unified memory controller (UMC)
- Extended memory controller (EMC)
- Internal direct memory access (IDMA) controller
- L1P memory
- L1D memory
- L2 memory

This device supports the static power-down feature from the C674x megamodule. The *TMS320C674x DSP Megamodule Reference Guide (SPRUFK5)* describes the power-down control in more detail.

- Static power-down: The PDC initiates power-down (clock gating) of the entire C674x megamodule and all internal memories immediately upon command from software.

Static power-down (clock gating) affects all components of the C674x megamodule and all internal memories. Software can initiate static power-down by way of a register bit in the power-down controller command register (PDCCMD) of the PDC. For more information on the PDC, see the *TMS320C674x DSP Megamodule Reference Guide (SPRUFK5)*.

### 2.2.2.3 Bandwidth Manager (BWM)

The bandwidth manager (BWM) provides a programmable interface for optimizing bandwidth among the requesters for resources, which include the following:

- EDMA3-initiated DMA transfers (and resulting coherency operations)
- DSP subsystem IDMA-initiated transfers (and resulting coherency operations)
- Programmable cache coherency operations
  - Block based coherency operations
  - Global coherency operations
- CPU direct-initiated transfers
  - Data access (load/store)
  - Program access

The resources include the following:

- L1P memory
- L1D memory
- L2 memory
- Resources outside of the C674x megamodule: external memory, on-chip peripherals, registers

Since any given requestor could potentially block a resource for extended periods of time, the bandwidth manager is implemented to assure fairness for all requesters.

The bandwidth manager implements a weighted-priority-driven bandwidth allocation. Each requestor (EDMA3, DSP subsystem IDMA, CPU, etc.) is assigned a priority level on a per-transfer basis. The programmable priority level has a single meaning throughout the system. There are a total of nine priority levels, where priority zero is the highest priority and priority eight is the lowest priority. When requests for a single resource contend, access is granted to the highest-priority requestor. When the contention occurs for multiple successive cycles, a contention counter assures that the lower-priority requestor gets access to the resource every 1 out of  $n$  arbitration cycles, where  $n$  is programmable. A priority level of -1 represents a transfer whose priority has been increased due to expiration of the contention counter or a transfer that is fixed as the highest-priority transfer to a given resource.

#### 2.2.2.4 Internal DMA (IDMA) Controller

The IDMA controller performs fast block transfers between any two memory locations local to the C674x megamodule. Local memory locations are defined as those in Level 1 program (L1P), Level 1 data (L1D), and Level 2 (L2) memories, or in the external peripheral configuration (CFG) port. The IDMA cannot transfer data to or from the internal DSP memory-mapped register space. The IDMA is fully described in the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

### 2.3 Memory Map

Refer to your device-specific data manual for the addresses of the memory-map registers.

#### 2.3.1 DSP Internal Memory

See the *System Memory* chapter for a description of the DSP internal memory.

#### 2.3.2 External Memory

See the *System Interconnect* chapter and the *System Memory* chapter for a description of the additional memory and peripherals that the DSP has access to.

### 2.4 Advanced Event Triggering (AET)

The C674x megamodule supports advanced event triggering (AET). This capability can be used to debug complex problems as well as understand performance characteristics of user applications. AET provides the following capabilities:

- **Hardware Program Breakpoints:** specify addresses or address ranges that can generate events such as halting the processor or triggering the trace capture.
- **Data Watchpoints:** specify data variable addresses, address ranges, or data values that can generate events such as halting the processor or triggering the trace capture.
- **Counters:** count the occurrence of an event or cycles for performance monitoring.
- **State Sequencing:** allows combinations of hardware program breakpoints and data watchpoints to precisely generate events for complex sequences.



## System Interconnect

---

---

---

Topic	Page
3.1 Introduction .....	61
3.2 System Interconnect Block Diagram .....	62

### 3.1 Introduction

The DSP, the EDMA3 transfer controllers, and the device peripherals are interconnected through a switch fabric architecture (see [Section 3.2](#)). The switch fabric is composed of multiple switched central resources (SCRs) and multiple bridges. The SCRs establish low-latency connectivity between master peripherals and slave peripherals.

Additionally, the SCRs provide priority-based arbitration and facilitate concurrent data movement between master and slave peripherals. Through the SCRs, the DSP can send data to the EMIF without affecting a data transfer between a device peripheral and internal shared memory. Bridges are mainly used to perform bus-width conversion as well as bus operating frequency conversion.

The DSP, the EDMA3 transfer controllers, and the various device peripherals can be classified into two categories: master peripherals and slave peripherals. Master peripherals are typically capable of initiating read and write transfers in the system and do not rely on the EDMA3 or on a CPU to perform transfers to and from them. The system master peripherals include the DSP, the EDMA3 transfer controllers, and HPI. Not all master peripherals may connect to all slave peripherals. The supported connections are designated by an X in [Table 3-1](#).

**Table 3-1. TMS320C6742 DSP System Interconnect Matrix**

Masters		Slaves					
Master	Default Priority	DSP SDMA	EMIFA	DDR2/ mDDR	EDMA3_0_TC0/ TC1	EDMA3_1_TC0	Peripheral Group <sup>(1)</sup>
EDMA3_0_CC0	0				X		
EDMA3_1_CC0	0					X	
EDMA3_0_TC0	0	X	X	X	X	X	X
EDMA3_0_TC1	0	X	X	X	X	X	X
DSP CFG	2				X	X	X
DSP MDMA	2		X	X			
EDMA3_1_TC0	4	X	X	X	X	X	X
HPI	6	X	X	X			X <sup>(2)</sup>

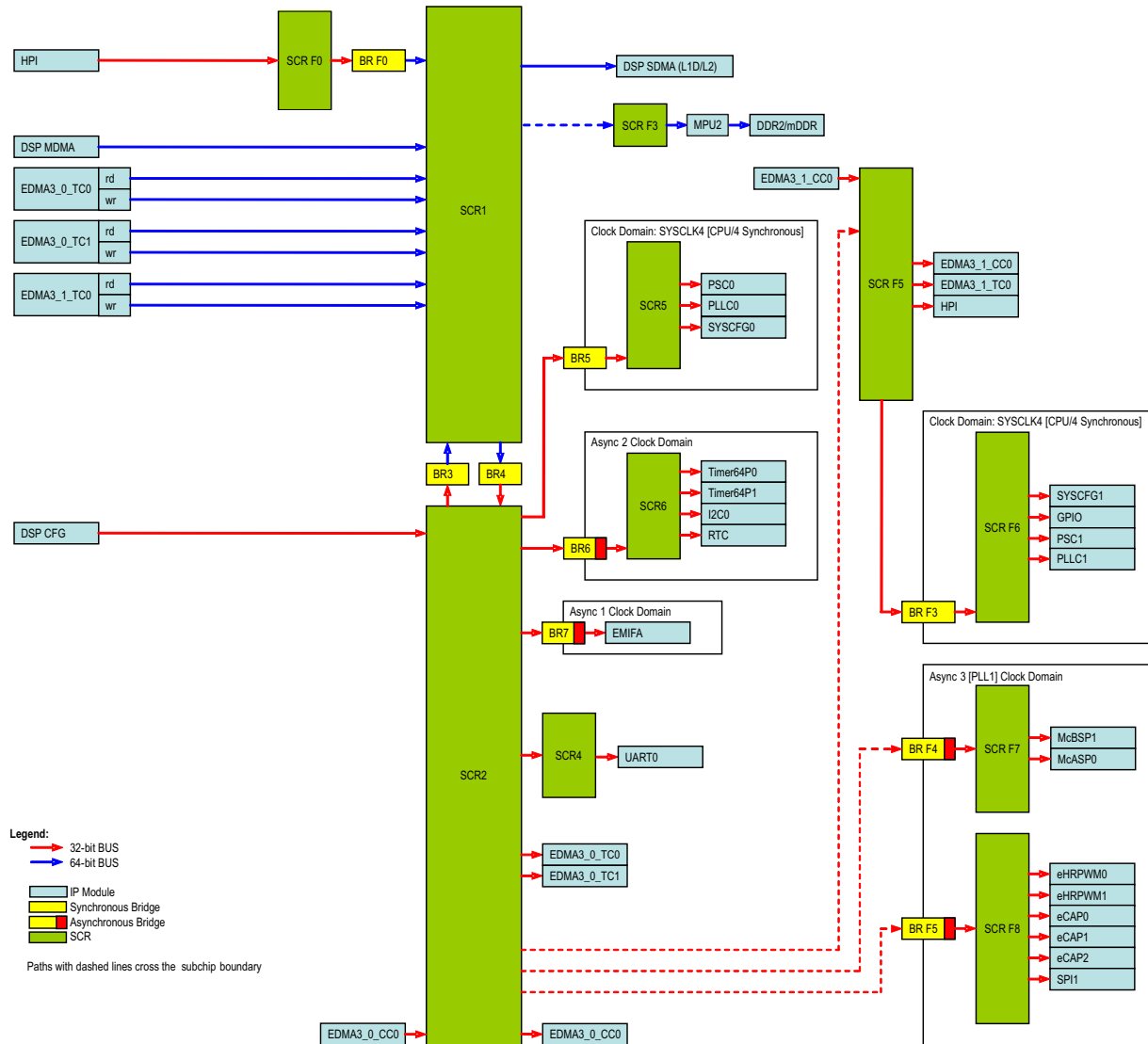
<sup>(1)</sup> Peripheral group: SYSCFG, eCAP0, eCAP1, eCAP2, eHRPWM0, eHRPWM1, GPIO, I2C0, McASP0, McBSP1, PLLC0, PLLC1, PSC0, PSC1, RTC, SPI1, TIMER64P0, TIMER64P1, EDMA3\_0\_CC0, EDMA3\_1\_CC0, UART0, HPI.

<sup>(2)</sup> The HPI does not have access to all registers in the SYSCFG module because it operates with the User Privilege Level.

### 3.2 System Interconnect Block Diagram

Figure 3-1 shows a system interconnect block diagram.

Figure 3-1. System Interconnect Block Diagram



## System Memory

---

---

---

Topic	Page
4.1 Introduction .....	64
4.2 DSP Memories .....	64
4.3 Peripherals .....	64

## 4.1 Introduction

This device has multiple on-chip/off-chip memories and several external device interfaces associated with the DSP and various subsystems. To help simplify software development, a unified memory-map is used wherever possible to maintain a consistent view of device resources across all masters.

For details on the memory addresses, actual memory supported and accessibility by various bus masters, see the detailed memory-map information in the device-specific data manual.

## 4.2 DSP Memories

The DSP internal memories are accessible by the and other master peripherals (as dictated by the connectivity matrix) via the system interconnect through the DSP SDMA port.

The DSP internal memory consists of L1P, L1D, and L2. The DSP internal memory configuration is:

- L1P memory includes 32 KB of RAM. The DSP program memory controller (PMC) allows you to configure part or all of the L1P RAM as normal program RAM or as cache. You can configure cache sizes of 0 KB, 4 KB, 8 KB, 16 KB, or 32 KB of the 32 KB of RAM. The default configuration is 32 KB cache.
- L1D memory includes 32 KB of RAM. The DSP data memory controller (DMC) allows you to configure part of the L1D RAM as normal data RAM or as cache. You can configure cache sizes of 0 KB, 4 KB, 8 KB, 16 KB, or 32 KB of the 32 KB of RAM. The default configuration is 32 KB cache.
- L2 memory includes 64 KB of RAM. The DSP unified memory controller (UMC) allows you to configure part or all of the L2 RAM as normal RAM or as cache. You can configure cache sizes of 0 KB, 4 KB, 8 KB, 16 KB, 32 KB, or 64 KB of the 64 KB of RAM. The default configuration is 64 KB normal RAM.
- L2 memory also includes 1024 KB of ROM.

## External Memories

This device has two external memory interfaces that provide multiple external memory options accessible by the CPU and master peripherals:

- EMIF:
  - 8/16-bit wide asynchronous EMIF module that supports asynchronous devices such as ASRAM, NAND Flash, and NOR Flash (up to 4 devices)
  - 8/16-bit wide NAND Flash with 4-bit ECC (up to 4 devices)
  - 16-bit SDRAM with 128-MB address space
- DDR2/mDDR memory controller:
  - 16-bit DDR2 with up to 256-MB memory address space
  - 16-bit mDDR with up to 256-MB memory address space

## Internal Peripherals

The following peripherals are internal to the DSP subsystem and are only accessible to the DSP:

- DSP interrupt controller (INTC)
- DSP power down controller (PDC)
- Bandwidth manager (BWM)
- Internal DMA (IDMA)

For more information on the internal peripherals, see the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

## 4.3 Peripherals

The DSP has access to all peripherals. See the device-specific data manual for the complete list of peripherals supported on your device.

## Memory Protection Unit (MPU)

---

---

Topic	Page
5.1 Introduction .....	66
5.2 Architecture .....	67
5.3 MPU Registers .....	71



## 5.1 Introduction

This device supports one memory protection unit (MPU2). MPU2 supports the DDR2/mDDR SDRAM.

### 5.1.1 Purpose of the MPU

The memory protection unit (MPU) is provided to manage access to memory. The MPU allows you to define multiple ranges and limit access to system masters based on their privilege ID. The MPU can record a detected fault, or invalid access, and notify the system through an interrupt.

### 5.1.2 Features

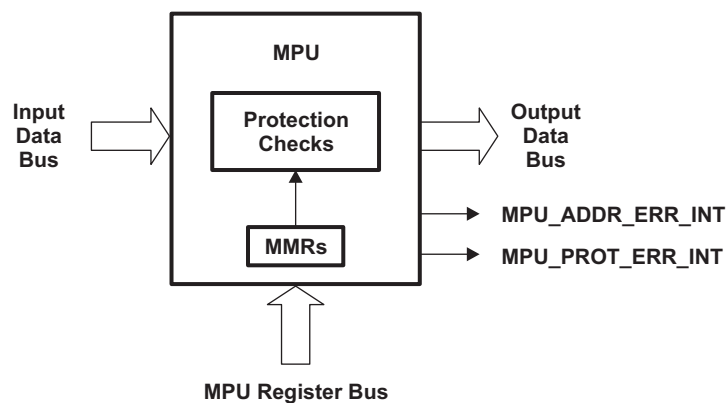
The MPU supports the following features:

- Supports multiple programmable address ranges
- Supports 0 or 1 fixed range
- Supports read, write, and execute access privileges
- Supports privilege ID associations with ranges
- Generates an interrupt when there is a protection violation, and saves violating transfer parameters
- Supports L1/L2 cache accesses
- Supports protection of its own registers

### 5.1.3 Block Diagram

Figure 5-1 shows a block diagram of the MPU. An access to a protected memory must pass through the MPU. During an access, the MPU checks the memory address on the input data bus against fixed and programmable ranges. If allowed, the transfer is passed unmodified to the output data bus. If the transfer fails the protection check then the MPU does not pass the transfer to the output bus but rather services the transfer internally back to the input bus (to prevent a hang) returning the fault status to the requestor as well as generating an interrupt about the fault. The MPU generates two interrupts: an address error interrupt (MPU\_ADDR\_ERR\_INT) and a protection interrupt (MPU\_PROT\_ERR\_INT).

Figure 5-1. MPU Block Diagram



### 5.1.4 MPU Default Configuration

Table 5-1 shows the memory region protected by the MPU2. Table 5-2 shows the configuration of the MPU2.

**Table 5-1. MPU Memory Regions**

Unit	Memory Protection	Memory Region	
		Start Address	End Address
MPU2	DDR2/mDDR SDRAM	C000 0000h	DFFF FFFFh

**Table 5-2. MPU2 Default Configuration**

Setting	MPU2
Default permission	Assume allowed
Number of allowed IDs supported	12
Number of fixed ranges supported	0
Number of programmable ranges supported	12
Compare width	64 KB granularity

## 5.2 Architecture

### 5.2.1 Privilege Levels

The privilege level of a memory access determines what level of permissions the originator of the memory access might have. Two privilege levels are supported: supervisor and user.

Supervisor level is generally granted access to peripheral registers and the memory protection configuration. User level is generally confined to the memory spaces that the OS specifically designates for its use.

DSP CPU instruction and data accesses have a privilege level associated with them. The privilege level is inherited from the code running on the CPU. See the *TMS320C674x DSP CPU and Instruction Set Reference Guide* ([SPRUFE8](#)) for more details on privilege levels of the DSP CPU.

Although master peripherals like the HPI do not execute code, they still have a privilege level associated with them. Unlike the DSP CPU, the privilege level of this peripheral is fixed.

Table 5-3 shows the privilege ID of the CPU and every mastering peripheral. Table 5-3 also shows the privilege level (supervisor vs. user) and access type (instruction read vs. data/DMA read or write) of each master on the device. In some cases, a particular setting depends on software being executed at the time of the access or the configuration of the master peripheral.

**Table 5-3. Device Master Settings**

Master	Privilege ID	Privilege Level	Access Type
EDMA3_0_CC0	Inherited	Inherited	DMA
EDMA3_0_TC0 and EDMA3_0_TC1	Inherited	Inherited	DMA
EDMA3_1_CC0	Inherited	Inherited	DMA
EDMA3_1_TC0	Inherited	Inherited	DMA
DSP	1	Software dependant	Software dependant
HPI	3	User	DMA

## 5.2.2 Memory Protection Ranges

**NOTE:** In some cases the amount of physical memory in actual use may be less than the maximum amount of memory supported by the device. For example, the device may support a total of 512 Mbytes of SDRAM memory, but your design may only populate 128 Mbytes. In such cases, the unpopulated memory range must be protected in order to prevent unintended/disallowed aliased access to protected memory. One of the programmable address ranges could be used to detect accesses to this unpopulated memory.

The MPU divides its assigned memory into address ranges. Each MPU can support one fixed address range and multiple programmable address ranges. The fixed address range is configured to an exact address. The programmable address range allows software to program the start and end addresses.

Each address range has the following set of registers:

- Range start and end address registers (MPSAR and MPEAR): Specifies the starting and ending address of the address range.
- Memory protection page attribute register (MPPA): Use to program the permission settings of the address range.

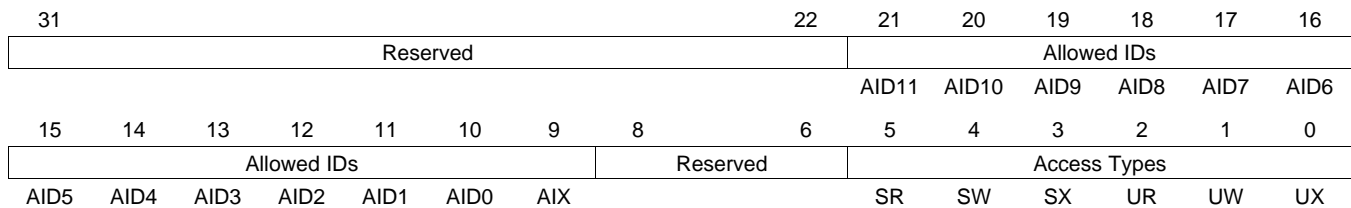
It is allowed to configure ranges such that they overlap each other. In this case, all the overlapped ranges must allow the access, otherwise the access is not allowed. The final permissions given to the access are the lowest of each type of permission from any hit range.

Addresses not covered by a range are either allowed or disallowed based on the configuration of the MPU. The MPU can be configured for assumed allowed or assumed disallowed mode as dictated by the ASSUME\_ALLOWED bit in the configuration register (CONFIG).

## 5.2.3 Permission Structures

The MPU defines a per-range permission structure with three permission fields in a 32-bit permission entry. [Figure 5-2](#) shows the structure of a permission entry.

**Figure 5-2. Permission Fields**



### 5.2.3.1 Requestor-ID Based Access Controls

Each master on the device has an N-bit code associated with it that identifies it for privilege purposes. This privilege ID accompanies all memory accesses made on behalf of that master. That is, when a master triggers a memory access command, the privilege ID will be carried alongside the command.

Each memory protection range has an allowed ID (AID) field associated with it that indicates which requestors may access the given address range. The MPU maps the privilege IDs of all the possible requestors to bits in the allowed IDs field in the memory protection page attribute registers (MPPA).

- AID0 through AID11 are used to specify the allowed privilege IDs.
- An additional allowed ID bit, AIDX, captures access made by all privilege IDs not covered by AID0 through AID11.

When set to 1, the AID bit grants access to the corresponding ID. When cleared to 0, the AID bit denies access to the corresponding requestor.

### 5.2.3.2 Request-Type Based Permissions

The memory protection model defines three fundamental functional access types: read, write, and execute. Read and write refer to data accesses -- accesses originating via the load/store units on the CPU or via a master peripheral. Execute refers to accesses associated with an instruction fetch.

The memory protection model allows controlling read, write, and execute permissions independently for both user and supervisor mode. This results in six permission bits, listed in [Table 5-4](#). For each bit, a 1 permits the access type and a 0 denies access. For example, UX = 1 means that User Mode may execute from the given page. The memory protection unit allows you to specify all six of these bits separately; 64 different encodings are permitted altogether, although programs might not use all of them.

**Table 5-4. Request Type Access Controls**

Bit	Field	Description
5	SR	Supervisor may read
4	SW	Supervisor may write
3	SX	Supervisor may execute
2	UR	User may read
1	UW	User may write
0	UX	User may execute

### 5.2.4 Protection Check

During a memory access, the MPU checks if the address range of the input transfer overlaps one of the address ranges. When the input transfer address is within a range the transfer parameters are checked against the address range permissions.

The MPU first checks the transfers privilege ID against the AID settings. If the AID bit is 0, then the range will not be checked; if the AID bit is 1, then the transfer parameters are checked against the memory protection page attribute register (MPPA) values to detect an allowed access.

For non-debug accesses, the read, write, and execute permissions are also checked. There is a set of permissions for supervisor mode and a set for user mode. For supervisor mode accesses, the SR, SW, and SX bits are checked. For user mode accesses, the UR, UW, and UX bits are checked.

If the transfer address range does not match any address range then the transfer is either allowed or disallowed based on the configuration of the MPU. The MPU can be configured for assumed allowed or assumed disallowed mode as dictated by the ASSUME\_ALLOWED bit in the configuration register (CONFIG).

In the case that a transfer spans multiple address ranges, all the overlapped ranges must allow the access, otherwise the access is not allowed. The final permissions given to the access are the lowest of each type of permission from any hit range. Therefore, if a transfer matches 2 ranges, one that is RW and one that is RX, then the final permission is just R.

The MPU has a special mechanism for handling DSP L1/L2 cache controller read accesses, see [Section 5.2.5](#) for more details.

### 5.2.5 DSP L1/L2 Cache Controller Accesses

A memory read access that originates from the DSP L1/L2 cache is treated differently to allow memory protection to be enforced by the DSP level. This is because a subsequent memory access that hits in the cache does not pass through the MPU. Instead the memory access is serviced directly by the L1/L2 memory controllers.

During a cache memory read, the permission settings stored in the memory protection page attribute registers (MPPA) are passed to the L1/L2 memory controllers along with the read data. The permissions settings returned by the MPU are taken from MPPA that covers the address range of the original request—only the SR, SW, SX, UR, UW, and UX bits are passed. If the request address is covered by multiple address ranges, then the returned value is the logical-AND of all MPPA permissions. If the transfer address range is not covered by an address range then the transfer is either allowed or disallowed based on the configuration of the MPU.

### 5.2.6 MPU Register Protection

Access to the range start and end address registers (MPSAR and MPEAR) and memory protection page attribute registers (MPPA) is also protected. All non-debug writes must be by a supervisor entity. A protection fault can occur from a register write with invalid permissions and this triggers an interrupt just like a memory access.

Faults are not recorded (nor interrupts generated) for debug accesses.

### 5.2.7 Invalid Accesses and Exceptions

When a transfer fails the protection check, the MPU does not pass the transfer to the output bus. The MPU instead services the transfer locally to prevent a hang and returns a protection error to the requestor. The behavior of the MPU depends on whether the access was a read or a write:

- For a read: The MPU returns 0s, a permission value is 0 (no access allowed), a protection error status.
- For a write: The MPU receives all the write data and returns a protection error status.

The MPU captures system faults due to addressing or protection violations in its registers. The MPU can store the fault information for only one fault, so the first detected fault is recorded into the fault registers and an interrupt is generated. Software must use the fault clear register (FLTCLR) to clear the fault status so that another fault can be recorded. The MPU will not record another fault nor generate another interrupt until the existing fault has been cleared. Also, additional faults will be ignored. Faults are not recorded (no interrupts generated) for debug accesses.

### 5.2.8 Reset Considerations

After reset, the memory protection page attribute registers (MPPA) default to 0. This disables all protection features.

### 5.2.9 Interrupt Support

#### 5.2.9.1 Interrupt Events and Requests

The MPU generates two interrupts: an address error interrupt (MPU\_ADDR\_ERR\_INT) and a protection interrupt (MPU\_PROT\_ERR\_INT). The MPU\_ADDR\_ERR\_INT is generated when there is an addressing violation due to an access to a non-existent location in the MPU register space. The MPU\_PROT\_ERR\_INT interrupt is generated when there is a protection violation of either in the defined ranges or to the MPU registers.

The transfer parameters that caused the violation are saved in the MPU registers.

### 5.2.9.2 Interrupt Multiplexing

The interrupts from both MPUs are combined with the boot configuration module into a single interrupt called MPU\_BOOTCFG\_ERR. The combined interrupt is routed to the DSP interrupt controller. [Table 5-5](#) shows the interrupt sources that are combined to make MPU\_BOOTCFG\_ERR.

**Table 5-5. MPU\_BOOTCFG\_ERR Interrupt Sources**

Interrupt	Source
MPU2_ADDR_ERR_INT	MPU2 address error interrupt
MPU2_PROT_ERR_INT	MPU2 protection interrupt
BOOTCFG_ADDR_ERR	Boot configuration address error
BOOTCFG_PROT_ERR	Boot configuration protection error

### 5.2.10 Emulation Considerations

Memory and MPU registers are not protected against emulation accesses.

## 5.3 MPU Registers

[Table 5-6](#) lists the memory-mapped registers for the MPU2.

**Table 5-6. Memory Protection Unit 2 (MPU2) Registers**

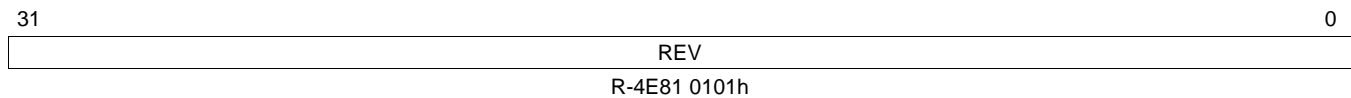
Address	Acronym	Register Description	Section
01E1 5000h	REVID	Revision identification register	<a href="#">Section 5.3.1</a>
01E1 5004h	CONFIG	Configuration register	<a href="#">Section 5.3.2</a>
01E1 5010h	IRAWSTAT	Interrupt raw status/set register	<a href="#">Section 5.3.3</a>
01E1 5014h	IENSTAT	Interrupt enable status/clear register	<a href="#">Section 5.3.4</a>
01E1 5018h	IENSET	Interrupt enable set register	<a href="#">Section 5.3.5</a>
01E1 501Ch	IENCLR	Interrupt enable clear register	<a href="#">Section 5.3.6</a>
01E1 5100h	FXD_MPSAR	Fixed range start address register	<a href="#">Section 5.3.7</a>
01E1 5104h	FXD_MPEAR	Fixed range end address register	<a href="#">Section 5.3.8</a>
01E1 5108h	FXD_MPPA	Fixed range memory protection page attributes register	<a href="#">Section 5.3.9</a>
01E1 5200h	PROG1_MPSAR	Programmable range 1 start address register	<a href="#">Section 5.3.10</a>
01E1 5204h	PROG1_MPEAR	Programmable range 1 end address register	<a href="#">Section 5.3.11</a>
01E1 5208h	PROG1_MPPA	Programmable range 1 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5210h	PROG2_MPSAR	Programmable range 2 start address register	<a href="#">Section 5.3.10</a>
01E1 5214h	PROG2_MPEAR	Programmable range 2 end address register	<a href="#">Section 5.3.11</a>
01E1 5218h	PROG2_MPPA	Programmable range 2 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5220h	PROG3_MPSAR	Programmable range 3 start address register	<a href="#">Section 5.3.10</a>
01E1 5224h	PROG3_MPEAR	Programmable range 3 end address register	<a href="#">Section 5.3.11</a>
01E1 5228h	PROG3_MPPA	Programmable range 3 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5230h	PROG4_MPSAR	Programmable range 4 start address register	<a href="#">Section 5.3.10</a>
01E1 5234h	PROG4_MPEAR	Programmable range 4 end address register	<a href="#">Section 5.3.11</a>
01E1 5238h	PROG4_MPPA	Programmable range 4 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5240h	PROG5_MPSAR	Programmable range 5 start address register	<a href="#">Section 5.3.10</a>
01E1 5244h	PROG5_MPEAR	Programmable range 5 end address register	<a href="#">Section 5.3.11</a>
01E1 5248h	PROG5_MPPA	Programmable range 5 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5250h	PROG6_MPSAR	Programmable range 6 start address register	<a href="#">Section 5.3.10</a>
01E1 5254h	PROG6_MPEAR	Programmable range 6 end address register	<a href="#">Section 5.3.11</a>

**Table 5-6. Memory Protection Unit 2 (MPU2) Registers (continued)**

Address	Acronym	Register Description	Section
01E1 5258h	PROG6_MPPA	Programmable range 6 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5260h	PROG7_MPSAR	Programmable range 7 start address register	<a href="#">Section 5.3.10</a>
01E1 5274h	PROG7_MPEAR	Programmable range 7 end address register	<a href="#">Section 5.3.11</a>
01E1 5268h	PROG7_MPPA	Programmable range 7 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5270h	PROG8_MPSAR	Programmable range 8 start address register	<a href="#">Section 5.3.10</a>
01E1 5274h	PROG8_MPEAR	Programmable range 8 end address register	<a href="#">Section 5.3.11</a>
01E1 5278h	PROG8_MPPA	Programmable range 8 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5280h	PROG9_MPSAR	Programmable range 9 start address register	<a href="#">Section 5.3.10</a>
01E1 5284h	PROG9_MPEAR	Programmable range 9 end address register	<a href="#">Section 5.3.11</a>
01E1 5288h	PROG9_MPPA	Programmable range 9 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5290h	PROG10_MPSAR	Programmable range 10 start address register	<a href="#">Section 5.3.10</a>
01E1 5294h	PROG10_MPEAR	Programmable range 10 end address register	<a href="#">Section 5.3.11</a>
01E1 5298h	PROG10_MPPA	Programmable range 10 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 52A0h	PROG11_MPSAR	Programmable range 11 start address register	<a href="#">Section 5.3.10</a>
01E1 52A4h	PROG11_MPEAR	Programmable range 11 end address register	<a href="#">Section 5.3.11</a>
01E1 52A8h	PROG11_MPPA	Programmable range 11 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 52B0h	PROG12_MPSAR	Programmable range 12 start address register	<a href="#">Section 5.3.10</a>
01E1 52B4h	PROG12_MPEAR	Programmable range 12 end address register	<a href="#">Section 5.3.11</a>
01E1 52B8h	PROG12_MPPA	Programmable range 12 memory protection page attributes register	<a href="#">Section 5.3.12</a>
01E1 5300h	FLTADDRR	Fault address register	<a href="#">Section 5.3.13</a>
01E1 5304h	FLTSTAT	Fault status register	<a href="#">Section 5.3.14</a>
01E1 5308h	FLTCLR	Fault clear register	<a href="#">Section 5.3.15</a>

### 5.3.1 Revision Identification Register (REVID)

The revision ID register (REVID) contains the MPU revision. The REVID is shown in [Figure 5-3](#) and described in [Table 5-7](#).

**Figure 5-3. Revision ID Register (REVID)**


LEGEND: R = Read only; -n = value after reset

**Table 5-7. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4E81 0101h	Revision ID of the MPU.

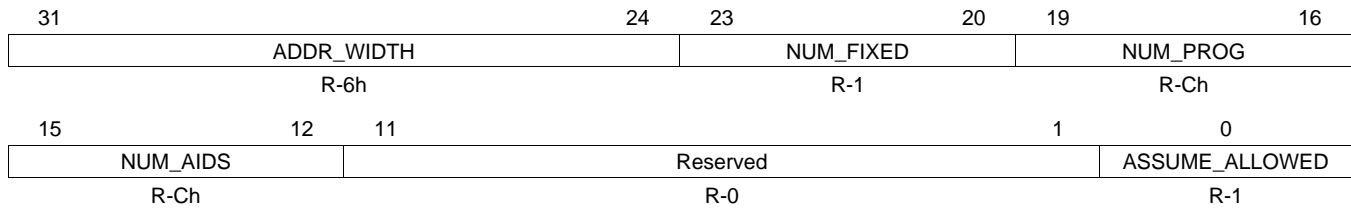


### 5.3.2 Configuration Register (CONFIG)

The configuration register (CONFIG) contains the configuration value of the MPU. The CONFIG is shown in [Figure 5-4](#) and described in [Table 5-8](#).

**NOTE:** Although the NUM\_AIDS bit defaults to 12 (Ch), not all AIDs may be supported on your device. Unsupported AIDs should be cleared to 0 in the memory page protection attributes registers (MPPA). See for a list of AIDs supported on your device.

**Figure 5-4. Configuration Register (CONFIG)**



LEGEND: R = Read only; -n = value after reset

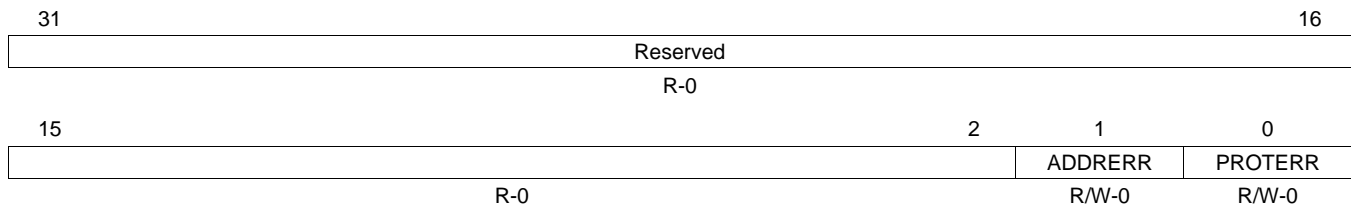
**Table 5-8. Configuration Register (CONFIG) Field Descriptions**

Bit	Field	Value	Description
31-24	ADDR_WIDTH	0-FFh	Address alignment (2 <sup>n</sup> KByte alignment) for range checking.
23-20	NUM_FIXED	0-Fh	Number of fixed address ranges.
19-16	NUM_PROG	0-Fh	Number of programmable address ranges.
15-12	NUM_AIDS	0-Fh	Number of supported AIDs.
11-1	Reserved	0	Reserved
0	ASSUME_ALLOWED	0 1	Assume allowed. When an address is not covered by any MPU protection range, this bit determines whether the transfer is assumed to be allowed or not allowed. Assume is disallowed. Assume is allowed.

### 5.3.3 Interrupt Raw Status/Set Register (IRAWSTAT)

Reading the interrupt raw status/set register (IRAWSTAT) returns the status of all interrupts. Software can write to IRAWSTAT to manually set an interrupt; however, an interrupt is generated only if the interrupt is enabled in the interrupt enable set register (IENSET). Writes of 0 have no effect. The IRAWSTAT is shown in Figure 5-5 and described in Table 5-9.

**Figure 5-5. Interrupt Raw Status/Set Register (IRAWSTAT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

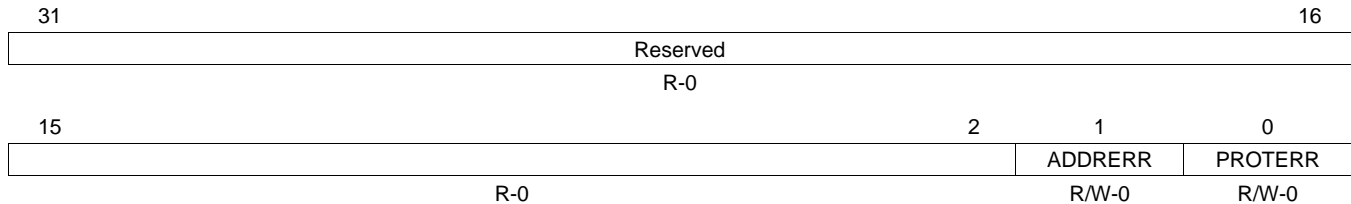
**Table 5-9. Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR	0	Address violation error. Reading this bit reflects the status of the interrupt. Writing 1 sets the status; writing 0 has no effect.
		0	Interrupt is not set.
		1	Interrupt is set.
0	PROTERR	0	Protection violation error. Reading this bit reflects the status of the interrupt. Writing 1 sets the status; writing 0 has no effect.
		0	Interrupt is not set.
		1	Interrupt is set.

### 5.3.4 Interrupt Enable Status/Clear Register (IENSTAT)

Reading the interrupt enable status/clear register (IENSTAT) returns the status of only those interrupts that are enabled in the interrupt enable set register (IENSET). Software can write to IENSTAT to clear an interrupt; the interrupt is cleared from both IENSTAT and the interrupt raw status/set register (IRAWSTAT). Writes of 0 have no effect. The IENSTAT is shown in Figure 5-6 and described in Table 5-10.

**Figure 5-6. Interrupt Enable Status/Clear Register (IENSTAT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

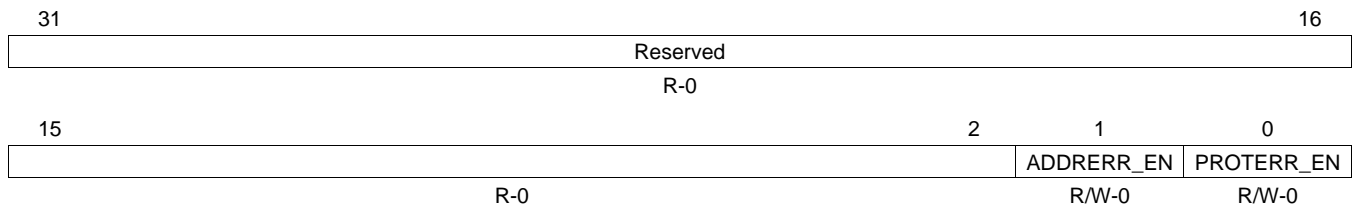
**Table 5-10. Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR	0	Address violation error. If the interrupt is enabled, reading this bit reflects the status of the interrupt. If the interrupt is disabled, reading this bit returns 0. Writing 1 sets the status; writing 0 has no effect.
		1	
0	PROTERR	0	Protection violation error. If the interrupt is enabled, reading this bit reflects the status of the interrupt. If the interrupt is disabled, reading this bit returns 0. Writing 1 sets the status; writing 0 has no effect.
		1	

### 5.3.5 Interrupt Enable Set Register (IENSET)

Reading the interrupt enable set register (IENSET) returns the interrupts that are enabled. Software can write to IENSET to enable an interrupt. Writes of 0 have no effect. The IENSET is shown in [Figure 5-7](#) and described in [Table 5-11](#).

**Figure 5-7. Interrupt Enable Set Register (IENSET)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

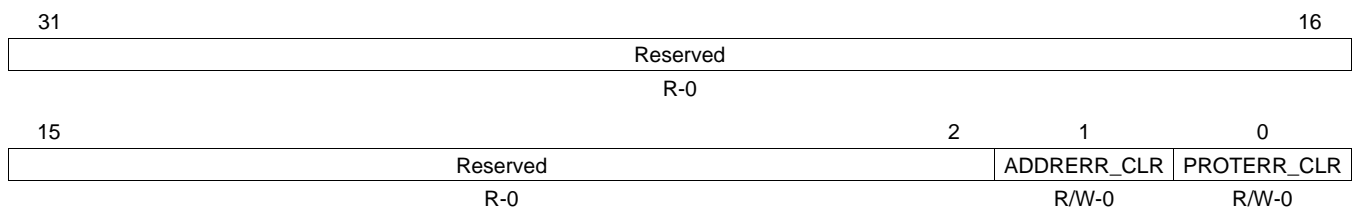
**Table 5-11. Interrupt Enable Set Register (IENSET) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR_EN	0	Address violation error enable. Writing 0 has no effect.
		1	Interrupt is enabled.
0	PROTERR_EN	0	Protection violation error enable. Writing 0 has no effect.
		1	Interrupt is enabled.

### 5.3.6 Interrupt Enable Clear Register (IENCLR)

Reading the interrupt enable clear register (IENCLR) returns the interrupts that are enabled. Software can write to IENCLR to clear/disable an interrupt. Writes of 0 have no effect. The IENCLR is shown in [Figure 5-8](#) and described in [Table 5-12](#).

**Figure 5-8. Interrupt Enable Clear Register (IENCLR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

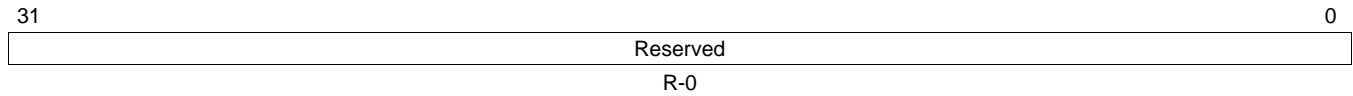
**Table 5-12. Interrupt Enable Clear Register (IENCLR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR_CLR	0	Address violation error disable. Writing 0 has no effect.
		1	Interrupt is cleared/disabled.
0	PROTERR_CLR	0	Protection violation error disable. Writing 0 has no effect.
		1	Interrupt is cleared/disabled.

### 5.3.7 Fixed Range Start Address Register (FXD\_MPSAR)

The fixed range start address register (FXD\_MPSAR) holds the start address for the fixed range. The fixed address range manages access to the DDR2/mDDR SDRAM control registers (B000 0000h–B000 7FFFh). However, these addresses are *not* indicated in FXD\_MPSAR and the fixed range end address register (FXD\_MPEAR), which instead read as 0. The FXD\_MPSAR is shown in [Figure 5-9](#).

**Figure 5-9. Fixed Range Start Address Register (FXD\_MPSAR)**

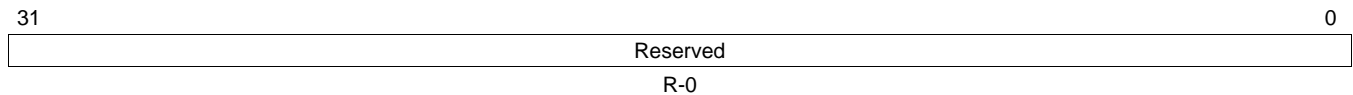


LEGEND: R = Read only; -n = value after reset

### 5.3.8 Fixed Range End Address Register (FXD\_MPEAR)

The fixed range end address register (FXD\_MPEAR) holds the end address for the fixed range. The fixed address range manages access to the DDR2/mDDR SDRAM control registers (B000 0000h–B000 7FFFh). However, these addresses are *not* indicated in FXD\_MPEAR and the fixed range start address register (FXD\_MPSAR), which instead read as 0. The FXD\_MPEAR is shown in [Figure 5-10](#).

**Figure 5-10. Fixed Range End Address Register (FXD\_MPEAR)**



LEGEND: R = Read only; -n = value after reset

### 5.3.9 Fixed Range Memory Protection Page Attributes Register (FXD\_MPPA)

The fixed range memory protection page attributes register (FXD\_MPPA) holds the permissions for the fixed region. This register is writeable by a supervisor entity only. The FXD\_MPPA is shown in [Figure 5-11](#) and described in [Table 5-13](#).

**Figure 5-11. Fixed Range Memory Protection Page Attributes Register (FXD\_MPPA)**

31	Reserved						26	Reserved			25	AID11	AID10	AID9	AID8	AID7	AID6
R-0						R-Fh			R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1		
15	AID5	AID4	AID3	AID2	AID1	AID0	AIDX	Rsvd	Rsvd	Rsvd	SR	SW	SX	UR	UW	UX	
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 5-13. Fixed Range Memory Protection Page Attributes Register (FXD\_MPPA)  
Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-22	Reserved	Fh	Reserved
21-10	AID <sub>n</sub>	0 1	Controls access from ID = n. Access is denied. Access is granted.
9	AIDX	0 1	Controls access from ID > 11. Access is denied. Access is granted.
8	Reserved	0	Reserved
7	Reserved	1	Reserved. This bit must be written as 1.
6	Reserved	1	Reserved. This bit must be written as 1.
5	SR	0 1	Supervisor Read permission. Access is denied. Access is allowed.
4	SW	0 1	Supervisor Write permission. Access is denied. Access is allowed.
3	SX	0 1	Supervisor Execute permission. Access is denied. Access is allowed.
2	UR	0 1	User Read permission. Access is denied. Access is allowed.
1	UW	0 1	User Write permission. Access is denied. Access is allowed.
0	UX	0 1	User Execute permission. Access is denied. Access is allowed.

### 5.3.10 Programmable Range *n* Start Address Registers (PROG<sub>*n*</sub>\_MPSAR)

**NOTE:** In some cases the amount of physical memory in actual use may be less than the maximum amount of memory supported by the device. For example, the device may support a total of 512 Mbytes of SDRAM memory, but your design may only populate 128 Mbytes. In such cases, the unpopulated memory range must be protected in order to prevent unintended/disallowed aliased access to protected memory, especially memory. One of the programmable address ranges could be used to detect accesses to this unpopulated memory.

The programmable range *n* start address register (PROG<sub>*n*</sub>\_MPSAR) holds the start address for the range *n*. The PROG<sub>*n*</sub>\_MPSAR is writeable by a supervisor entity only.

The start address must be aligned on a page boundary. The page size for MPU2 is 64 Kbytes. The size of the page determines the width of the address field in PROG<sub>*n*</sub>\_MPSAR and the programmable range *n* end address register (PROG<sub>*n*</sub>\_MPEAR). For example, to protect a 64-KB page starting at byte address 8001 0000h, write 8001 0000h to PROG<sub>*n*</sub>\_MPSAR and 8001 FFFFh to PROG<sub>*n*</sub>\_MPEAR.

The PROG<sub>*n*</sub>\_MPSAR for MPU2 is shown in [Figure 5-12](#) and described in [Table 5-14](#).

**Figure 5-12. MPU2 Programmable Range *n* Start Address Register (PROG<sub>*n*</sub>\_MPSAR)**

31	16 15	0
START_ADDR		Reserved
R/W-C000h		R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 5-14. MPU2 Programmable Range *n* Start Address Register (PROG<sub>*n*</sub>\_MPSAR) Field Descriptions**

Bit	Field	Value	Description
31-16	START_ADDR	C000h–DFFFh	Start address for range N.
15-0	Reserved	0	Reserved



### 5.3.11 Programmable Range $n$ End Address Registers (PROG $_n$ \_MPEAR)

The programmable range  $n$  end address register (PROG $_n$ \_MPEAR) holds the end address for the range  $n$ . This register is writeable by a supervisor entity only.

The end address must be aligned on a page boundary. The page size for MPU2 is 64 KBytes. The size of the page determines the width of the address field in the programmable range  $n$  start address register (PROG $_n$ \_MPSAR) and PROG $_n$ \_MPEAR. For example, to protect a 64-KB page starting at byte address 8001 0000h, write 8001 0000h to PROG $_n$ \_MPSAR and 8001 FFFFh to PROG $_n$ \_MPEAR.

The PROG $_n$ \_MPEAR for MPU2 is shown in [Figure 5-13](#) and described in [Table 5-15](#).

**Figure 5-13. MPU2 Programmable Range  $n$  End Address Register (PROG $_n$ \_MPEAR)**

31	16 15	0
END_ADDR		Reserved
R/W-DFFFh		R-FFFFh

LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 5-15. MPU2 Programmable Range  $n$  End Address Register (PROG $_n$ \_MPEAR) Field Descriptions**

Bit	Field	Value	Description
31-16	END_ADDR	C000h–DFFFh	Start address for range N.
15-0	Reserved	FFFFh	Reserved

### 5.3.12 Programmable Range $n$ Memory Protection Page Attributes Register (PROG $_n$ MPPA)

The programmable range  $n$  memory protection page attributes register (PROG $_n$ MPPA) holds the permissions for the region  $n$ . This register is writeable only by a supervisor entity. The PROG $_n$ MPPA is shown in Figure 5-14 and described in Table 5-16.

**Figure 5-14. Programmable Range Memory Protection Page Attributes Register (PROG $_n$ MPPA)**

31				26				25				22				21		20		19		18		17		16					
Reserved								Reserved								AID11	AID10	AID9	AID8	AID7	AID6										
R-0								R-Fh								R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1									
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
AID5	AID4	AID3	AID2	AID1	AID0	AIDX	Rsvd	Rsvd	Rsvd	SR	SW	SX	UR	UW	UX																
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1		

LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

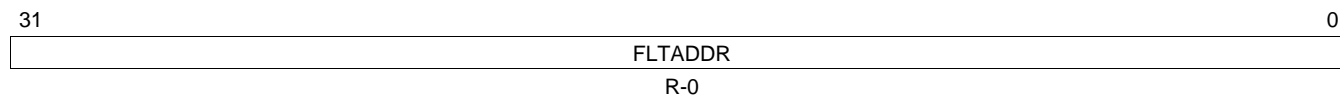
**Table 5-16. Programmable Range Memory Protection Page Attributes Register (PROG $_n$ MPPA) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-22	Reserved	Fh	Reserved
21-10	AID $n$	0 1	Controls access from ID = $n$ . Access is denied. Access is granted.
9	AIDX	0 1	Controls access from ID > 11. Access is denied. Access is granted.
8	Reserved	0	Reserved
7	Reserved	1	Reserved. This bit must be written as 1.
6	Reserved	1	Reserved. This bit must be written as 1.
5	SR	0 1	Supervisor Read permission. Access is denied. Access is allowed.
4	SW	0 1	Supervisor Write permission. Access is denied. Access is allowed.
3	SX	0 1	Supervisor Execute permission. Access is denied. Access is allowed.
2	UR	0 1	User Read permission. Access is denied. Access is allowed.
1	UW	0 1	User Write permission. Access is denied. Access is allowed.
0	UX	0 1	User Execute permission. Access is denied. Access is allowed.

### 5.3.13 Fault Address Register (FLTADDR)

The fault address register (FLTADDR) holds the address of the first protection fault transfer. The FLTADDR is shown in [Figure 5-15](#) and described in [Table 5-17](#).

**Figure 5-15. Fault Address Register (FLTADDR)**



LEGEND: R = Read only; -n = value after reset

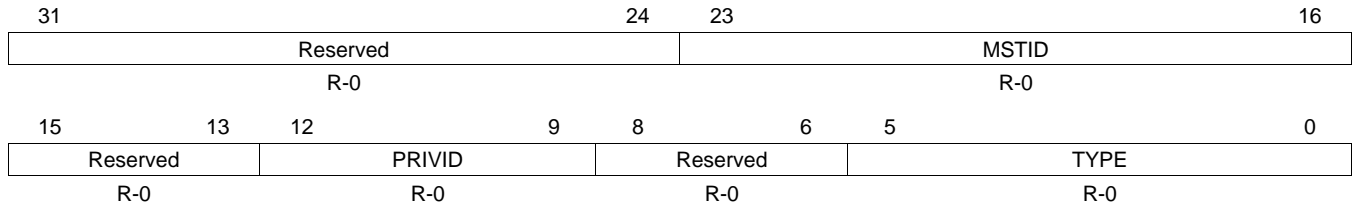
**Table 5-17. Fault Address Register (FLTADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	FLTADDR	0-FFFF FFFFh	Memory address of fault.

### 5.3.14 Fault Status Register (FLTSTAT)

The fault status register (FLTSTAT) holds the status and attributes of the first protection fault transfer. The FLTSTAT is shown in [Figure 5-16](#) and described in [Table 5-18](#).

**Figure 5-16. Fault Status Register (FLTSTAT)**



LEGEND: R = Read only; -n = value after reset

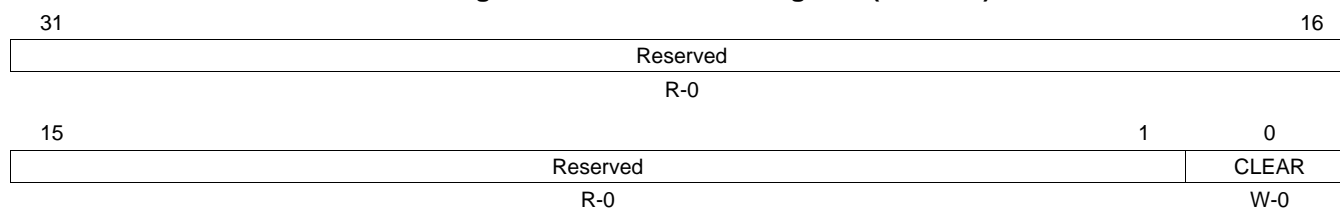
**Table 5-18. Fault Status Register (FLTSTAT) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	MSTID	0-FFh	Master ID of fault transfer.
15-13	Reserved	0	Reserved
12-9	PRIVID	0-Fh	Privilege ID of fault transfer.
8-6	Reserved	0	Reserved
5-0	TYPE	0-3Fh	Fault type. The TYPE bit field is cleared when a 1 is written to the CLEAR bit in the fault clear register (FLTCLR).
		0	No fault.
		1h	User execute fault.
		2h	User write fault.
		3h	Reserved
		4h	User read fault.
		5h-7h	Reserved
		8h	Supervisor execute fault.
		9h-Fh	Reserved
		10h	Supervisor write fault.
		11h	Reserved
		12h	Relaxed cache write back fault.
		13h-1Fh	Reserved
		20h	Supervisor read fault.
		21h-3Eh	Reserved
		3Fh	Relaxed cache line fill fault.

### 5.3.15 Fault Clear Register (FLTCLR)

The fault clear register (FLTCLR) allows software to clear the current fault so that another can be captured in the fault status register (FLTSTAT) as well as produce an interrupt. Only the TYPE bit field in FLTSTAT is cleared when a 1 is written to the CLEAR bit. The FLTCLR is shown in Figure 5-17 and described in Table 5-19.

**Figure 5-17. Fault Clear Register (FLTCLR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 5-19. Fault Clear Register (FLTCLR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLEAR	0	Command to clear the current fault. Writing 0 has no effect.
		1	Clear the current fault.

## Device Clocking

---

---

Topic	Page
6.1 Overview .....	86
6.2 Frequency Flexibility .....	88
6.3 Peripheral Clocking .....	89

## 6.1 Overview

This device requires two primary reference clocks:

- One reference clock is required for the phase-locked loop controllers (PLLs)
- One reference clock is required for the real-time clock (RTC) module.

These reference clocks may be sourced from either the on-board oscillator via an externally supplied crystal or by a direct external oscillator input. For detailed specifications on clock frequency and voltage requirements, see the electrical specifications in your device-specific data manual.

All possible input clocks are described in [Table 6-1](#). The CPU and the majority of the device peripherals operate at fixed ratios of the primary system/CPU clock frequency, as listed in [Table 6-2](#). However, there are two system clock domains that do not require a fixed ratio to the CPU, these are PLL0\_SYSCLK3 and PLL0\_SYSCLK7. [Figure 6-1](#) shows the clocking architecture.

**Table 6-1. Device Clock Inputs**

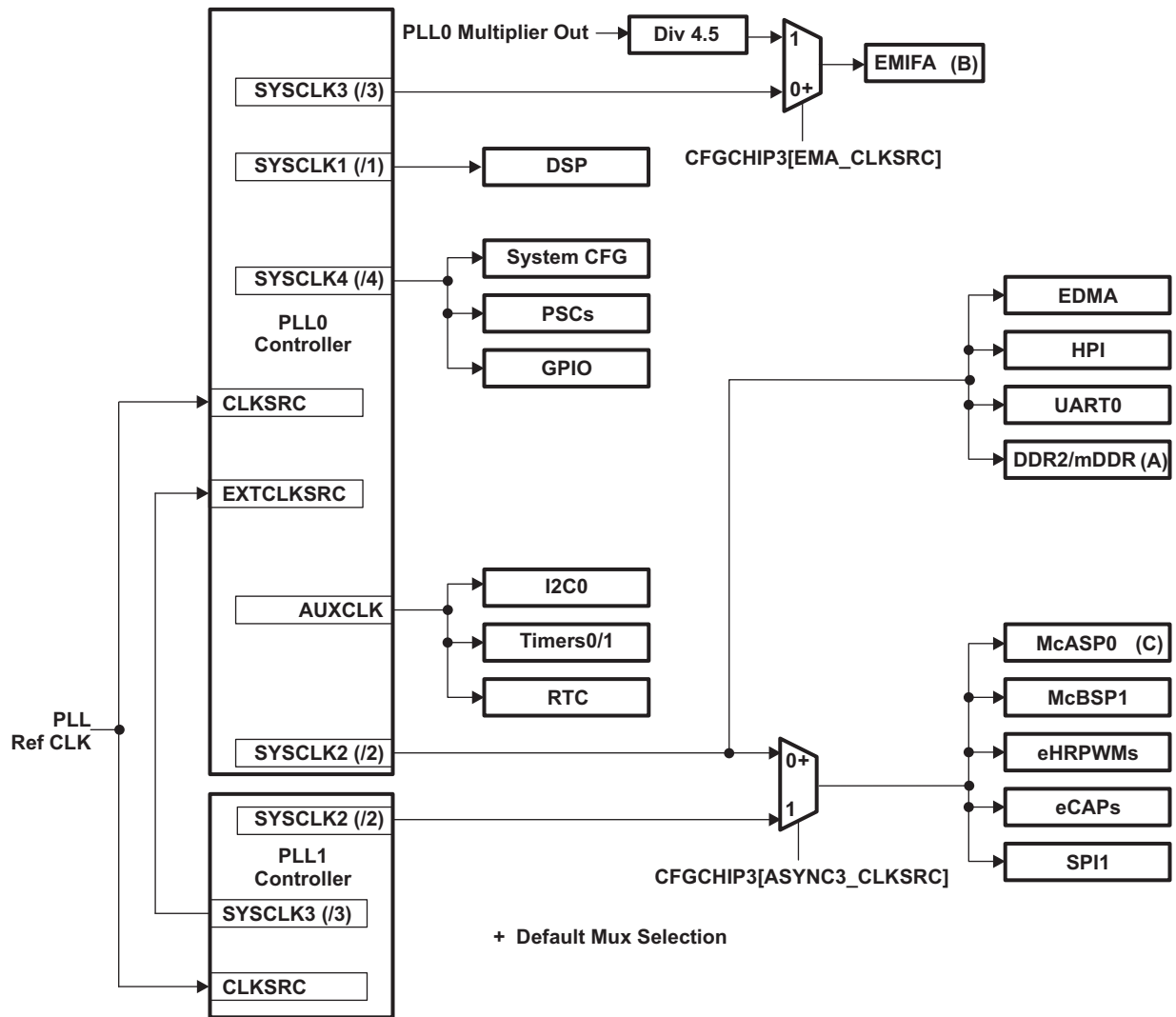
Peripheral	Input Clock Signal Name
Oscillator/PLL	OSCIN
RTC	RTC_XI
JTAG	TCK
I2C0	I2C0_SCL
Timers	TM64Pn_IN12
SPI1	SPI1_CLK
McBSP1	CLKS1, CLKR1, CLKX1
McASP0	ACLKR, AHCLKR, ACLKX, AHCLKX

**Table 6-2. System Clock Domains**

CPU/Device Peripherals	System Clock Domain	Fixed Ratio to CPU Clock Required?	Default Ratio to CPU Clock
DSP	PLL0_SYSCLK1	Yes	1:1
DSP ports, UART0, EDMA, DDR2/mDDR (bus ports), HPI,	PLL0_SYSCLK2	Yes	1:2
EMIFA	PLL0_SYSCLK3	No	1:3
System configuration (SYSCFG), GPIO, PLLs, PSCs	PLL0_SYSCLK4	Yes	1:4
I2C0, Timer64P0/P1, RTC, McASP0 serial clock	PLL0_AUXCLK	Not Applicable	Not Applicable
DDR2/mDDR PHY	PLL1_SYSCLK1	Not Applicable	Not Applicable
PLL0 input reference clock (not configured by default)	PLL1_SYSCLK3	Not Applicable	Not Applicable
ECAPs, eHRPWMs, McBSP1, McASP0, SPI1	ASYNC3	Not Applicable	Not Applicable



Figure 6-1. Overall Clocking Diagram



- A See Section 6.3.1 for DDR2/mDDR clocking.
- B See Section 6.3.2 for EMIFA clocking.
- C See Section 6.3.3 for McASP clocking.

## 6.2 Frequency Flexibility

There are two PLLs on the device with similar architecture and behavior. Each PLL has two clocking modes:

- PLL Bypass
- PLL Active

When the PLL is in Bypass mode, the reference clock supplied on OSCIN serves as the clock source from which all of the system clocks (SYSCLK1 to SYSCLK7) are derived. This means that when the PLL is in Bypass mode, the reference clock supplied on OSCIN passes directly to the system of PLLDIV blocks that creates each of the system clocks. For PLL0 only, the EXTCLKSRC bit in PLLCTL can be configured to use PLL1\_SYSCLK3 as the Bypass mode reference clock.

When the PLL operates in Active mode, the PLL is enabled and the PLL multiplier setting is used to multiply the input clock frequency supplied on the OSCIN pin up to the desired frequency. It is this multiplied frequency that all system clocks are derived from in PLL Active mode.

The output of the PLL multiplier passes through a post divider (POSTDIV) block and then is applied to the system of PLLDIV blocks that creates each of the system clock domains (SYSCLK1 to SYSCLK7). Each SYSCLK $n$  has a PLLDIV $n$  block associated with it. See the *Phase-Locked Loop Controller (PLLC)* chapter for more details on the PLL.

The combination of the PLL multiplier, POSTDIV, and PLLDIV blocks provides flexibility in the frequencies that the system clock domains support. This flexibility does have limitations, as follows:

- OSCIN input frequency is limited to a supported range.
- The output of the PLL Multiplier must be within the range specified in the device-specific data manual.
- The output of each PLLDIV block must be less than or equal to the maximum device frequency specified in the device-specific data manual.

---

**NOTE:** The above limitations are provided here as an example and are used to illustrate the recommended configuration of the PLL controller. These limitations may vary based on core voltage and between devices. See the device-specific data manual for more details.

---

[Table 6-3](#) shows examples of possible PLL multiplier settings, along with the available PLL post-divider modes. The PLL post-divider modes are defined by the value programmed in the RATIO field of the PLL post-divider control register (POSTDIV). For Div1, Div2, Div3, and Div4 modes, the RATIO field would be programmed to 0, 1, 2, and 3, respectively. The Div1, Div2, Div3, and Div4 modes are shown here as an example. Additional post-divider modes are supported and are documented in the *Phase-Locked Loop Controller (PLLC)* chapter.

---

**NOTE:** PLL power consumption increases as the output frequency of the PLL multiplier increases. To decrease PLL power consumption, the lowest PLL multiplier (PLLM) setting should be chosen that achieves the desired frequency. For example, if 200 MHz is the desired CPU operating frequency and the OSCIN frequency is 25 MHz; lower power consumption is achieved by choosing a PLLM setting of  $\times 16$  and a post-divider (POSTDIV) setting of  $/2$  instead of a PLLM setting of  $\times 24$  and a POSTDIV setting of  $/3$ , even though both of these modes would result in a CPU frequency of 200 MHz.

---

**Table 6-3. Example PLL Frequencies**

OSCIN Frequency	PLL Multiplier	Multiplier Frequency	Div1	Div2	Div3	Div4
20	30	600 MHz	600	300	200	150
24	25	600 MHz	600	300	200	150
25	24	600 MHz	600	300	200	150
30	20	600 MHz	600	300	200	150
20	25	500 MHz	500	250	167	125
24	20	480 MHz	480	240	160	120
25	18	450 MHz	450	225	150	112.5
30	14	420 MHz	420	210	140	105
25	16	400 MHz	400	200	133	100

## 6.3 Peripheral Clocking

### 6.3.1 DDR2/mDDR Memory Controller Clocking

The DDR2/mDDR memory controller requires two input clocks to source VCLK and 2X\_CLK (see [Figure 6-2](#)):

- VCLK is sourced from PLL0\_SYSCLOCK2/2 that clocks the command FIFO, write FIFO, and read FIFO of the DDR2/mDDR memory controller. From this, VCLK drives the interface to the peripheral bus.
- 2X\_CLK is sourced from PLL1\_SYSCLOCK1.

2X\_CLK clock is again divided down by 2 in the DDR PHY controller to generate a clock called MCLK. The MCLK domain consists of the DDR2/mDDR memory controller state machine and memory-mapped registers. This clock domain is clocked at the rate of the external DDR2/mDDR memory, 2X\_CLK/2.

[Table 6-4](#) shows example PLL register settings based on the OSCIN reference clock frequency of 25 MHz. From these example configurations, the following observations are made:

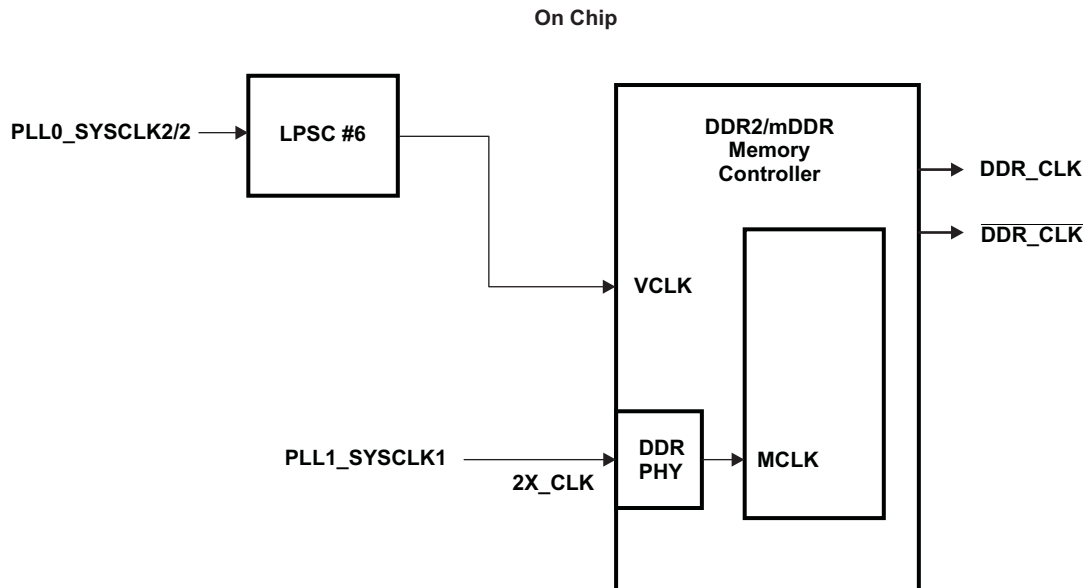
- To achieve the maximum frequency (150 MHz) supported by the DDR2/mDDR memory controller and the typical CPU frequency of 300 MHz, the output of the PLL multiplier should be set to be 300 MHz and the DDR\_CLK source should be set to PLL1\_SYSCLOCK1.
- The frequency of the PLL1 direct output clock is fixed at the output frequency of the PLL1 multiplier block.
- The PLLDIV1 block that sets the divider ratio for SYSCLOCK1 can be changed to achieve various clock frequencies.
- For certain PLL1 multiplier and PLL1 post-divider control register (POSTDIV) settings, a higher clock frequency can be achieved by selecting SYSCLOCK1 as the clock source for 2X\_CLK.

If the DDR2/mDDR memory controller is not in use and the DDR\_CLK and  $\overline{\text{DDR\_CLK}}$  are used in the application as a free running clock that could be used by an FPGA or for some other purpose, then 2X\_CLK should be used as the source for DDR\_CLK and  $\overline{\text{DDR\_CLK}}$  and VCLK should be gated off. This allows clock gating of the majority of the logic in the DDR2/mDDR memory controller via the LPSC while still providing a clock on the DDR\_CLK and  $\overline{\text{DDR\_CLK}}$ .

---

**NOTE:** DDR\_CLK and  $\overline{\text{DDR\_CLK}}$  are output clock signals.

---

**Figure 6-2. DDR2/mDDR Memory Controller Clocking Diagram**

**Table 6-4. DDR2/mDDR Memory Controller MCLK Frequencies**

OSCIN Frequency	PLL1 Multiplier Register Setting	PLL1 Multiplier Frequency	PLL1 Post Divider Mode <sup>(1)</sup>	PLL1 POSTDIV Output Frequency	PLL1 PLLDIV1 Register Setting	PLL1_SYSCLK1	MCLK
24	18h	600 MHz	Div2	300 MHz	8000h	300 MHz	150 MHz
24	15h	528 MHz	Div2	264 MHz	8000h	264 MHz	132 MHz
24	14h	504 MHz	Div2	252 MHz	8000h	252 MHz	126 MHz

<sup>(1)</sup> See [Section 6.2](#) for explanation of POSTDIV divider modes.

### 6.3.2 EMIFA Clocking

EMIFA requires a single input clock source. The EMIFA clock can be sourced from either PLL0\_SYSCLK3 or DIV4P5 (see Figure 6-3). The EMA\_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module controls whether PLL0\_SYSCLK3 or DIV4P5 is selected as the clock source for EMIFA.

Selecting the appropriate clock source for EMIFA is determined by the desired clock rate. Table 6-5 shows example PLL register settings and the resulting DIV4P5 and PLL0\_SYSCLK3 frequencies based on the OSCIN reference clock frequency of 25 MHz. From these example configurations, the following observations can be made:

- To achieve a typical frequency of 100 MHz supported by EMIFA and the typical CPU frequency of 300 MHz, the output of the PLL multiplier should be set to 600 MHz and the EMA\_CLK source should be set to PLL0\_SYSCLK3 with the PLLDIV3 register set to 3.
- The frequency of the DIV4P5 clock is fixed at the output frequency of the PLL multiplier block divided by 4.5.
- The PLLDIV3 block that sets the divider ratio for PLL0\_SYSCLK3 can be changed to achieve various clock frequencies.

Figure 6-3. EMIFA Clocking Diagram

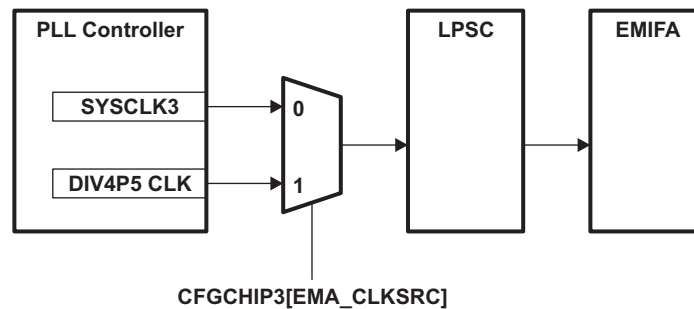


Table 6-5. EMIFA Frequencies

OSCIN Frequency	PLL Multiplier Register Setting	Multiplier Frequency	Post Divider Mode <sup>(1)</sup>	POSTDIV Output Frequency	DIV4P5	PLLDIV3 Register Setting	PLL0_SYSCLK3
25	24	600 MHz	Div2	300 MHz	133 MHz	2	100 MHz
			Div3	200 MHz	133 MHz	2	66.6 MHz
						1	100 MHz
			Div4	150 MHz	133 MHz	1	75 MHz
25	18	450 MHz	Div2	225 MHz	100 MHz	3	56.3 MHz
						2	75 MHz
			Div3	150 MHz	100 MHz	1	75 MHz
			Div4	112.5 MHz	100 MHz	1	56.3 MHz
					0	112.5 MHz	
25	16	400 MHz	Div2	200 MHz	89 MHz	2	66.6 MHz
						1	100 MHz
			Div3	133 MHz	89 MHz	1	66.5 MHz
			Div4	100 MHz	89 MHz	0	100 MHz

<sup>(1)</sup> See Section 6.2 for explanation of POSTDIV divider modes.

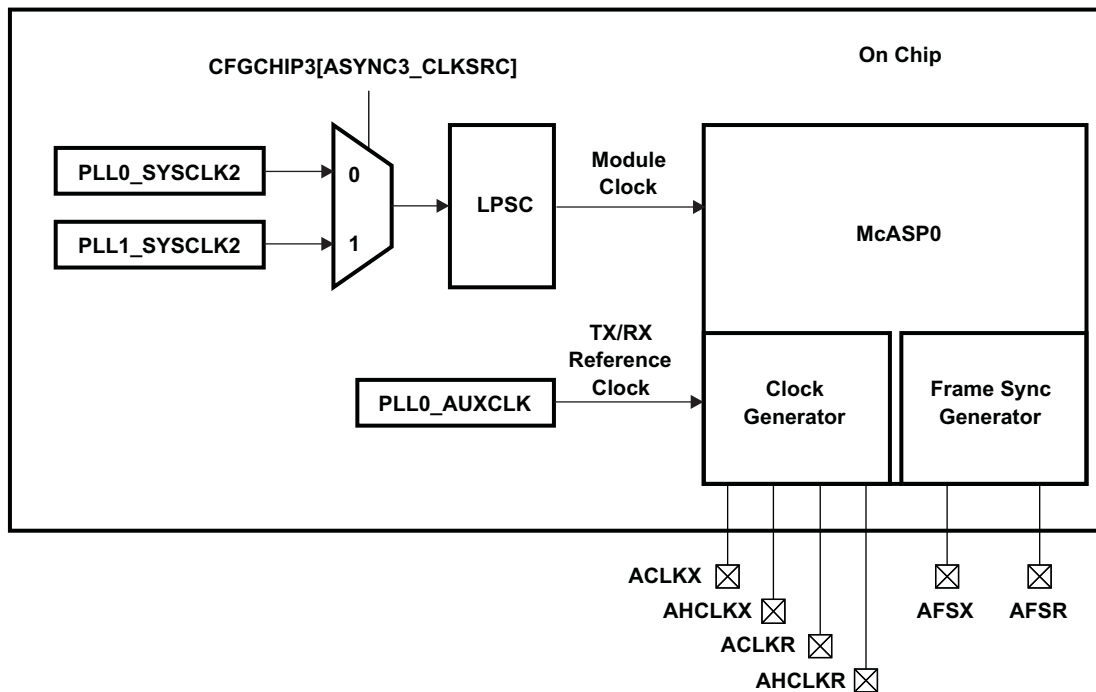
### 6.3.3 McASP Clocking

As shown in [Figure 6-4](#), the McASP peripheral requires multiple clock sources. Internally, the module clock is selected to be either PLL0\_SYSCLK2 or PLL1\_SYSCLK2 by configuring the ASYNC3\_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module.

The transmit and receive clocks are sourced internally or externally by configuring the McASP clock control registers ACLKCTL, AHCLKCTL, ACLKXCTL, and AHCLKXCTL. If an external clock is driven into a high-frequency master clock (AHCLKX or AHCLKR), the McASP module allows for a mixed clock mode where the associated lower frequency clock (ACLKX or ACLKR) can be derived from the high-frequency master clock through a programmable divider.

When the internal clock source option is selected, the transmit and receive clocks are derived from the PLL0\_AUXCLK clock through programmable dividers.

**Figure 6-4. McASP Clocking Diagram**



### 6.3.4 I/O Domains

The I/O domains refer to the frequencies of the peripherals that communicate through device pins. In many cases, there are frequency requirements for a peripheral pin interface that are set by an outside standard and must be met. It is not necessarily possible to obtain these frequencies from the on-chip clock generation circuitry, so the frequencies must be obtained from external sources and are asynchronous to the CPU frequency by definition.

The peripherals can be divided into the following groups, depending upon their clock requirements, as shown in [Table 6-6](#).

**Table 6-6. Peripherals**

Peripheral Group	Peripheral Group Definition	Peripherals Contained within Group	Source of Peripheral Clock
RTC	Operates off of a dedicated 32 kHz crystal oscillator.	RTC	—
Fixed-Frequency Peripherals	As the name suggests, fixed-frequency peripherals have a fixed-frequency. They are fed the AUXCLK directly from the oscillator input.	Timer64P0/P1 I2C0	— —
Synchronous Peripherals	Synchronous peripherals have their frequencies derived from the CPU clock frequency. The peripheral system clock frequency changes accordingly, if the PLL0 frequency changes. Most synchronous peripherals have internal dividers so they can generate their required clock frequencies.	HPI UART0 GPIO	PLL0_SYSCLK2 PLL0_SYSCLK2 PLL0_SYSCLK4
Asynchronous Peripherals	Asynchronous peripherals are not required to operate at a fixed ratio of the CPU clock.	eCAPs eHRPWMs EMIFA DDR2/mDDR	ASYNC3 ASYNC3 DIV_4P5 or PLL0_SYSCLK3 PLL1_SYSCLK1 or PLL1 Direct Output
Synchronous/Asynchronous Peripherals	Synchronous/asynchronous peripherals can be run with either internally generated synchronous clocks, or externally generated asynchronous clocks.	McASP0 McBSP1 SPI1	ASYNC3 or Peripheral Serial Clock ASYNC3 or Peripheral Serial Clock ASYNC3 or Peripheral Serial Clock



## Phase-Locked Loop Controller (PLL)

---

---

Topic	Page
7.1 Introduction .....	95
7.2 PLL Controllers.....	95
7.3 PLLC Registers .....	100

## 7.1 Introduction

This device has two phase-locked loop (PLL) controllers, PLLC0 and PLLC1. These PLL controllers provide clock signals to most of the components of the device through various clock dividers.

Both PLL0 and PLL1 provide the following:

- Glitch-free transitions when clock settings are changed
- Domain clock alignment
- Clock gating
- PLL power-down

The clock outputs generated by the PLL controllers are:

- Domain clocks: PLL0\_SYSCLK[1-7] and PLL1\_SYSCLK[1-3]
- Auxiliary clock (PLL0\_AUXCLK) from the PLLC0 reference clock source

Dividers that can be used for the PLL controllers are:

- Pre-PLL divider: PREDIV
- Post-PLL divider: POSTDIV
- SYSCLK divider: D1, ..., Dn

Various other control signals supported are:

- PLL multiplier: PLLM
- Software-programmable PLL bypass: PLEN

## 7.2 PLL Controllers

PLL0 and PLL1 share the same internal architecture so they also share the same approach for mode configuration.

PLL0 provides the primary system clock to the device. PLL0 operations are software programmable through the PLL controller 0 (PLLC0) registers.

PLL1 provides the reference clocks to various peripherals (including DDR2/mDDR) and may generate clocks that are asynchronous to the PLL0 clocks. PLL1 operations are software programmable through the PLL controller 1 (PLLC1) registers.

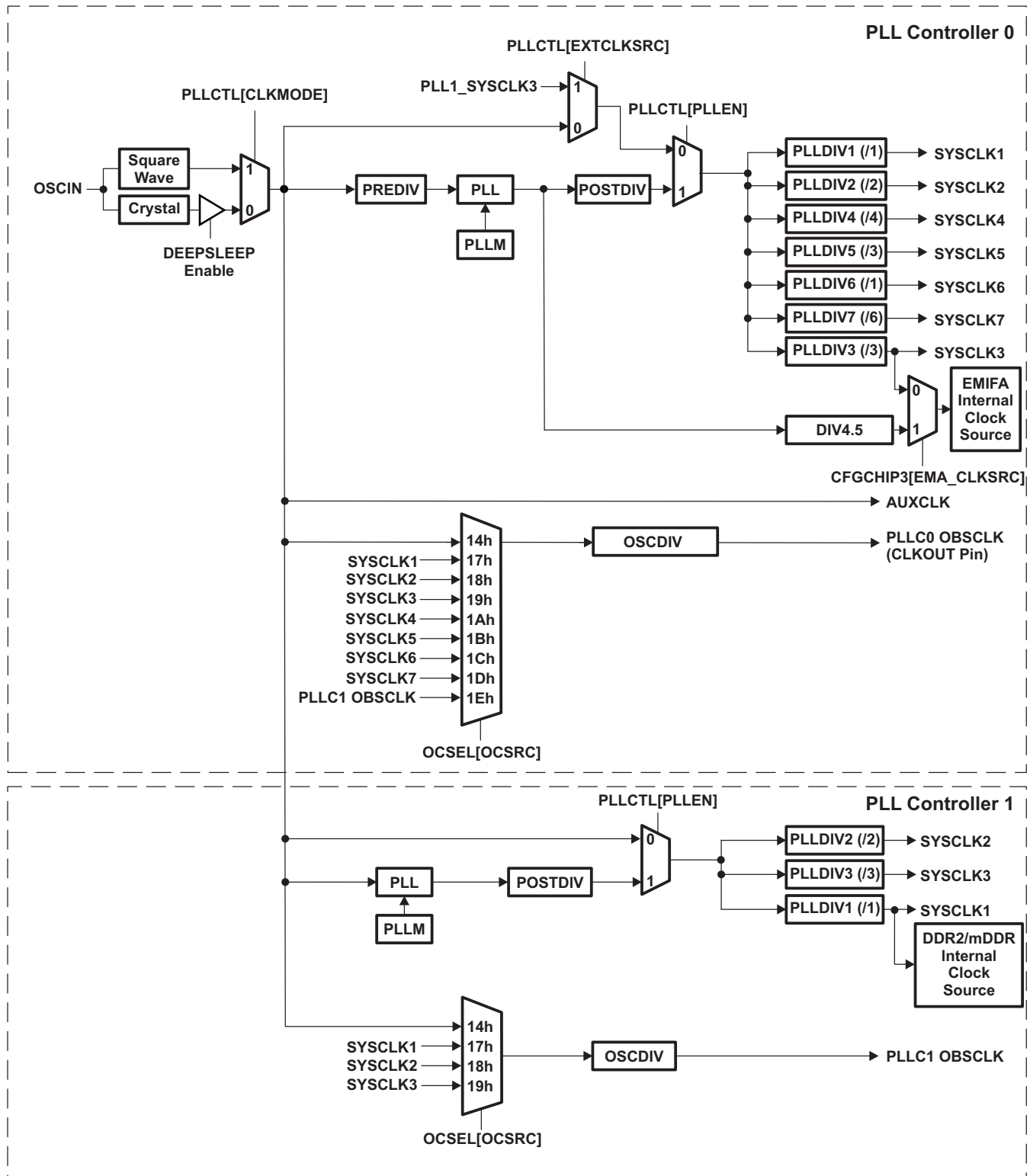
[Figure 7-1](#) shows the PLLC0 and PLLC1 architecture.

The PLL0 and PLL1 multipliers are controlled by their respective PLL multiplier control register (PLLM). The PLLM defaults to a multiplier value of 13h at power-up, which results in a PLL multiplier of 20x. The PLL0 and PLL1 output clocks may be divided-down for slower device operation using the PLL post-divider control register (POSTDIV). The POSTDIV has a default value of /2, but may be modified through software (using the RATIO field in POSTDIV) to achieve lower device operation frequencies. The default PLLM and POSTDIV settings produce a 300-MHz PLL output clock when given a 30-MHz clock source.

At power-up, PLL0 and PLL1 are powered-down/disabled and must be powered-up by software through the PLLPWRDN bit in their respective PLL control register (PLLCTL). Before each PLL completes the power-up and frequency-lock sequence, the system operates in bypass mode by default and the system clock (OSCIN) is provided directly from an input reference clock (square wave or internal oscillator) selected by the CLKMODE bit in PLLCTL. After the power-up and frequency-lock sequences are complete, software can switch the device to PLL mode operation (set the PLEN bit in PLLCTL to 1).

The PLL controller registers are listed in [Section 7.3](#).

Figure 7-1. PLLC Structure



### 7.2.1 Device Clock Generation

The PLL controllers (PLL0 and PLL1) manage the clock ratios, alignment, and gating for the device system clocks. Various PLL mode attributes such as pre-division, multiplier, and post-division are software programmable through the PLL controller registers. Additionally, the reset controller in PLL0 manages reset propagation through the device, clock alignment, and test points.

The PLL0 stage in PLL0 and PLL1 is capable of providing frequencies greater than what the SYSCLK dividers can handle. The POSTDIV stage should be programmed to keep the input to the SYSCLK dividers within operating limits. See the device datasheet for the maximum operating frequencies.

PLL0 and PLL1 generate several clocks for use by the various processors and modules. These reference clocks are summarized in [Table 7-1](#). Some output clock dividers require fixed values so that clock ratios between various device components are maintained regardless of PLL or bypass frequency.

**Table 7-1. System PLLC Output Clocks**

Output Clock	Used by	Default Ratio (relative to PLL <sub>n</sub> _SYSCLK1)	Fixed Clock Ratio
<b>PLL0<sup>(1)</sup></b>			
PLL0_SYSCLK1	DSP	/1	Yes
PLL0_SYSCLK2	DSP ports, UART0, EDMA, DDR2/mDDR (bus ports), HPI,	/2	Yes
PLL0_SYSCLK3 <sup>(2)</sup>	EMIFA	/3	No
PLL0_SYSCLK4	System configuration (SYSCFG), GPIO, PLLs, PSCs	/4	Yes
PLL0_SYSCLK5	Not used	/3	No
PLL0_SYSCLK6	Not used	/1	Yes
PLL0_AUXCLK	I2C0, Timer64P0/P1, RTC, McASP0 serial clock	PLL bypass clock	No
PLL0_OBSCLK	Observation clock (OBSCLK) source	Pin configurable	No
<b>PLL1</b>			
PLL1_SYSCLK1	DDR2/mDDR PHY	/1 or disabled	No
PLL1_SYSCLK2 <sup>(3)</sup>	ECAPs, eHRPWMs, McBSP1, McASP0, SPI1 (all these modules use PLL0_SYSCLK2 by default)	/2 or disabled	No
PLL1_SYSCLK3 <sup>(4)</sup>	PLL0 input reference clock (not configured by default)	/3 or disabled	No

<sup>(1)</sup> The divide values in PLL0 for PLL0\_SYSCLK1/PLL0\_SYSCLK6, PLL0\_SYSCLK2, and PLL0\_SYSCLK4 can be changed for power savings, but the device must maintain the 1:2:4 clock ratios between the clock domains.

<sup>(2)</sup> PLL0 supports an additional post-divider value of /4.5 that can be used for EMIFA clock generation. When this /4.5 value is used, the resulting clock will not have a 50% duty cycle. Instead, the duty cycle will be 44.4%. The EMIFA uses PLL0\_SYSCLK3 by default, but can be configured to use a /4.5 divide-down of PLL0\_PLLOUT instead of PLL0\_SYSCLK3 by programming the EMA\_CLKSRC and DIV45PENA bits in the chip configuration 3 register (CFGCHIP3) of the system configuration (SYSCFG) module.

<sup>(3)</sup> The ASYNC3 modules use PLL0\_SYSCLK2 by default, but all these modules can be configured as a group to use PLL1\_SYSCLK2 by programming the ASYNC3\_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the system configuration (SYSCFG) module.

<sup>(4)</sup> The PLL0 input clock source can be configured to use PLL1\_SYSCLK3 instead of OSCIN by programming the EXTCLKSRC bit in the PLL0 PLL control register (PLLCTL). The PLL1 input clock source will also be OSCIN.

## 7.2.2 Steps for Programming the PLLs

Note that there is a lock mechanism implemented to protect the PLL controller registers. See [Section 7.2.2.1](#) for information on unlocking the PLL controller registers.

Refer to the appropriate subsection on how to program the PLL clocks:

- If the PLL is powered down (PLL\_PWRDN bit in PLL\_CTL is set to 1), follow the full PLL initialization procedure in [Section 7.2.2.2](#).
- If the PLL is not powered down (PLL\_PWRDN bit in PLL\_CTL is cleared to 0), follow the sequence in [Section 7.2.2.3](#) to change the PLL multiplier.
- If the PLL is already running at a desired multiplier and only the SYSCLK dividers will be updated, follow the sequence in [Section 7.2.2.4](#).

Note that the PLLs are powered down after a Power-on Reset (POR). The PLLs are not powered down after a Warm Reset (RESET), but the PLEN bit in PLL\_CTL is cleared to 0 (bypass mode) and the PLLDIVx registers are reset to default values.

### 7.2.2.1 Locking/Unlocking PLL Register Access

A lock mechanism is implemented on the device to prevent inadvertent writes to the PLL controller registers. This provides protection from stopping modules when the module clocks are disabled. For example, the watchdog timer that runs on the PLL0\_AUXCLK will stop if this PLL clock is unintentionally disabled.

The PLL lock bits are located within the system configuration (SYSCFG) module:

- When set, the PLL\_MASTER\_LOCK bit in the chip configuration 0 register (CFGCHIP0) locks PLLC0.
- When set, the PLL1\_MASTER\_LOCK bit in the chip configuration 3 register (CFGCHIP3) locks PLLC1.

Because the SYSCFG module has its own lock mechanism, the SYSCFG module must be unlocked first by writing to the KICK0R and KICK1R registers before the PLL lock bits can be cleared. Like the KICK registers, the PLL lock bits can only be modified while in a privileged mode. See the *System Configuration (SYSCFG) Module* chapter for information on privilege type and the KICK0R and KICK1R registers.

---

**NOTE:** The PLL\_MASTER\_LOCK bit in CFGCHIP0 and the PLL1\_MASTER\_LOCK bit in CFGCHIP3 default to unlocked after reset, so the following procedure is only required if the PLLs have been locked (set to 1).

---

To modify the PLL controller registers, use the following sequence:

1. Write the correct key values to KICK0R and KICK1R registers.
2. Clear the PLL\_MASTER\_LOCK bit in CFGCHIP0 and/or the PLL1\_MASTER\_LOCK bit in CFGCHIP3, as required.
3. Configure the desired PLL controller register values.
4. Set the PLL\_MASTER\_LOCK bit in CFGCHIP0 and/or the PLL1\_MASTER\_LOCK bit in CFGCHIP3, as required.
5. Write an incorrect key value to the KICK0R and KICK1R registers.

### 7.2.2.2 Initializing PLL Mode from PLL Power Down

If the PLL is powered down (PLLPWDN bit in PLLCTL is set to 1), perform the following procedure to initialize the PLL:

1. Program the CLKMODE bit in PLLC0 PLLCTL.
2. Switch the PLL to bypass mode:
  - (a) Clear the PLENSRC bit in PLLCTL to 0 (allows PLEN bit to take effect).
  - (b) For PLL0 only, select the clock source by programming the EXTCLKSRC bit in PLLCTL.
  - (c) Clear the PLEN bit in PLLCTL to 0 (PLL in bypass mode).
  - (d) Wait for 4 OSCIN cycles to ensure that the PLLC has switched to bypass mode.
3. Clear the PLLRST bit in PLLCTL to 0 (resets PLL).
4. Clear the PLLPWDN bit in PLLCTL to 0 (brings PLL out of power-down mode).
5. Program the desired multiplier value in PLLM. Program the POSTDIV, as needed.
6. If desired, program PLLDIV $n$  registers to change the SYSCLK $n$  divide values:
  - (a) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (indicates that no operation is currently in progress).
  - (b) Program the RATIO field in PLLDIV $n$ .
  - (c) Set the GOSET bit in PLLCMD to 1 (initiates a new divider transition).
  - (d) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).
7. Set the PLLRST bit in PLLCTL to 1 (brings PLL out of reset).
8. Wait for the PLL to lock. See the device-specific data manual for PLL lock time.
9. Set the PLEN bit in PLLCTL to 1 (removes PLL from bypass mode).

### 7.2.2.3 Changing PLL Multiplier

If the PLL is not powered down (PLLPWDN bit in PLLCTL is cleared to 0), perform the following procedure to change the PLL multiplier:

1. Switch the PLL to bypass mode:
  - (a) Clear the PLENSRC bit in PLLCTL to 0 (allows PLEN bit to take effect).
  - (b) For PLL0 only, select the clock source by programming the EXTCLKSRC bit in PLLCTL.
  - (c) Clear the PLEN bit in PLLCTL to 0 (PLL in bypass mode).
  - (d) Wait for 4 OSCIN cycles to ensure that the PLLC has switched to bypass mode.
2. Clear the PLLRST bit in PLLCTL to 0 (resets PLL).
3. Program the desired multiplier value in PLLM. Program the POSTDIV, as needed.
4. If desired, program PLLDIV $n$  registers to change the SYSCLK $n$  divide values:
  - (a) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (indicates that no operation is currently in progress).
  - (b) Program the RATIO field in PLLDIV $n$ .
  - (c) Set the GOSET bit in PLLCMD to 1 (initiates a new divider transition).
  - (d) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).
5. Set the PLLRST bit in PLLCTL to 1 (brings PLL out of reset).
6. Wait for the PLL to lock. See the device-specific data manual for PLL lock time.
7. Set the PLEN bit in PLLCTL to 1 (removes PLL from bypass mode).

### 7.2.2.4 Changing SYSCLK Dividers

If the PLL is already operating at the desired multiplier mode, perform the following procedure to change the SYSCLK divider values:

1. Wait for the GOSTAT bit in PLLSTAT to clear to 0 (indicates that no operation is currently in progress).
2. Program the RATIO field in PLLDIV $n$ .
3. Set the GOSET bit in PLLCMD to 1 (initiates a new divider transition).
4. Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).

## 7.3 PLLC Registers

Table 7-2 lists the memory-mapped registers for the PLLC0 and Table 7-3 lists the memory-mapped registers for the PLLC1.

**Table 7-2. PLL Controller 0 (PLLC0) Registers**

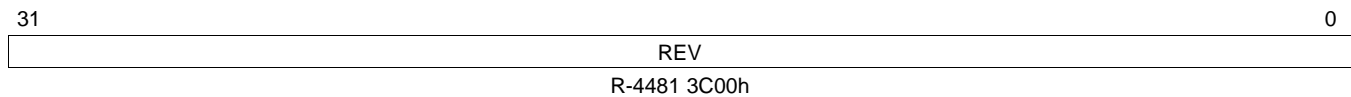
Address	Acronym	Register Description	Section
01C1 1000h	REVID	PLLC0 Revision Identification Register	<a href="#">Section 7.3.1</a>
01C1 10E4h	RSTYPE	PLLC0 Reset Type Status Register	<a href="#">Section 7.3.3</a>
01C1 10E8h	RSCTRL	PLLC0 Reset Control Register	<a href="#">Section 7.3.4</a>
01C1 1100h	PLLCTL	PLLC0 Control Register	<a href="#">Section 7.3.5</a>
01C1 1104h	OCSEL	PLLC0 OBSCLK Select Register	<a href="#">Section 7.3.7</a>
01C1 1110h	PLLM	PLLC0 PLL Multiplier Control Register	<a href="#">Section 7.3.9</a>
01C1 1114h	PREDIV	PLLC0 Pre-Divider Control Register	<a href="#">Section 7.3.10</a>
01C1 1118h	PLLDIV1	PLLC0 Divider 1 Register	<a href="#">Section 7.3.11</a>
01C1 111Ch	PLLDIV2	PLLC0 Divider 2 Register	<a href="#">Section 7.3.13</a>
01C1 1120h	PLLDIV3	PLLC0 Divider 3 Register	<a href="#">Section 7.3.15</a>
01C1 1124h	OSCDIV	PLLC0 Oscillator Divider 1 Register	<a href="#">Section 7.3.21</a>
01C1 1128h	POSTDIV	PLLC0 PLL Post-Divider Control Register	<a href="#">Section 7.3.23</a>
01C1 1138h	PLLCMD	PLLC0 PLL Controller Command Register	<a href="#">Section 7.3.24</a>
01C1 113Ch	PLLSTAT	PLLC0 PLL Controller Status Register	<a href="#">Section 7.3.25</a>
01C1 1140h	ALNCTL	PLLC0 Clock Align Control Register	<a href="#">Section 7.3.26</a>
01C1 1144h	DCHANGE	PLLC0 PLLDIV Ratio Change Status Register	<a href="#">Section 7.3.28</a>
01C1 1148h	CKEN	PLLC0 Clock Enable Control Register	<a href="#">Section 7.3.30</a>
01C1 114Ch	CKSTAT	PLLC0 Clock Status Register	<a href="#">Section 7.3.32</a>
01C1 1150h	SYSTAT	PLLC0 SYSCLK Status Register	<a href="#">Section 7.3.34</a>
01C1 1160h	PLLDIV4	PLLC0 Divider 4 Register	<a href="#">Section 7.3.17</a>
01C1 1164h	PLLDIV5	PLLC0 Divider 5 Register	<a href="#">Section 7.3.18</a>
01C1 1168h	PLLDIV6	PLLC0 Divider 6 Register	<a href="#">Section 7.3.19</a>
01C1 116Ch	PLLDIV7	PLLC0 Divider 7 Register	<a href="#">Section 7.3.20</a>
01C1 11F0h	EMUCNT0	PLLC0 Emulation Performance Counter 0 Register	<a href="#">Section 7.3.36</a>
01C1 11F4h	EMUCNT1	PLLC0 Emulation Performance Counter 1 Register	<a href="#">Section 7.3.37</a>

**Table 7-3. PLL Controller 1 (PLL<sub>C</sub>1) Registers**

Address	Acronym	Register Description	Section
01E1 A00h	REVID	PLL <sub>C</sub> 1 Revision Identification Register	<a href="#">Section 7.3.2</a>
01E1 A100h	PLLCTL	PLL <sub>C</sub> 1 Control Register	<a href="#">Section 7.3.6</a>
01E1 A104h	OCSEL	PLL <sub>C</sub> 1 OBSCLK Select Register	<a href="#">Section 7.3.8</a>
01E1 A110h	PLLM	PLL <sub>C</sub> 1 PLL Multiplier Control Register	<a href="#">Section 7.3.9</a>
01E1 A118h	PLLDIV1	PLL <sub>C</sub> 1 Divider 1 Register	<a href="#">Section 7.3.12</a>
01E1 A11Ch	PLLDIV2	PLL <sub>C</sub> 1 Divider 2 Register	<a href="#">Section 7.3.14</a>
01E1 A120h	PLLDIV3	PLL <sub>C</sub> 1 Divider 3 Register	<a href="#">Section 7.3.16</a>
01E1 A124h	OSCDIV	PLL <sub>C</sub> 1 Oscillator Divider 1 Register	<a href="#">Section 7.3.22</a>
01E1 A128h	POSTDIV	PLL <sub>C</sub> 1 PLL Post-Divider Control Register	<a href="#">Section 7.3.23</a>
01E1 A138h	PLLCMD	PLL <sub>C</sub> 1 PLL Controller Command Register	<a href="#">Section 7.3.24</a>
01E1 A13Ch	PLLSTAT	PLL <sub>C</sub> 1 PLL Controller Status Register	<a href="#">Section 7.3.25</a>
01E1 A140h	ALNCTL	PLL <sub>C</sub> 1 Clock Align Control Register	<a href="#">Section 7.3.27</a>
01E1 A144h	DCHANGE	PLL <sub>C</sub> 1 PLLDIV Ratio Change Status Register	<a href="#">Section 7.3.29</a>
01E1 A148h	CKEN	PLL <sub>C</sub> 1 Clock Enable Control Register	<a href="#">Section 7.3.31</a>
01E1 A14Ch	CKSTAT	PLL <sub>C</sub> 1 Clock Status Register	<a href="#">Section 7.3.33</a>
01E1 A150h	SYSTAT	PLL <sub>C</sub> 1 SYSCLK Status Register	<a href="#">Section 7.3.35</a>
01E1 A1F0h	EMUCNT0	PLL <sub>C</sub> 1 Emulation Performance Counter 0 Register	<a href="#">Section 7.3.36</a>
01E1 A1F4h	EMUCNT1	PLL <sub>C</sub> 1 Emulation Performance Counter 1 Register	<a href="#">Section 7.3.37</a>

### 7.3.1 PLL<sub>C</sub>0 Revision Identification Register (REVID)

The PLL<sub>C</sub>0 revision identification register (REVID) is shown in [Figure 7-2](#) and described in [Table 7-4](#).

**Figure 7-2. PLL<sub>C</sub>0 Revision Identification Register (REVID)**

LEGEND: R = Read only; -n = value after reset

**Table 7-4. PLL<sub>C</sub>0 Revision Identification Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4481 3C00h	Peripheral revision ID for PLL <sub>C</sub> 0.



### 7.3.2 PLLC1 Revision Identification Register (REVID)

The PLLC1 revision identification register (REVID) is shown in [Figure 7-3](#) and described in [Table 7-5](#).

**Figure 7-3. PLLC1 Revision Identification Register (REVID)**



LEGEND: R = Read only; -n = value after reset

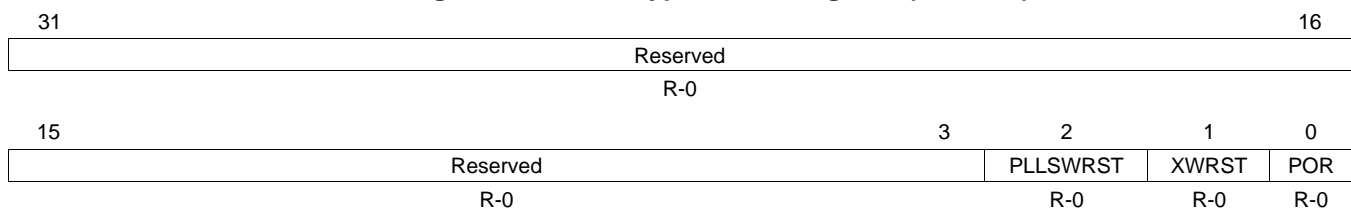
**Table 7-5. PLLC1 Revision Identification Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4481 4400h	Peripheral revision ID for PLLC1.

### 7.3.3 Reset Type Status Register (RSTYPE)

The reset type status register (RSTYPE) latches the cause of the last reset. If multiple reset sources are asserted simultaneously, RSTYPE records the reset source that deasserts last. If multiple reset sources are asserted and deasserted simultaneously, RSTYPE latches the highest priority reset source. RSTYPE is shown in [Figure 7-4](#) and described in [Table 7-6](#).

**Figure 7-4. Reset Type Status Register (RSTYPE)**



LEGEND: R = Read only; -n = value after reset

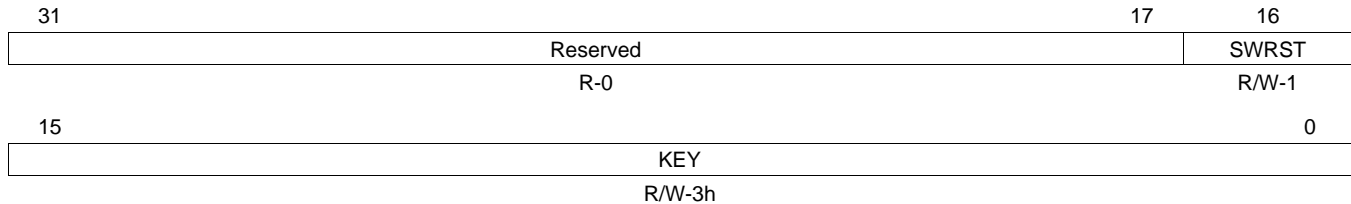
**Table 7-6. Reset Type Status Register (RSTYPE) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	PLLSWRST	0	PLL software reset.
		0	PLL soft reset was not the last reset to occur.
		1	PLL soft was the last reset to occur.
1	XWRST	0	External warm reset.
		0	External warm reset was not the last reset to occur.
		1	External warm reset was the last reset to occur.
0	POR	0	Power on reset.
		0	Power On Reset (POR) was not the last reset to occur.
		1	Power On Reset (POR) was the last reset to occur.

### 7.3.4 PLLC0 Reset Control Register (RSCTRL)

The reset control register (RSCTRL) allows the device to perform a software-initiated reset. Before writing to the SWRST bit, the register must be unlocked by writing the key value of 5A69h to the KEY bit field. The KEY bit field reads back as Ch when the register is unlocked; any other key value is invalid and indicates that the register is locked. Any write to the register following a successful unlock relocks the register. RSCTRL is shown in [Figure 7-5](#) and described in [Table 7-7](#).

**Figure 7-5. Reset Control Register (RSCTRL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-7. Reset Control Register (RSCTRL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	SWRST	0	PLL software reset. Register must be unlocked before writing to this bit. Writes are possible only when qualified with a valid key. In software reset
		1	Not in software reset
15-0	KEY	0-FFFFh	RSCTRL unlock key. Key used to enable writes to RSCTRL.
		3h	Register is locked when read value is 3h.
		Ch	Register is unlocked when read value is Ch.
		5A69h	RSCTRL unlock key

### 7.3.5 PLLC0 Control Register (PLLCTL)

The PLLC0 control register (PLLCTL) is shown in [Figure 7-6](#) and described in [Table 7-8](#).

**Figure 7-6. PLLC0 Control Register (PLLCTL)**

31	Reserved						16
R-0							
15	Reserved				10	9	8
R-0					R/W-0	R/W-0	
7	6	5	4	3	2	1	0
Reserved	PLEN	PWRDN	RST	Reserved	SRC	MODE	LEN
R-1	R/W-1	R/W-1	R/W-0	R-0	R/W-1	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

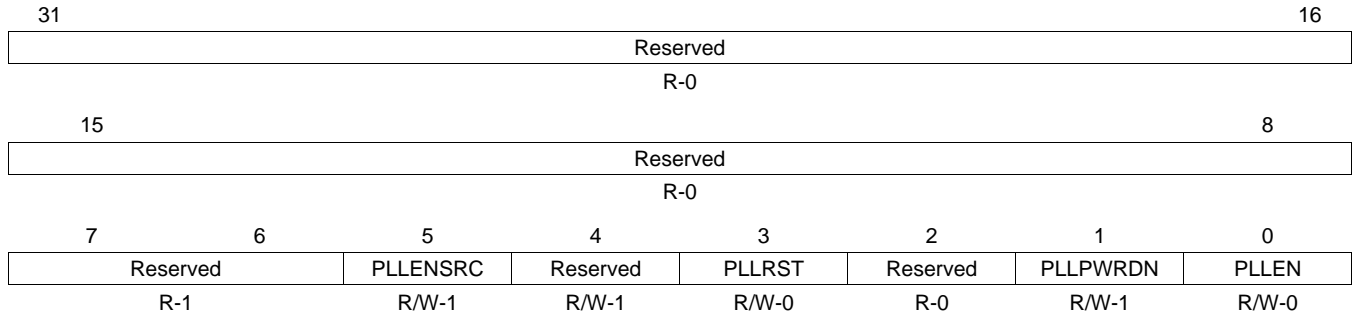
**Table 7-8. PLLC0 Control Register (PLLCTL) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	EXTCLKSRC	0	External clock source selection. Use OSCIN for the PLL bypass clock.
		1	Use PLL1_SYCLK3 for the PLL bypass clock.
8	CLKMODE	0	Reference clock selection. Internal oscillator (crystal)
		1	Square wave
7-6	Reserved	1	Reserved
5	PLENSRC	0	This bit must be cleared before the PLEN bit will have any effect.
4	Reserved	1	Reserved. Write the default value when modifying this register.
3	PLLST	0	PLL0 reset. PLL0 reset is asserted.
		1	PLL0 reset is not asserted.
2	Reserved	0	Reserved
1	PLLPWRDN	0	PLL0 power-down. PLL0 is operating.
		1	PLL0 is powered-down.
0	PLEN	0	PLL0 mode enables. PLL0 is in bypass mode.
		1	PLL0 mode is enabled, not bypassed.

### 7.3.6 PLLC1 Control Register (PLLCTL)

The PLLC1 control register (PLLCTL) is shown in [Figure 7-7](#) and described in [Table 7-9](#).

**Figure 7-7. PLLC1 Control Register (PLLCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

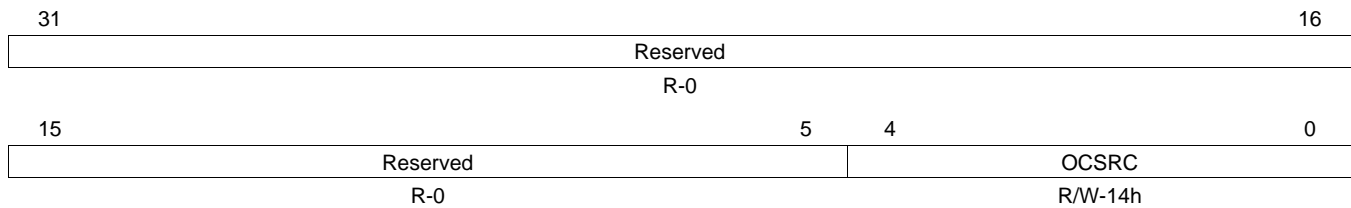
**Table 7-9. PLLC1 Control Register (PLLCTL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	Reserved	1	Reserved
5	PPLENSRC	0	This bit must be cleared before the PLL1LEN bit will have any effect.
4	Reserved	1	Reserved. Write the default value when modifying this register.
3	PLL1RST	0	PLL1 reset is asserted.
		1	PLL1 reset is not asserted.
2	Reserved	0	Reserved
1	PLL1PWRDN	0	PLL1 is operating.
		1	PLL1 is powered-down.
0	PLL1LEN	0	PLL1 mode enables.
		0	PLL1 is in bypass mode.
		1	PLL1 mode is enabled, not bypassed.

### 7.3.7 PLLC0 OBSCLK Select Register (OCSEL)

The PLLC0 OBSCLK select register (OCSEL) controls which clock is output on the CLKOUT pin so that it may be used for test and debug purposes (in addition to its normal function of being a direct input clock divider). The OCSEL is shown in Figure 7-8 and described in Table 7-10.

**Figure 7-8. PLLC0 OBSCLK Select Register (OCSEL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

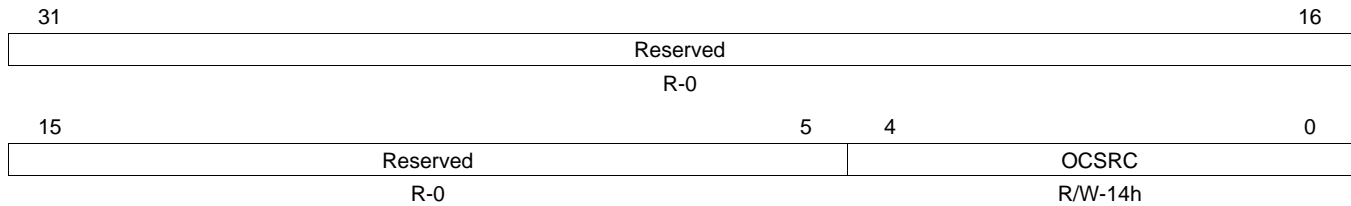
**Table 7-10. PLLC0 OBSCLK Select Register (OCSEL) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	OCSRC	0-1Fh	PLLC0 OBSCLK source. Output on CLKOUT pin.
		0-13h	Reserved
		14h	OSCIN
		15h-16h	Reserved
		17h	PLL0_SYSCLK1
		18h	PLL0_SYSCLK2
		19h	PLL0_SYSCLK3
		1Ah	PLL0_SYSCLK4
		1Bh	PLL0_SYSCLK5
		1Ch	PLL0_SYSCLK6
		1Dh	PLL0_SYSCLK7
		1Eh	PLL1 OBSCLK
		1Fh	Disabled

### 7.3.8 PLLC1 OBSCLK Select Register (OCSEL)

The PLLC1 OBSCLK select register (OCSEL) controls which clock is output on PLLC1 OBSCLK so that it may be used for test and debug purposes (in addition to its normal function of being a direct input clock divider). The OCSEL is shown in [Figure 7-9](#) and described in [Table 7-11](#).

**Figure 7-9. PLLC1 OBSCLK Select Register (OCSEL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

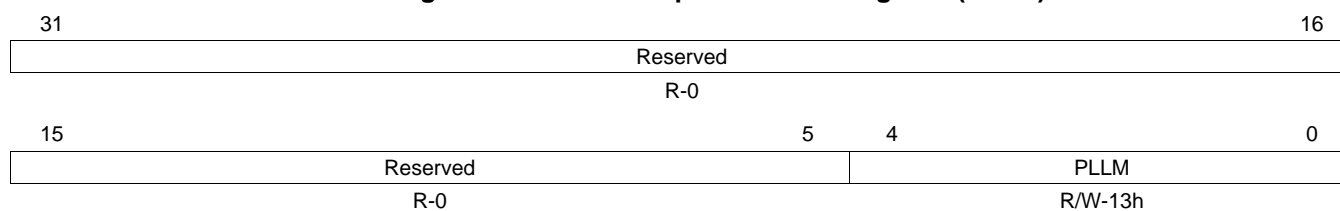
**Table 7-11. PLLC1 OBSCLK Select Register (OCSEL) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	OCSRC	0-1Fh	PLLC1 OBSCLK source.
		0-13h	Reserved
		14h	OSCIN
		15h-16h	Reserved
		17h	PLL1_SYSCLK1
		18h	PLL1_SYSCLK2
		19h	PLL1_SYSCLK3
		1A-1Fh	Reserved

### 7.3.9 PLL Multiplier Control Register (PLLM)

The PLL multiplier control register (PLLM) is shown in [Figure 7-10](#) and described in [Table 7-12](#).

**Figure 7-10. PLL Multiplier Control Register (PLLM)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

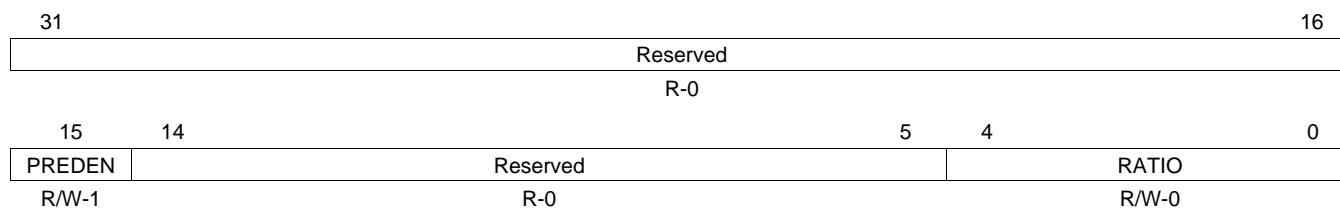
**Table 7-12. PLL Multiplier Control Register (PLLM) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PLLM	0-1Fh	PLL multiplier select. Multiplier Value = PLLM + 1. The valid range of multiplier values for a given OSCIN is defined by the minimum and maximum frequency limits on the PLL VCO frequency. See the device-specific data manual for PLL VCO frequency specification limits.

### 7.3.10 PLLC0 Pre-Divider Control Register (PREDIV)

The PLLC0 pre-divider control register (PREDIV) is shown in [Figure 7-11](#) and described in [Table 7-13](#).

**Figure 7-11. PLLC0 Pre-Divider Control Register (PREDIV)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

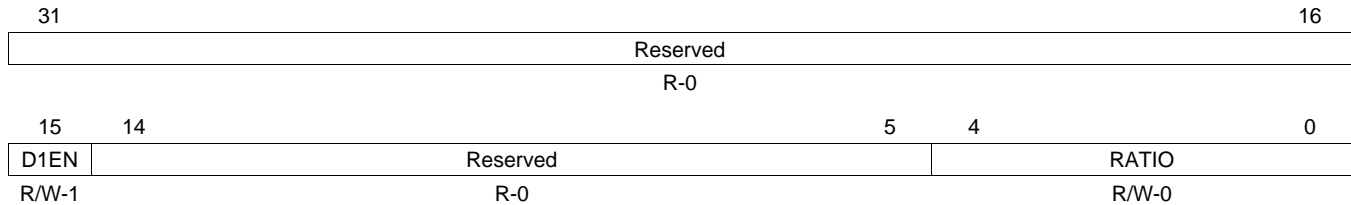
**Table 7-13. PLLC0 Pre-Divider Control Register (PREDIV) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
15	PREDEN	0 1	PLLC0 pre-divider enable. 0 PLLC0 pre-divider is disabled. Clock output from the PREDIV stage is disabled. 1 PLLC0 pre-divider is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL pre-divide by 1).

### 7.3.11 PLLC0 Divider 1 Register (PLLDIV1)

The PLLC0 divider 1 register (PLLDIV1) controls the divider for PLL0\_SYSCLK1. PLLDIV1 is shown in [Figure 7-12](#) and described in [Table 7-14](#).

**Figure 7-12. PLLC0 Divider 1 Register (PLLDIV1)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

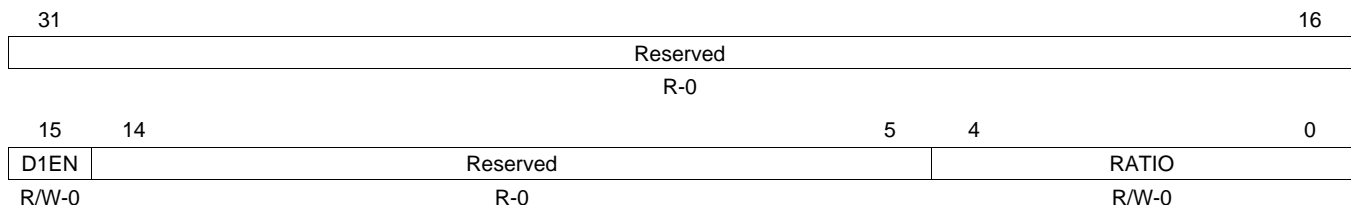
**Table 7-14. PLLC0 Divider 1 Register (PLLDIV1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN	0	Divider 1 enable. Divider 1 is disabled.
		1	Divider 1 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).

### 7.3.12 PLLC1 Divider 1 Register (PLLDIV1)

The PLLC1 divider 1 register (PLLDIV1) controls the divider for PLL1\_SYSCLK1. PLLDIV1 is shown in [Figure 7-13](#) and described in [Table 7-15](#).

**Figure 7-13. PLLC1 Divider 1 Register (PLLDIV1)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-15. PLLC1 Divider 1 Register (PLLDIV1) Field Descriptions**

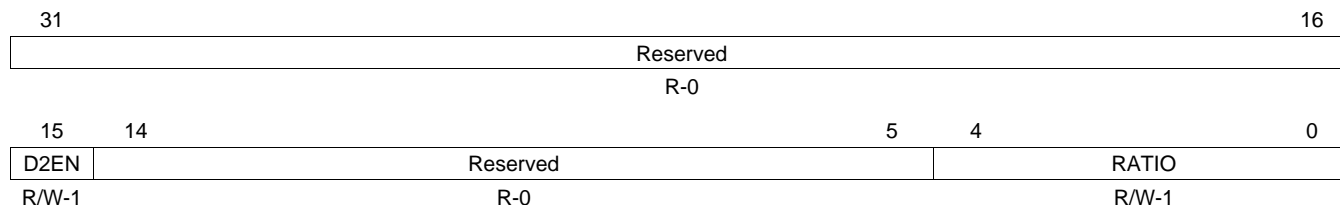
Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN	0	Divider 1 enable. Divider 1 is disabled.
		1	Divider 1 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).



### 7.3.13 PLLC0 Divider 2 Register (PLLDIV2)

The PLLC0 divider 2 register (PLLDIV2) controls the divider for PLL0\_SYSCLK2. PLLDIV2 is shown in Figure 7-14 and described in Table 7-16.

**Figure 7-14. PLLC0 Divider 2 Register (PLLDIV2)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

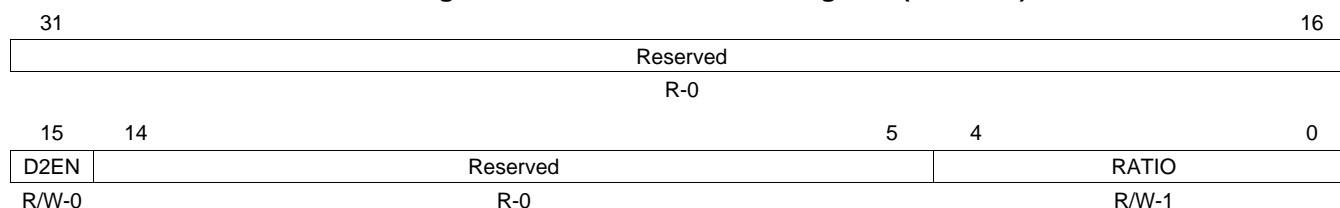
**Table 7-16. PLLC0 Divider 2 Register (PLLDIV2) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN	0 1	Divider 2 enable. Divider 2 is disabled. Divider 2 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL divide by 2).

### 7.3.14 PLLC1 Divider 2 Register (PLLDIV2)

The PLLC1 divider 2 register (PLLDIV2) controls the divider for PLL1\_SYSCLK2. PLLDIV2 is shown in Figure 7-15 and described in Table 7-17.

**Figure 7-15. PLLC1 Divider 2 Register (PLLDIV2)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-17. PLLC1 Divider 2 Register (PLLDIV2) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN	0 1	Divider 2 enable. Divider 2 is disabled. Divider 2 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL divide by 2).

### 7.3.15 PLLC0 Divider 3 Register (PLLDIV3)

The PLLC0 divider 3 register (PLLDIV3) controls the divider for PLL0\_SYSCLK3. PLLDIV3 is shown in [Figure 7-16](#) and described in [Table 7-18](#).

**Figure 7-16. PLLC0 Divider 3 Register (PLLDIV3)**

31	Reserved										16	
R-0												
15	14	Reserved					5	4	RATIO			0
D3EN		Reserved					RATIO					
R/W-1		R-0					R/W-2h					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-18. PLLC0 Divider 3 Register (PLLDIV3) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN	0 1	Divider 3 enable. Divider 3 is disabled. Divider 3 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 2h (PLL divide by 3).

### 7.3.16 PLLC1 Divider 3 Register (PLLDIV3)

The PLLC1 divider 3 register (PLLDIV3) controls the divider for PLL1\_SYSCLK3. PLLDIV3 is shown in [Figure 7-17](#) and described in [Table 7-19](#).

**Figure 7-17. PLLC1 Divider 3 Register (PLLDIV3)**

31	Reserved										16	
R-0												
15	14	Reserved					5	4	RATIO			0
D3EN		Reserved					RATIO					
R/W-0		R-0					R/W-2h					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

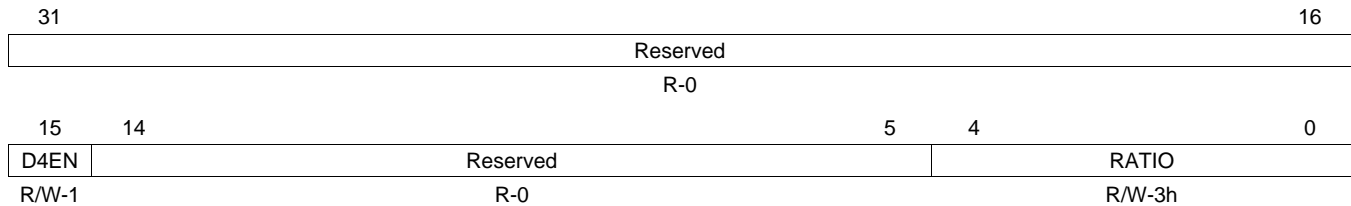
**Table 7-19. PLLC1 Divider 3 Register (PLLDIV3) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN	0 1	Divider 3 enable. Divider 3 is disabled. Divider 3 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 2h (PLL divide by 3).

### 7.3.17 PLLC0 Divider 4 Register (PLLDIV4)

The PLLC0 divider 4 register (PLLDIV4) controls the divider for PLL0\_SYSCLK4. PLLDIV4 is shown in [Figure 7-18](#) and described in [Table 7-20](#).

**Figure 7-18. PLLC0 Divider 4 Register (PLLDIV4)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

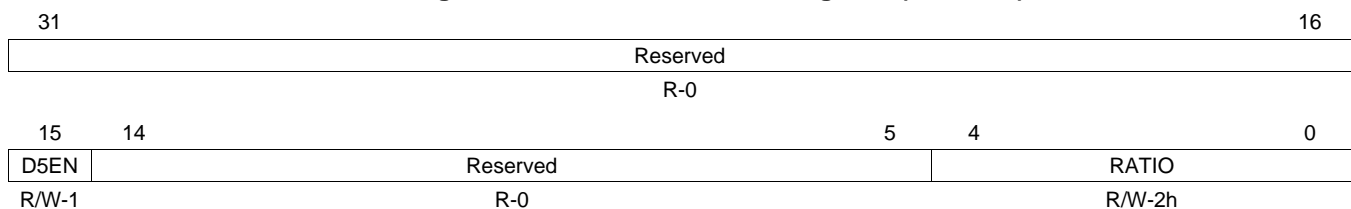
**Table 7-20. PLLC0 Divider 4 Register (PLLDIV4) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D4EN	0	Divider 4 enable.
		0	Divider 4 is disabled.
		1	Divider 4 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults 3 (PLL divide by 4).

### 7.3.18 PLLC0 Divider 5 Register (PLLDIV5)

The PLLC0 divider 5 register (PLLDIV5) controls the divider for PLL0\_SYSCLK5. PLLDIV5 is shown in [Figure 7-19](#) and described in [Table 7-21](#).

**Figure 7-19. PLLC0 Divider 5 Register (PLLDIV5)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

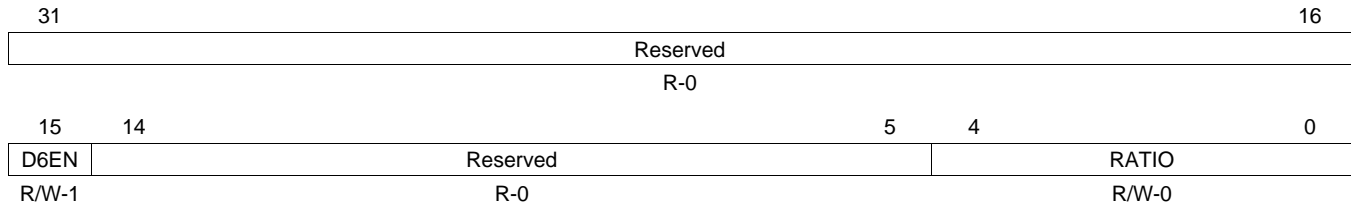
**Table 7-21. PLLC0 Divider 5 Register (PLLDIV5) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D5EN	0	Divider 5 enable.
		0	Divider 5 is disabled.
		1	Divider 5 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults 2 (PLL divide by 3).

### 7.3.19 PLLC0 Divider 6 Register (PLLDIV6)

The PLLC0 divider 6 register (PLLDIV6) controls the divider for PLL0\_SYSCLK6. PLLDIV6 is shown in [Figure 7-20](#) and described in [Table 7-22](#).

**Figure 7-20. PLLC0 Divider 6 Register (PLLDIV6)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

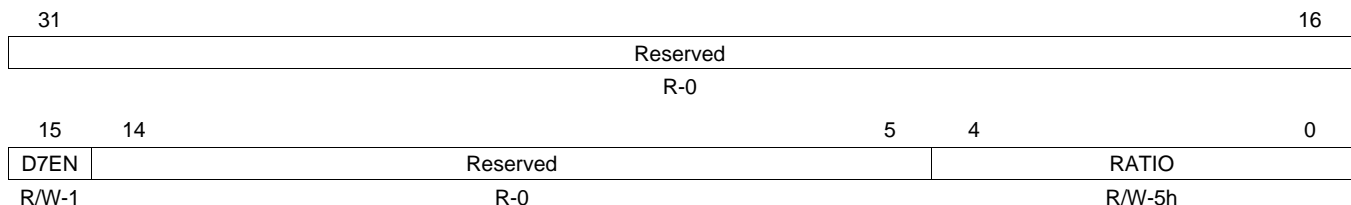
**Table 7-22. PLLC0 Divider 6 Register (PLLDIV6) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D6EN	0 1	Divider 6 enable. Divider 6 is disabled. Divider 6 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).

### 7.3.20 PLLC0 Divider 7 Register (PLLDIV7)

The PLLC0 divider 7 register (PLLDIV7) controls the divider for PLL0\_SYSCLK7. PLLDIV7 is shown in [Figure 7-21](#) and described in [Table 7-23](#).

**Figure 7-21. PLLC0 Divider 7 Register (PLLDIV7)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-23. PLLC0 Divider 7 Register (PLLDIV7) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D7EN	0 1	Divider 7 enable. Divider 7 is disabled. Divider 7 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 5 (PLL divide by 6).

### 7.3.21 PLLC0 Oscillator Divider 1 Register (OSCDIV)

The PLLC0 oscillator divider 1 register (OSCDIV) controls the divider for PLLC0 OBSCLK, dividing down the clock selected as the PLLC0 OBSCLK source. The PLLC0 OBSCLK is connected to the CLKOUT pin. The OSCDIV is shown in [Figure 7-22](#) and described in [Table 7-24](#).

**Figure 7-22. PLLC0 Oscillator Divider 1 Register (OSCDIV)**

31	Reserved										16	
R-0												
15	14						5	4				0
OD1EN	Reserved						RATIO					
R/W-1	R-0						R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-24. PLLC0 Oscillator Divider 1 Register (OSCDIV) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	OD1EN	0	Oscillator divider 1 is disabled.
		1	Oscillator divider 1 is enabled. For PLLC0 OBSCLK to toggle, both the OD1EN bit and the OBSEN bit in the PLLC0 clock enable control register (CKEN) must be set to 1.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider value = RATIO + 1. For example, RATIO = 0 means divide by 1.

### 7.3.22 PLLC1 Oscillator Divider 1 Register (OSCDIV)

The PLLC1 oscillator divider 1 register (OSCDIV) controls the divider for PLLC1 OBSCLK, dividing down the clock selected as the PLLC1 OBSCLK source. The PLLC1 OBSCLK signal may be selected as the output on the CLKOUT pin. The OSCDIV is shown in [Figure 7-23](#) and described in [Table 7-25](#).

**Figure 7-23. PLLC1 Oscillator Divider 1 Register (OSCDIV)**

31	Reserved										16	
R-0												
15	14						5	4				0
OD1EN	Reserved						RATIO					
R/W-1	R-0						R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

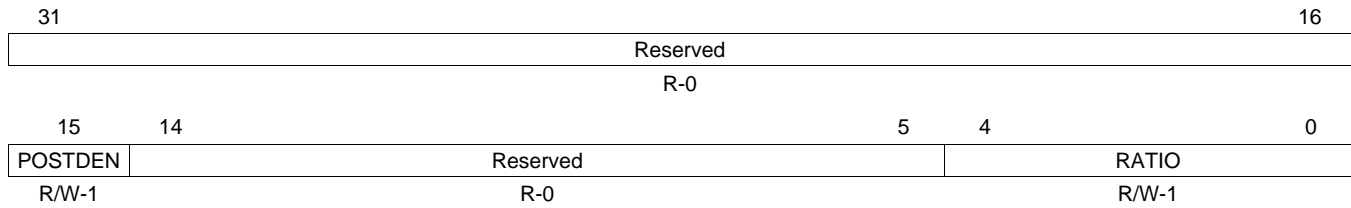
**Table 7-25. PLLC1 Oscillator Divider 1 Register (OSCDIV) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	OD1EN	0	Oscillator divider 1 is disabled.
		1	Oscillator divider 1 is enabled. For PLLC1 OBSCLK to toggle, both the OD1EN bit and the OBSEN bit in the PLLC1 clock enable control register (CKEN) must be set to 1.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider value = RATIO + 1. For example, RATIO = 0 means divide by 1.

### 7.3.23 PLL Post-Divider Control Register (POSTDIV)

The PLL post-divider control register (POSTDIV) is shown in [Figure 7-24](#) and described in [Table 7-26](#).

**Figure 7-24. PLL Post-Divider Control Register (POSTDIV)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

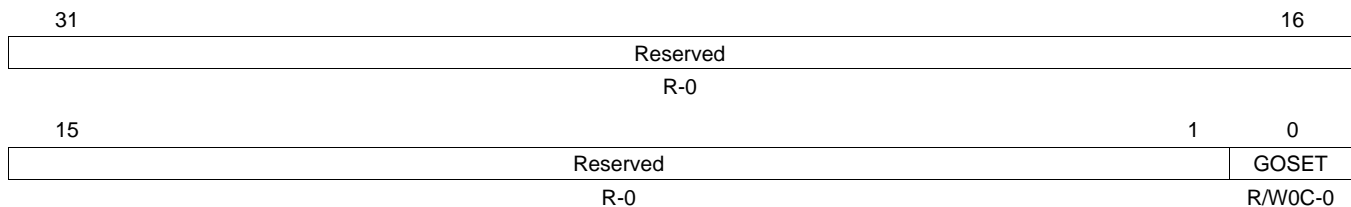
**Table 7-26. PLL Post-Divider Control Register (POSTDIV) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	POSTDEN	0	Post-divider enable. Post-divider is disabled.
		1	Post-divider is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL post-divide by 2).

### 7.3.24 PLL Controller Command Register (PLLCMD)

The PLL controller command register (PLLCMD) contains the command bit for phase alignment. A write of 1 initiates the command; a write of 0 clears the bit, but has no effect. PLLCMD is shown in [Figure 7-25](#) and described in [Table 7-27](#).

**Figure 7-25. PLL Controller Command Register (PLLCMD)**



LEGEND: R/W = Read/Write; R = Read only; W0C = Write 0 to clear bit; -n = value after reset

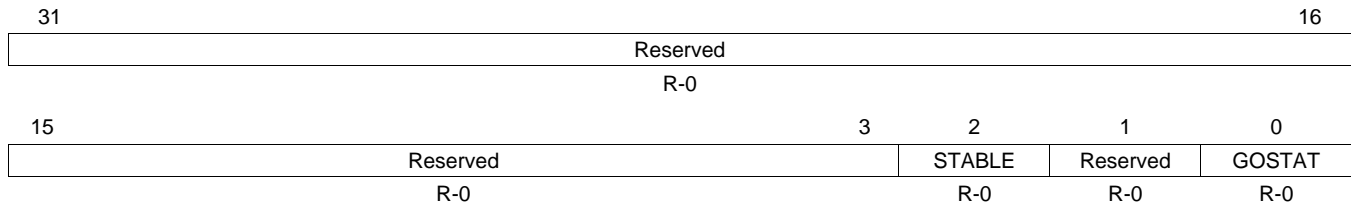
**Table 7-27. PLL Controller Command Register (PLLCMD) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GOSET	0	GO bit for phase alignment. Clear bit (no effect)
		1	Phase alignment

### 7.3.25 PLL Controller Status Register (PLLSTAT)

The PLL controller status register (PLLSTAT) is shown in [Figure 7-26](#) and described in [Table 7-28](#).

**Figure 7-26. PLL Controller Status Register (PLLSTAT)**



LEGEND: R = Read only; -n = value after reset

**Table 7-28. PLL Controller Status Register (PLLSTAT) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	STABLE	0	OSC counter done, oscillator assumed to be stable. By the time the device comes out of reset, this bit should become 1.
		1	No Yes
1	Reserved	0	Reserved
0	GOSTAT	0	Status of GO operation. If 1, indicates GO operation is in progress.
		1	GO operation is not in progress. GO operation is in progress.

### 7.3.26 PLLC0 Clock Align Control Register (ALNCTL)

The PLLC0 clock align control register (ALNCTL) indicates which PLL0\_SYSCLK $n$  needs to be aligned for proper device operation. ALNCTL is shown in [Figure 7-27](#) and described in [Table 7-29](#).

**Figure 7-27. PLLC0 Clock Align Control Register (ALNCTL)**

31	Reserved								16					
R-0														
15	Reserved						7	6	5	4	3	2	1	0
R-3h							ALN7	ALN6	ALN5	ALN4	ALN3	ALN2	ALN1	
							R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	

LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 7-29. PLLC0 Clock Align Control Register (ALNCTL) Field Descriptions**

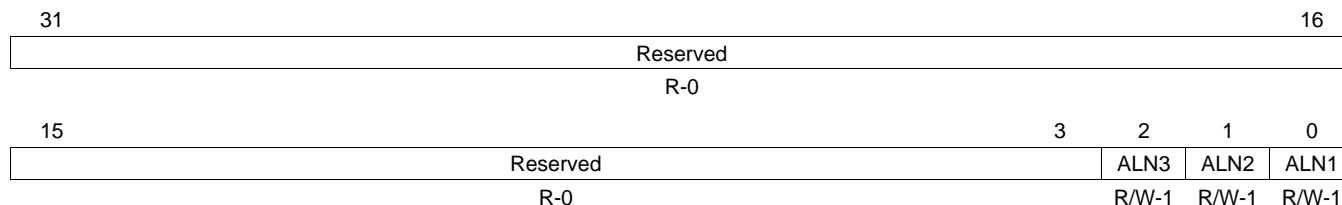
Bit	Field	Value	Description
31-7	Reserved	3h	Reserved
6	ALN7	0 1	PLL0_SYSCLK7 needs to be aligned to others selected in this register. No Yes
5	ALN6	0 1	PLL0_SYSCLK6 needs to be aligned to others selected in this register. No Yes
4	ALN5	0 1	PLL0_SYSCLK5 needs to be aligned to others selected in this register. No Yes
3	ALN4	0 1	PLL0_SYSCLK4 needs to be aligned to others selected in this register. No Yes
2	ALN3	0 1	PLL0_SYSCLK3 needs to be aligned to others selected in this register. No Yes
1	ALN2	0 1	PLL0_SYSCLK2 needs to be aligned to others selected in this register. No Yes
0	ALN1	0 1	PLL0_SYSCLK1 needs to be aligned to others selected in this register. No Yes



### 7.3.27 PLLC1 Clock Align Control Register (ALNCTL)

The PLLC1 clock align control register (ALNCTL) indicates which PLL1\_SYSCLK $n$  needs to be aligned for proper device operation. ALNCTL is shown in [Figure 7-28](#) and described in [Table 7-30](#).

**Figure 7-28. PLLC1 Clock Align Control Register (ALNCTL)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

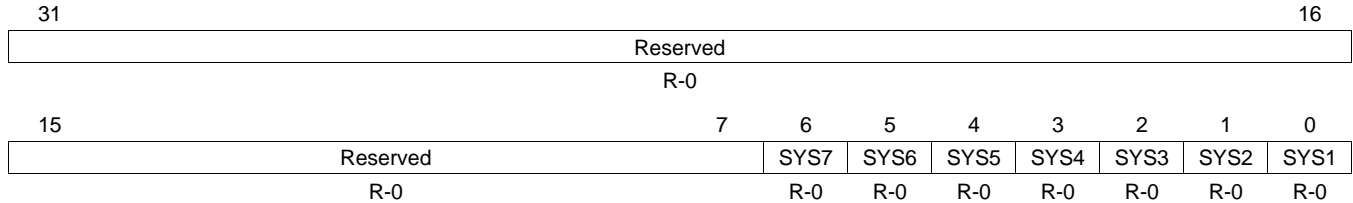
**Table 7-30. PLLC1 Clock Align Control Register (ALNCTL) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	ALN3	0 1	PLL1_SYSCLK3 needs to be aligned to others selected in this register. No Yes
1	ALN2	0 1	PLL1_SYSCLK2 needs to be aligned to others selected in this register. No Yes
0	ALN1	0 1	PLL1_SYSCLK1 needs to be aligned to others selected in this register. No Yes

### 7.3.28 PLLC0 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLC0 PLLDIV ratio change status register (DCHANGE) indicates if the PLL0\_SYSCCLK $n$  divide ratio has been modified. DCHANGE is shown in [Figure 7-29](#) and described in [Table 7-31](#).

**Figure 7-29. PLLC0 PLLDIV Ratio Change Status Register (DCHANGE)**



LEGEND: R = Read only; - $n$  = value after reset

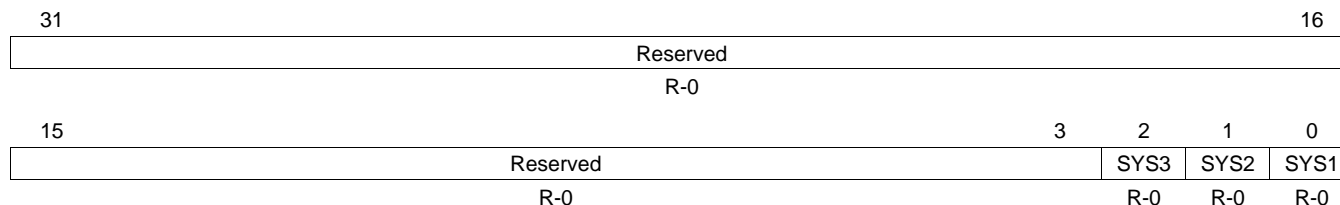
**Table 7-31. PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions**

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	SYS7	0	PLL0_SYSCCLK7 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
5	SYS6	0	PLL0_SYSCCLK6 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
4	SYS5	0	PLL0_SYSCCLK5 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
3	SYS4	0	PLL0_SYSCCLK4 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
2	SYS3	0	PLL0_SYSCCLK3 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
1	SYS2	0	PLL0_SYSCCLK2 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
0	SYS1	0	PLL0_SYSCCLK1 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.

### 7.3.29 PLLC1 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLC1 PLLDIV ratio change status register (DCHANGE) indicates if the PLL1\_SYSCCLK $n$  divide ratio has been modified. DCHANGE is shown in [Figure 7-30](#) and described in [Table 7-32](#).

**Figure 7-30. PLLC1 PLLDIV Ratio Change Status Register (DCHANGE)**



LEGEND: R = Read only; - $n$  = value after reset

**Table 7-32. PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	SYS3	0	PLL1_SYSCCLK3 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
1	SYS2	0	PLL1_SYSCCLK2 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.
0	SYS1	0	PLL1_SYSCCLK1 divide ratio is modified. Ratio is not modified.
		1	Ratio is modified.

### 7.3.30 PLLC0 Clock Enable Control Register (CKEN)

The PLLC0 clock enable control register (CKEN) controls the PLLC0 OBSCLK and AUXCLK clock. CKEN is shown in [Figure 7-31](#) and described in [Table 7-33](#).

**Figure 7-31. PLLC0 Clock Enable Control Register (CKEN)**

31	Reserved			16	
R-0					
15	Reserved		2	1	0
R-0			R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-33. PLLC0 Clock Enable Control Register (CKEN) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN	0	OBSCLK enable. Actual PLLC0 OBSCLK status is shown in the PLLC0 clock status register (CKSTAT). PLL0 OBSCLK is disabled.
		1	PLL0 OBSCLK is enabled. For PLLC0 OBSCLK to toggle, both the OBSEN bit and the OD1EN bit in the PLLC0 oscillator divider 1 register (OSCDIV) must be set to 1.
0	AUXEN	0	AUXCLK enable. Actual PLLC0 AUXCLK status is shown in the PLLC0 clock status register (CKSTAT). PLL0 AUXCLK is disabled.
		1	PLL0 AUXCLK is enabled.

### 7.3.31 PLLC1 Clock Enable Control Register (CKEN)

The PLLC1 clock enable control register (CKEN) controls the PLLC1 OBSCLK clock. CKEN is shown in [Figure 7-32](#) and described in [Table 7-34](#).

**Figure 7-32. PLLC1 Clock Enable Control Register (CKEN)**

31	Reserved			16	
R-0					
15	Reserved		2	1	0
R-0			R/W-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

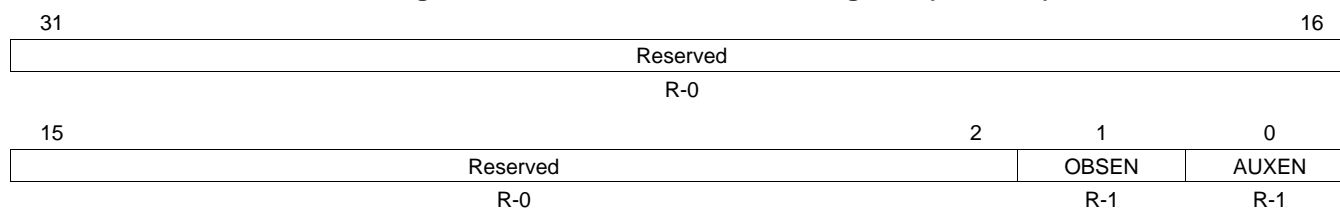
**Table 7-34. PLLC1 Clock Enable Control Register (CKEN) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN	0	OBSCLK enable. Actual PLLC1 OBSCLK status is shown in the PLLC1 clock status register (CKSTAT). PLL1 OBSCLK is disabled.
		1	PLL1 OBSCLK is enabled. For PLLC1 OBSCLK to toggle, both the OBSEN bit and the OD1EN bit in the PLLC1 oscillator divider 1 register (OSCDIV) must be set to 1.
0	Reserved	0	Reserved

### 7.3.32 PLLC0 Clock Status Register (CKSTAT)

The PLLC0 clock status register (CKSTAT) indicates the PLLC0 OBSCLK and AUXCLK on/off status. The PLL0\_SYSCLK status is shown in the PLLC0 SYSCLK status register (SYSTAT). CKSTAT is shown in [Figure 7-33](#) and described in [Table 7-35](#).

**Figure 7-33. PLLC0 Clock Status Register (CKSTAT)**



LEGEND: R = Read only; -n = value after reset

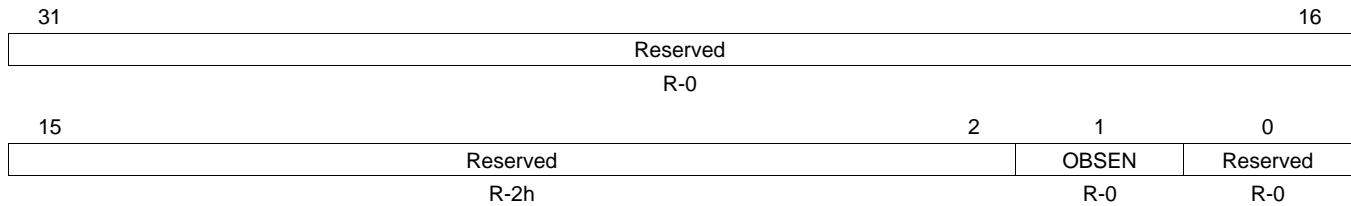
**Table 7-35. PLLC0 Clock Status Register (CKSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN	0	OBSCCLK on status. PLLC0 OBSCCLK is controlled in the PLLC0 oscillator divider 1 register (OSCDIV) by the OBSEN bit in the PLLC0 clock enable control register (CKEN).
		1	PLLC0 OBSCCLK is off. PLLC0 OBSCCLK is on.
0	AUXEN	0	AUXCLK on status. PLLC0 AUXCLK is controlled by the AUXEN bit in the PLLC0 clock enable control register (CKEN).
		1	PLLC0 AUXCLK is off. PLLC0 AUXCLK is on.

### 7.3.33 PLLC1 Clock Status Register (CKSTAT)

The PLLC1 clock status register (CKSTAT) indicates the PLLC1 OBSCLK on/off status. The PLL1\_SYSCLK status is shown in the PLLC1 SYSCLK status register (SYSTAT). CKSTAT is shown in [Figure 7-34](#) and described in [Table 7-36](#).

**Figure 7-34. PLLC1 Clock Status Register (CKSTAT)**



LEGEND: R = Read only; -n = value after reset

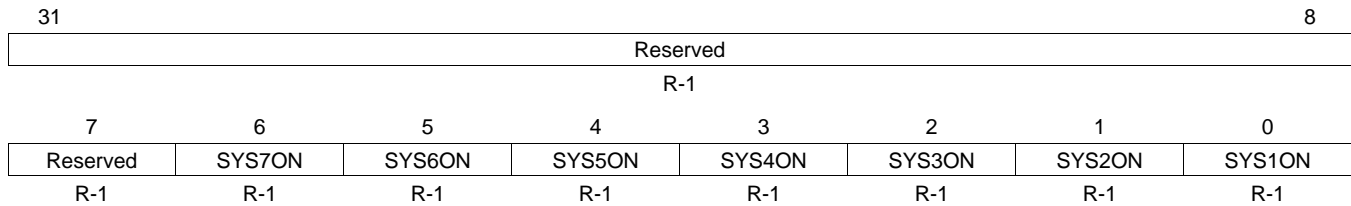
**Table 7-36. PLLC1 Clock Status Register (CKSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN	0	OBSClk on status. PLLC1 OBSClk is controlled in the PLLC1 oscillator divider 1 register (OSCDIV) by the OBSEN bit in the PLLC1 clock enable control register (CKEN). PLLC1 OBSClk is off.
		1	PLLC1 OBSClk is on.
0	Reserved	0	Reserved

### 7.3.34 PLLC0 SYSCLK Status Register (SYSTAT)

The PLLC0 SYSCLK status register (SYSTAT) indicates the PLL0\_SYSCLK $n$  on/off status. The actual default is determined by the actual clock on/off status, which depends on the D $n$ EN bit in PLLC0 PLLDIV $n$ . SYSTAT is shown in [Figure 7-35](#) and described in [Table 7-37](#).

**Figure 7-35. PLLC0 SYSCLK Status Register (SYSTAT)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

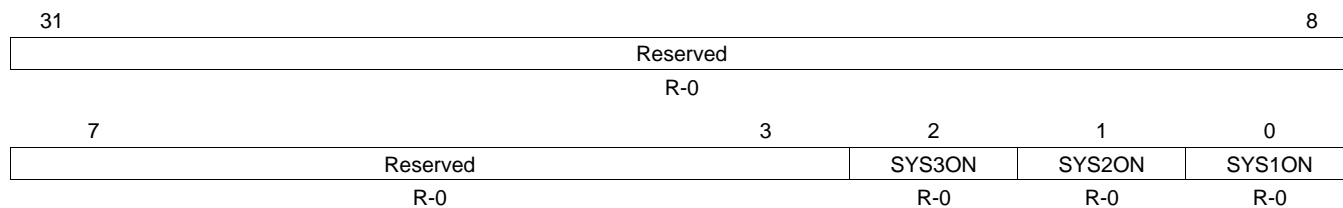
**Table 7-37. PLLC0 SYSCLK Status Register (SYSTAT) Field Descriptions**

Bit	Field	Value	Description
31-7	Reserved	3h	Reserved
6	SYS7ON	0 1	PLL0_SYSCLK7 on status. Off On
5	SYS6ON	0 1	PLL0_SYSCLK6 on status. Off On
4	SYS5ON	0 1	PLL0_SYSCLK5 on status. Off On
3	SYS4ON	0 1	PLL0_SYSCLK4 on status. Off On
2	SYS3ON	0 1	PLL0_SYSCLK3 on status. Off On
1	SYS2ON	0 1	PLL0_SYSCLK2 on status. Off On
0	SYS1ON	0 1	PLL0_SYSCLK1 on status. Off On

### 7.3.35 PLL<sub>C</sub>1 SYSCLK Status Register (SYSTAT)

The PLL<sub>C</sub>1 SYSCLK status register (SYSTAT) indicates the PLL1\_SYSCLK<sub>n</sub> on/off status. The actual default is determined by the actual clock on/off status, which depends on the D<sub>n</sub>EN bit in PLL<sub>C</sub>1 PLLDIV<sub>n</sub>. SYSTAT is shown in [Figure 7-36](#) and described in [Table 7-38](#).

**Figure 7-36. PLL<sub>C</sub>1 SYSCLK Status Register (SYSTAT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-38. PLL<sub>C</sub>1 SYSCLK Status Register (SYSTAT) Field Descriptions**

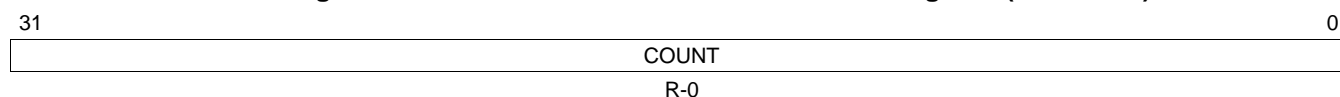
Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	SYS3ON	0 1	PLL1_SYSCLK3 on status. Off On
1	SYS2ON	0 1	PLL1_SYSCLK2 on status. Off On
0	SYS1ON	0 1	PLL1_SYSCLK1 on status. Off On



### 7.3.36 Emulation Performance Counter 0 Register (EMUCNT0)

The emulation performance counter 0 register (EMUCNT0) is shown in [Figure 7-37](#) and described in [Table 7-39](#). EMUCNT0 is for emulation performance profiling. It counts in a divide-by-4 of the system clock. To start the counter, a write must be made to EMUCNT0. This register is not writable, but only used to start the register. After the register is started, it can not be stopped except for power on reset. When EMUCNT0 is read, it snapshots EMUCNT0 and EMUCNT1. The snapshot version is what is read. It is important to read the EMUCNT0 followed by EMUCNT1 or else the snapshot version may not get updated correctly.

**Figure 7-37. Emulation Performance Counter 0 Register (EMUCNT0)**



LEGEND: R = Read only; -n = value after reset

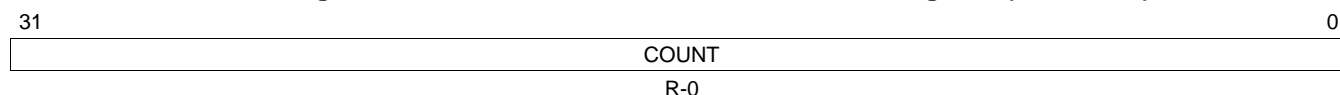
**Table 7-39. Emulation Performance Counter 0 Register (EMUCNT0) Field Descriptions**

Bit	Field	Value	Description
31-0	COUNT	0-FFFF FFFFh	Counter value for lower 64-bits.

### 7.3.37 Emulation Performance Counter 1 Register (EMUCNT1)

The emulation performance counter 1 register (EMUCNT1) is shown in [Figure 7-38](#) and described in [Table 7-40](#). EMUCNT1 is for emulation performance profiling. To start the counter, a write must be made to EMUCNT0. This register is not writable, but only used to start the register. After the register is started, it can not be stopped except for power on reset. When EMUCNT0 is read, it snapshots EMUCNT0 and EMUCNT1. The snapshot version is what is read. It is important to read the EMUCNT0 followed by EMUCNT1 or else the snapshot version may not get updated correctly.

**Figure 7-38. Emulation Performance Counter 1 Register (EMUCNT1)**



LEGEND: R = Read only; -n = value after reset

**Table 7-40. Emulation Performance Counter 1 Register (EMUCNT1) Field Descriptions**

Bit	Field	Value	Description
31-0	COUNT	0-FFFF FFFFh	Counter value for upper 64-bits.

---

---

## ***Power and Sleep Controller (PSC)***

---

---

Topic	Page
8.1 Introduction .....	<b>128</b>
8.2 Power Domain and Module Topology .....	<b>128</b>
8.3 Executing State Transitions .....	<b>132</b>
8.4 IcePick Emulation Support in the PSC .....	<b>133</b>
8.5 PSC Interrupts.....	<b>133</b>
8.6 PSC Registers .....	<b>136</b>

## 8.1 Introduction

The Power and Sleep Controllers (PSC) are responsible for managing transitions of system power on/off, clock on/off, resets (device level and module level). It is used primarily to provide granular power control for on chip modules (peripherals and CPU). A PSC module consists of a Global PSC (GPSC) and a set of Local PSCs (LPSCs).

The GPSC contains memory mapped registers, PSC interrupts, a state machine for each peripheral/module it controls. An LPSC is associated with every module that is controlled by the PSC and provides clock and reset control. Many of the operations of the PSC are transparent to user (software), such as power on and reset control. However, the PSC module(s) also provide you with interface to control several important power, clock and reset operations. The module level power, clock and reset operations managed and controlled by the PSC are the focus of this chapter.

The PSC includes the following features:

- Manages chip power-on/off
- Provides a software interface to:
  - Control module clock enable/disable
  - Control module reset
  - Control CPU local reset
- Manages on-chip RAM sleep modes (for DSP memories)
- Supports IcePick emulation features: power, clock and reset

## 8.2 Power Domain and Module Topology

This device includes two PSC modules. Each PSC module consists of:

- an Always On power domain
- an additional pseudo/internal power domain that manages the sleep modes for the RAMs present in the DSP subsystem

Each PSC module controls clock states for several on the on chip modules, controllers and interconnect components. [Table 8-1](#) and [Table 8-2](#) lists the set of peripherals/modules that are controlled by the PSC, the power domain they are associated with, the LPSC assignment and the default (power-on reset) module states. See the device-specific data manual for the peripherals available on a given device. The module states and terminology are defined in [Section 8.2.2](#).

Even though there are 2 PSC modules with 2 power domains each on the device, both PSC modules and all the power domains are powered by the CVDD pins of the device. All power domains are on when the chip is powered on. There is no provision to remove power externally for the non Always On domains, that is, the pseudo/internal power domains.

There are a few modules/peripherals on the device that do not have an LPSC assigned to them. These modules do not have their module reset/clocks controlled by the PSC module. The decision to assign an LPSC to a module on a device is primarily based on whether or not disabling the clocks to a module will result in significant power savings. This typically depends on the size and the frequency of operation of the module.

---

**NOTE:** There are no LPSCs for peripherals in the Async2 clock domain (this includes RTC, Timer64P0/P1, and I2C0); from a power savings stand point, clock-gating these peripherals does not result in significant power savings.

---

**Table 8-1. PSC0 Default Module Configuration**

LPSC Number	Module Name	Power Domain	Default Module State	Auto Sleep/Wake Only
0	EDMA3_0 Channel Controller 0	AlwaysON (PD0)	SwRstDisable	—
1	EDMA3_0 Transfer Controller 0	AlwaysON (PD0)	SwRstDisable	—
2	EDMA3_0 Transfer Controller 1	AlwaysON (PD0)	SwRstDisable	—
3	EMIFA (BR7)	AlwaysON (PD0)	SwRstDisable	—
4-8	Not Used	—	—	—
9	UART0	AlwaysON (PD0)	SwRstDisable	—
10	Not Used	—	—	—
11	SCR1 (BR4)	AlwaysON (PD0)	Enable	Yes
12	SCR2 (BR3, BR5, BR6)	AlwaysON (PD0)	Enable	Yes
13-14	Not Used	—	—	—
15	DSP	PD_DSP (PD1)	Enable	—

**Table 8-2. PSC1 Default Module Configuration**

LPSC Number	Module Name	Power Domain	Default Module State	Auto Sleep/Wake Only
0	EDMA3_1 Channel Controller 0	AlwaysON (PD0)	SwRstDisable	—
1-2	Not Used	—	—	—
1-2	Not Used	—	—	—
3	GPIO	AlwaysON (PD0)	SwRstDisable	—
4	HPI	AlwaysON (PD0)	SwRstDisable	—
5	Not Used	—	—	—
6	DDR2/mDDR	AlwaysON (PD0)	SwRstDisable	—
7	McASP0 (+ McASP0 FIFO)	AlwaysON (PD0)	SwRstDisable	—
8-9	Not Used	—	—	—
10	SPI1	AlwaysON (PD0)	SwRstDisable	—
11-14	Not Used	—	—	—
15	McBSP1 (+ McBSP1 FIFO)	AlwaysON (PD0)	SwRstDisable	—
16	Not Used	—	—	—
17	eHRPWM0/1	AlwaysON (PD0)	SwRstDisable	—
18-19	Not Used	—	—	—
20	eCAP0/1/2	AlwaysON (PD0)	SwRstDisable	—
21	EDMA3_1 Transfer Controller 0	AlwaysON (PD0)	SwRstDisable	—
22-23	Not Used	—	—	—
24	SCR F0	AlwaysON (PD0)	Enable	Yes
25-26	Not Used	—	—	—
27	SCR F6	AlwaysON (PD0)	Enable	Yes
28	SCR F7	AlwaysON (PD0)	Enable	Yes
29	SCR F8	AlwaysON (PD0)	Enable	Yes
30-31	Not Used	—	—	—

### 8.2.1 Power Domain States

A power domain can only be in one of the two states: ON or OFF, defined as follows:

- ON: power to the domain is on
- OFF: power to the domain is off

In this device, for both PSC0 and PSC1, the Always ON domain (or PD0 power domain), is always in the ON state when the chip is powered-on. This domain is not programmable to OFF state (See details on PDCTL register).

Additionally, for both PSC0 and PSC1, the PD1 power domains, the internal/pseudo power domain can either be in the ON state or OFF state. Furthermore, for these power domains the transition from ON to OFF state is further qualified by the PSC0/1.PDCTL1.PDMODE settings. The PDCTL1.PDMODE settings determines the various sleep mode for the on-chip RAM associated with module in the PD1 domain.

- On PSC0 PD1/PD\_DSP Domain: Controls the sleep state for DSP L1 and L2 Memories

---

**NOTE:** Currently programming the PD1 power domain state to OFF is not supported. You should leave both the PDCTL1.NEXT and PDCTL1.PDMODE values at default/power on reset values.

Both PD0 and PD1 power domains in PSC0 and PSC1 are powered by the CVDD pins of the device. There is no capability to individually remove voltage/power from the DSP power domains .

---

### 8.2.2 Module States

The PSC defines several possible states for a module. This various states are essentially a combination of the module reset asserted or de-asserted and module clock on/enabled or off/disabled. The various module states are defined in [Table 8-3](#).

The key difference between the Auto Sleep and Auto Wake states is that once the module is configured in Auto Sleep mode, it will transition back to the clock disabled state (automatically sleep) after servicing the internal read/write access request where as in Auto Wake mode, on receiving the first internal read/write access request, the module will permanently transition from the clock disabled to clock enabled state (automatically wake).

When the module state is programmed to Disable, SwRstDisable, Auto Sleep or Auto Wake modes, where in the module clocks are off/disabled, an external event or I/O request cannot enable the clocks. For the module to appropriately respond to such external request, it would need to be reconfigured to the Enable state.

#### 8.2.2.1 Auto Sleep/Wake Only Configurations and Limitation

---

**NOTE:** Currently no modules should be configured in Auto Sleep or Auto Wake modes. If the module clocks need to gated/disabled for power savings, you should program the module state to Disable. For Auto Sleep/Auto Wake Only modules, disabling the clock is not supported and they should be kept in their default "Enable" state.

---

[Table 8-1](#) and [Table 8-2](#) each have a column to indicate whether or not the LPSC configuration for a module is Auto Sleep/Wake Only. Modules that have a "Yes" marked for the Auto Sleep/Wake Only column can be programmed in software to be in Enable, Auto Sleep and Auto Wake states only; that is, if the software tries to program these modules to Disable, SyncReset, or SwRstDisable state the power sleep controller ignores these transition requests and transitions the module state to Enable.

### 8.2.2.2 Local Reset

In addition to module reset, the following module can be reset using a special local reset that is also a part of the PSC module control for resets.

- **DSP:** When the DSP local reset is asserted the DSP internal memories (L1P, L1D and L2) are still accessible. The local reset only resets the DSP CPU core, not the rest of DSP subsystem, as the DSP module reset would. Local Reset is useful in cases where the DSP is in enable or disable state; since when module is in SyncReset or SwRstDisable state the module reset is asserted, and the module reset takes precedence over the local reset.

The procedures for asserting and de-asserting the local reset are as follows (where  $n$  corresponds to the module that supports local reset):

1. Clear the LRST bit in the module control register (MDCTL $n$ ) to 0 to assert the module's local reset.
2. Set the LRST bit in the module control register (MDCTL $n$ ) to 1 to de-assert module's local reset.

If the CPU is in the enable state, it immediately executes program instructions after reset is de-asserted.

**Table 8-3. Module States**

Module State	Module Reset	Module Clock	Module State Definition
Enable	De-asserted	On	A module in the enable state has its module reset de-asserted and it has its clock on. This is the normal operational state for a given module
Disable	De-asserted	Off	A module in the disabled state has its module reset de-asserted and it has its module clock off. This state is typically used for disabling a module clock to save power. This device is designed in full static CMOS, so when you stop a module clock, it retains the module's state. When the clock is restarted, the module resumes operating from the stopping point.
SyncReset	Asserted	On	A module state in the SyncReset state has its module reset asserted and it has its clock on. Generally, software is not expected to initiate this state
SwRstDisable	Asserted	Off	A module in the SwResetDisable state has its module reset asserted and it has its clock disabled. After initial power-on, several modules come up in the SwRstDisable state. Generally, software is not expected to initiate this state
Auto Sleep	De-asserted	Off	A module in the Auto Sleep state also has its module reset de-asserted and its module clock disabled, similar to the Disable state. However this is a special state, once a module is configured in this state by software, it can "automatically" transition to "Enable" state whenever there is an internal read/write request made to it, and after servicing the request it will "automatically" transition into the sleep state (with module reset re de-asserted and module clock disabled), without any software intervention. The transition from sleep to enabled and back to sleep state has some cycle latency associated with it. It is not envisioned to use this mode when peripherals are fully operational and moving data. See <a href="#">Section 8.2.2.1</a> for additional considerations, constraints, limitations around this mode.
Auto Wake	De-asserted	Off	A module in the Auto Wake state also has its module reset de-asserted and its module clock disabled, similar to the Disable state. However this is a special state, once a module is configured in this state by software, it will "automatically" transition to "Enable" state whenever there is an internal read/write request made to it, and will remain in the "Enabled" state from then on (with module reset re de-asserted and module clock on), without any software intervention. The transition from sleep to enabled state has some cycle latency associated with it. It is not envisioned to use this mode when peripherals are fully operational and moving data. See <a href="#">Section 8.2.2.1</a> for additional considerations, constraints, limitations around this mode.

## 8.3 Executing State Transitions

This section describes how to execute the state transitions modules.

### 8.3.1 Power Domain State Transitions

This device consists of two types of domain (in each PSC controller):

- Always On domain(s)
- pseudo/RAM power domain(s)

The Always On power domains are always in the ON state when the chip is powered on. You are not allowed to change the power domain state to OFF.

The pseudo/RAM power domains allow internally powering down the state of the RAMs associated with these domains (L1/L2 for PD\_DSP in PSC0) so that these RAMs can run in lower power sleep modes via the power sleep controller.

---

**NOTE:** Currently powering down the RAMs via the pseudo/RAM power domain is not supported; therefore, these domains and the RAM should be left in their default power on state.

As mentioned in [Section 8.2](#), the pseudo/RAM power domains are powered down internally, and in this context powering down does not imply removing the core voltage from pins externally.

---

### 8.3.2 Module State Transitions

This section describes the procedure for transitioning the module state (clock and reset control). Note that some peripherals have special programming requirements and additional recommended steps you must take before you can invoke the PSC module state transition. See the individual peripheral user guides for more details. For example, the external memory controller requires that you first place the SDRAM memory in self-refresh mode before you invoke the PSC module state transitions, if you want to maintain the memory contents.

The following procedure is directly applicable for all modules that are controlled via the PSC (shown in [Table 8-1](#) and [Table 8-2](#)), except for the core(s). To transition the DSP module state, there are additional system considerations and constraints that you should be aware of. These system considerations and the procedure for transitioning the DSP module state are described in details in the *Power Management* chapter.

---

**NOTE:** In the following procedure, x is 0 for modules in PD0 (Power Domain 0 or Always On domain) and x is 1 for modules in PD1 (Power Domain 1). See [Table 8-1](#) and [Table 8-2](#) for power domain associations.

---

The procedure for module state transitions is:

1. Wait for the GOSTAT[x] bit in PTSTAT to clear to 0. You must wait for any previously initiated transitions to finish before initiating a new transition.
2. Set the NEXT bit in MDCTL<sub>n</sub> to SwRstDisable (0), SyncReset (1), Disable (2h), Enable (3h), Auto Sleep (4h) or Auto Wake (5h).

---

**NOTE:** You may set transitions in multiple NEXT bits in MDCTL<sub>n</sub> in this step. Transitions do not actually take place until you set the GO[x] bit in PTCMD in a later step.

---

3. Set the GO[x] bit in PTCMD to 1 to initiate the transition(s).
4. Wait for the GOSTAT[x] bit in PTSTAT to clear to 0. The modules are safely in the new states only after the GOSTAT[x] bit in PTSTAT is cleared to 0.

## 8.4 IcePick Emulation Support in the PSC

The PSC supports IcePick commands that allow IcePick emulation tools to have some control over the state of power domains and modules. This IcePick support only applies to the following module:

- DSP [MDCTL15]

In particular, [Table 8-4](#) shows IcePick emulation commands recognized by the PSC.

**Table 8-4. IcePick Emulation Commands**

Power On and Enable Features	Power On and Enable Descriptions	Reset Features	Reset Descriptions
Inhibit Sleep	Allows emulation to prevent software from transitioning the module out of the enable state.	Assert Reset	Allows emulation to assert the module's local reset.
Force Power	Allows emulation to force the power domain into an on state. Not applicable as AlwaysOn power domain is always on.	Wait Reset	Allows emulation to keep local reset asserted for an extended period of time after software initiates local reset de-assert.
Force Active	Allows emulation to force the module into the enable state.	Block Reset	Allows emulation to block software initiated local and module resets.

---

**NOTE:** When emulation tools remove the above commands, the PSC immediately executes a state transition based on the current values in the NEXT bit in PDCTL0 and the NEXT bit in MDCTL $n$ , as set by software.

---

## 8.5 PSC Interrupts

The PSC has an interrupt that is tied to the core interrupt controller. This interrupt is named PSCINT in the interrupt map. The PSC interrupt is generated when certain IcePick emulation events occur.

### 8.5.1 Interrupt Events

The PSC interrupt is generated when any of the following events occur:

- Power Domain Emulation Event (applies to pseudo/RAM power domain only)
- Module State Emulation event
- Module Local Reset Emulation event

These interrupt events are summarized in [Table 8-5](#) and described in more detail in this section.

**Table 8-5. PSC Interrupt Events**

Interrupt Enable Bits		
Control Register	Enable Bit	Interrupt Condition
PDCTL $n$	EMUIHBIE	Interrupt occurs when the emulation alters the power domain state
MDCTL $n$	EMUIHBIE	Interrupt occurs when the emulation alters the module state
MDCTL $n$	EMURSTIE	Interrupt occurs when the emulation tries to alter the module's local reset

The PSC interrupt events only apply when IcePick emulation alters the state of the module from the user-programmed state in the NEXT bit in the MDCTL/PDCTL registers. IcePick support only applies to the modules listed in [Section 8.4](#); therefore, the PSC interrupt conditions only apply to those modules listed.



### 8.5.1.1 Power Domain Emulation Events

A power domain emulation event occurs when emulation alters the state of a power domain (does not apply to the Always On domain). Status is reflected in the EMUIHB bit in PDSTAT $n$ . In particular, a power domain emulation event occurs under the following conditions:

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the on state
- When force power is asserted by emulation and power domain is not already in the on state
- When force active is asserted by emulation and power domain is not already in the on state

---

**NOTE:** Putting the pseudo/RAM power domain associated with the DSP (PD\_DSP) to the off state currently is **not** supported.

---

### 8.5.1.2 Module State Emulation Events

A module state emulation event occurs when emulation alters the state of a module. Status is reflected in the EMUIHB bit in the module status register (MDSTAT $n$ ). In particular, a module state emulation event occurs under the following conditions:

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the enable state
- When force active is asserted by emulation and module is not already in the enable state

### 8.5.1.3 Local Reset Emulation Events

A local reset emulation event occurs when emulation alters the local reset of a module. Status is reflected in the EMURST bit in the module status register (MDSTAT $n$ ). In particular, a module local reset emulation event occurs under the following conditions:

- When assert reset is asserted by emulation although software de-asserted the local reset
- When wait reset is asserted by emulation
- When block reset is asserted by emulation and software attempts to change the state of local reset

## 8.5.2 Interrupt Registers

The PSC interrupt enable bits are: the EMUIHBIE bit in PDCTL1 (PSC0), the EMUIHBIE and the EMURSTIE bits in MDCTL $n$  (where  $n$  is the modules that have IcePick emulation support, as specified in [Section 8.4](#)).

---

**NOTE:** To interrupt the CPU, the power sleep controller interrupt (PSC0\_ALLINT and PSC1\_ALLINT) must also be enabled in the DSP interrupt controller. For details on the DSP interrupt controller, see the *DSP Subsystem* chapter.

---

The PSC interrupt status bits are:

- For DSP:
  - The M[15] bit in the module error pending register 0 (MERRPR0) in PSC0 module.
  - The EMUIHB and the EMURST bits in the module status register for DSP (MDSTAT15).
  - The P[1] bit in the power error pending register (PERRPR) for the pseudo/RAM power domain associated with DSP memories.

The status bit in MERRPR0 and PERRPR registers is read by software to determine which module or power domain has generated an emulation interrupt and then software can read the corresponding status bits in MDSTAT register or the PDSTAT $n$  (PDCTL1 for pseudo/RAM power domain in PSC0) to determine which event caused the interrupt.

The PSC interrupt can be cleared by writing to bit corresponding to the module number in the module error clear register (MERRCR0), or the bit corresponding to the power domain number in the power error clear register (PERRCR) in PSC0 module.

The PSC interrupt evaluation bit is the ALLEV bit in the INTEVAL register. When set, this bit forces the PSC interrupt logic to re-evaluate event status. If any events are still active (if any status bits are set) when the ALLEV bit in the INTEVAL is set to 1, the PSC interrupt is re-asserted to the interrupt controller. Set the ALLEV bit in the INTEVAL before exiting your PSC interrupt service routine to ensure that you do not miss any PSC interrupts.

See [Section 8.6](#) for a description of the PSC registers.

### 8.5.3 Interrupt Handling

Handle the PSC interrupts as described in the following procedure:

First, enable the interrupt:

1. Set the EMUIHBIE bit in PDCTL $n$ , the EMUIHBIE and the EMURSTIE bits in MDCTL $n$  to enable the interrupt events that you want.

---

**NOTE:** The PSC interrupt is sent to the device interrupt controller when at least one enabled event becomes active.

---

2. Enable the power sleep controller interrupt (PSC $n$ \_ALLINT) in the device interrupt controller. To interrupt the CPU, PSC $n$ \_ALLINT must be enabled in the device interrupt controller. See the *DSP Subsystem* chapter for more information on interrupts.

The CPU enters the interrupt service routine (ISR) when it receives the interrupt.

1. Read the P[n] bit in PERRPR, and/or the M[n] bit in MERRPR0, the M[n] bit in MERRPR1, to determine the source of the interrupt(s).
2. For each active event that you want to service:
  - (a) Read the event status bits in PDSTAT $n$  and MDSTAT $n$ , depending on the status bits read in the previous step to determine the event that caused the interrupt.
  - (b) Service the interrupt as required by your application.
  - (c) Write the M[n] bit in MERRCR $n$  and the P[n] bit in PERRCR to clear corresponding status.
  - (d) Set the ALLEV bit in INTEVAL. Setting this bit reasserts the PSC interrupt to the device interrupt controller, if there are still any active interrupt events.

## 8.6 PSC Registers

Table 8-6 lists the memory-mapped registers for the PSC0 and Table 8-7 lists the memory-mapped registers for the PSC1.

**Table 8-6. Power and Sleep Controller 0 (PSC0) Registers**

Address	Acronym	Register Description	Section
01C1 0000h	REVID	Revision Identification Register	<a href="#">Section 8.6.1</a>
01C1 0018h	INTEVAL	Interrupt Evaluation Register	<a href="#">Section 8.6.2</a>
01C1 0040h	MERRPR0	Module Error Pending Register 0 (module 0-15)	<a href="#">Section 8.6.3</a>
01C1 0050h	MERRCR0	Module Error Clear Register 0 (module 0-15)	<a href="#">Section 8.6.5</a>
01C1 0060h	PERRPR	Power Error Pending Register	<a href="#">Section 8.6.7</a>
01C1 0068h	PERRCR	Power Error Clear Register	<a href="#">Section 8.6.8</a>
01C1 0120h	PTCMD	Power Domain Transition Command Register	<a href="#">Section 8.6.9</a>
01C1 0128h	PTSTAT	Power Domain Transition Status Register	<a href="#">Section 8.6.10</a>
01C1 0200h	PDSTAT0	Power Domain 0 Status Register	<a href="#">Section 8.6.11</a>
01C1 0204h	PDSTAT1	Power Domain 1 Status Register	<a href="#">Section 8.6.12</a>
01C1 0300h	PDCTL0	Power Domain 0 Control Register	<a href="#">Section 8.6.13</a>
01C1 0304h	PDCTL1	Power Domain 1 Control Register	<a href="#">Section 8.6.14</a>
01C1 0400h	PDCFG0	Power Domain 0 Configuration Register	<a href="#">Section 8.6.15</a>
01C1 0404h	PDCFG1	Power Domain 1 Configuration Register	<a href="#">Section 8.6.16</a>
01C1 0800h- 01C1 083Ch	MDSTAT0- MDSTAT15	Module Status <i>n</i> Register (modules 0-15)	<a href="#">Section 8.6.17</a>
01C1 0A00h- 01C1 0A3Ch	MDCTL0- MDCTL15	Module Control <i>n</i> Register (modules 0-15)	<a href="#">Section 8.6.18</a>

**Table 8-7. Power and Sleep Controller 1 (PSC1) Registers**

Address	Acronym	Register Description	Section
01E2 7000h	REVID	Revision Identification Register	<a href="#">Section 8.6.1</a>
01E2 7018h	INTEVAL	Interrupt Evaluation Register	<a href="#">Section 8.6.2</a>
01E2 7040h	MERRPR0	Module Error Pending Register 0 (module 0-31)	<a href="#">Section 8.6.4</a>
01E2 7050h	MERRCR0	Module Error Clear Register 0 (module 0-31)	<a href="#">Section 8.6.6</a>
01E2 7060h	PERRPR	Power Error Pending Register	<a href="#">Section 8.6.7</a>
01E2 7068h	PERRCR	Power Error Clear Register	<a href="#">Section 8.6.8</a>
01E2 7120h	PTCMD	Power Domain Transition Command Register	<a href="#">Section 8.6.9</a>
01E2 7128h	PTSTAT	Power Domain Transition Status Register	<a href="#">Section 8.6.10</a>
01E2 7200h	PDSTAT0	Power Domain 0 Status Register	<a href="#">Section 8.6.11</a>
01E2 7204h	PDSTAT1	Power Domain 1 Status Register	<a href="#">Section 8.6.12</a>
01E2 7300h	PDCTL0	Power Domain 0 Control Register	<a href="#">Section 8.6.13</a>
01E2 7304h	PDCTL1	Power Domain 1 Control Register	<a href="#">Section 8.6.14</a>
01E2 7400h	PDCFG0	Power Domain 0 Configuration Register	<a href="#">Section 8.6.15</a>
01E2 7404h	PDCFG1	Power Domain 1 Configuration Register	<a href="#">Section 8.6.16</a>
01E2 7800h- 01E2 787Ch	MDSTAT0- MDSTAT31	Module Status <i>n</i> Register (modules 0-31)	<a href="#">Section 8.6.17</a>
01E2 7A00h- 01E2 7A7Ch	MDCTL0- MDCTL31	Module Control <i>n</i> Register (modules 0-31)	<a href="#">Section 8.6.19</a>

### 8.6.1 Revision Identification Register (REVID)

The revision identification register (REVID) is shown in [Figure 8-1](#) and described in [Table 8-8](#).

**Figure 8-1. Revision Identification Register (REVID)**



LEGEND: R = Read only; -n = value after reset

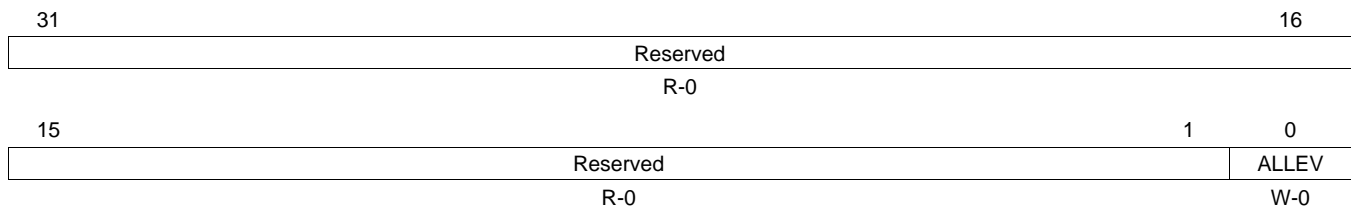
**Table 8-8. Revision Identification Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4482 5A00h	Peripheral revision ID.

### 8.6.2 Interrupt Evaluation Register (INTEVAL)

The interrupt evaluation register (INTEVAL) is shown in [Figure 8-2](#) and described in [Table 8-9](#).

**Figure 8-2. Interrupt Evaluation Register (INTEVAL)**



LEGEND: R = Read only; W= Write only; -n = value after reset

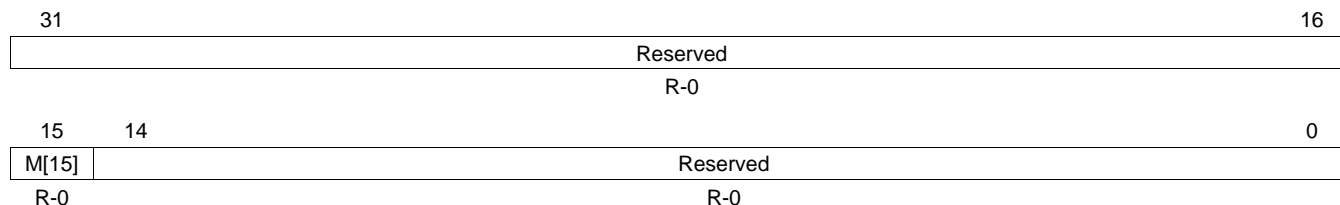
**Table 8-9. Interrupt Evaluation Register (INTEVAL) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ALLEV		Evaluate PSC interrupt (PSC <sub>n</sub> _ALLINT).
		0	A write of 0 has no effect.
		1	A write of 1 re-evaluates the interrupt condition.

### 8.6.3 PSC0 Module Error Pending Register 0 (modules 0-15) (MERRPR0)

The PSC0 module error pending register 0 (MERRPR0) is shown in [Figure 8-3](#) and described in [Table 8-10](#).

**Figure 8-3. PSC0 Module Error Pending Register 0 (MERRPR0)**



LEGEND: R = Read only; -n = value after reset

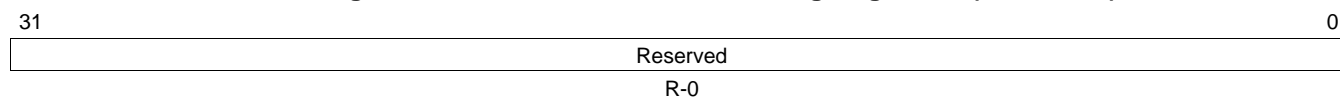
**Table 8-10. PSC0 Module Error Pending Register 0 (MERRPR0) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	M[15]	0	Module interrupt status bit for module 15 (DSP).
		1	Module 15 does not have an error condition.
			Module 15 has an error condition. See the module status 15 register (MDSTAT15) for the error condition.
14-0	Reserved	0	Reserved

### 8.6.4 PSC1 Module Error Pending Register 0 (modules 0-31) (MERRPR0)

The PSC1 module error pending register 0 (MERRPR0) is shown in [Figure 8-4](#).

**Figure 8-4. PSC1 Module Error Pending Register 0 (MERRPR0)**

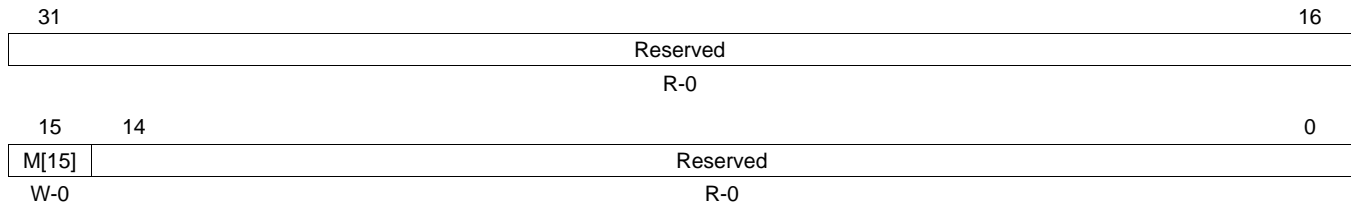


LEGEND: R = Read only; -n = value after reset

### 8.6.5 PSC0 Module Error Clear Register 0 (modules 0-15) (MERRCR0)

The PSC0 module error clear register 0 (MERRCR0) is shown in [Figure 8-5](#) and described in [Table 8-11](#).

**Figure 8-5. PSC0 Module Error Clear Register 0 (MERRCR0)**



LEGEND: R = Read only; W = Write only; -n = value after reset

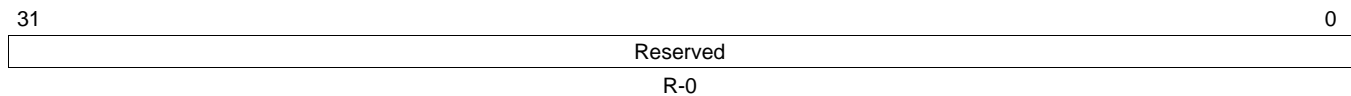
**Table 8-11. PSC0 Module Error Clear Register 0 (MERRCR0) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	M[15]	0	Clears the interrupt status bit (M[15]) set in the PSC0 module error pending register 0 (MERRPR0) and the interrupt status bits set in the module status 15 register (MDSTAT15). A write of 0 has no effect.
		1	A write of 1 clears the M[15] bit in MERRPR0 and the EMUIHB and EMURST bits in MDSTAT15.
14-0	Reserved	0	Reserved

### 8.6.6 PSC1 Module Error Clear Register 0 (modules 0-31) (MERRCR0)

The PSC1 module error clear register 0 (MERRCR0) is shown in [Figure 8-6](#).

**Figure 8-6. PSC1 Module Error Clear Register 0 (MERRCR0)**

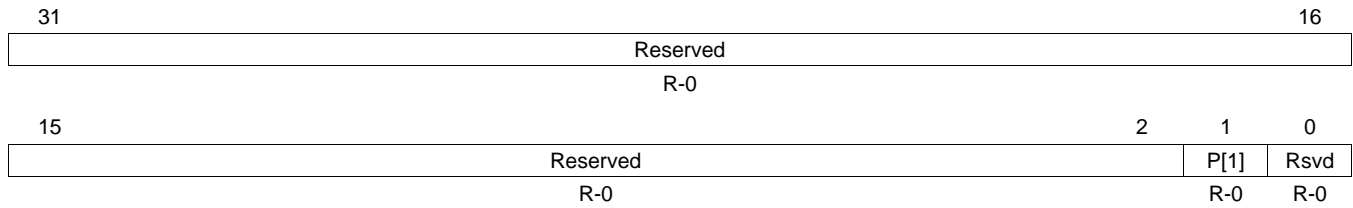


LEGEND: R = Read only; -n = value after reset

### 8.6.7 Power Error Pending Register (PERRPR)

The power error pending register (PERRPR) is shown in [Figure 8-7](#) and described in [Table 8-12](#).

**Figure 8-7. Power Error Pending Register (PERRPR)**



LEGEND: R = Read only; -n = value after reset

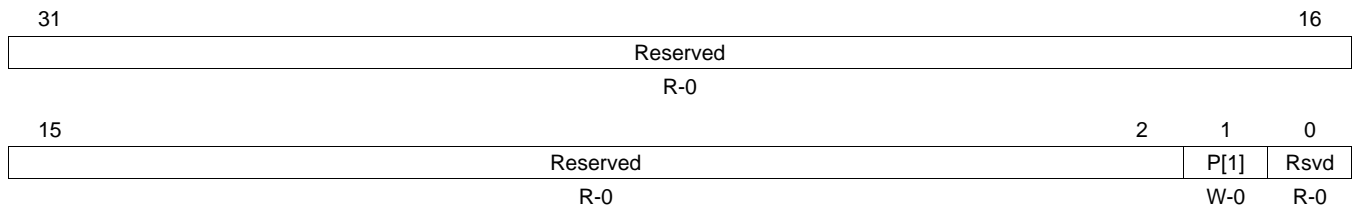
**Table 8-12. Power Error Pending Register (PERRPR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	P[1]	0	RAM/Pseudo (PD1) power domain interrupt status. RAM/Pseudo power domain does not have an error condition.
		1	RAM/Pseudo power domain has an error condition. See the power domain 1 status register (PDSTAT1) for the error condition.
0	Reserved	0	Reserved

### 8.6.8 Power Error Clear Register (PERRCR)

The power error clear register (PERRCR) is shown in [Figure 8-8](#) and described in [Table 8-13](#).

**Figure 8-8. Power Error Clear Register (PERRCR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

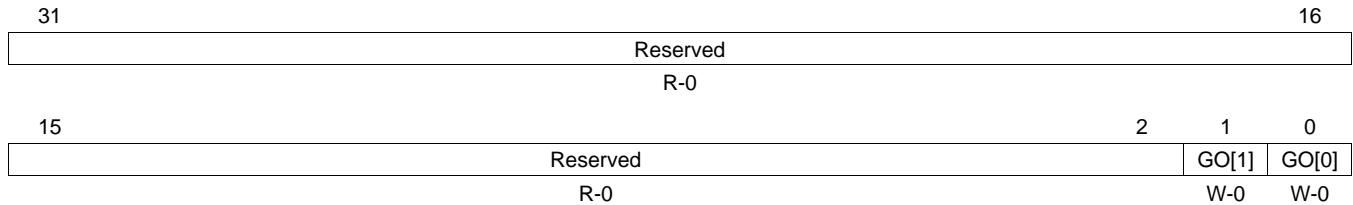
**Table 8-13. Power Error Clear Register (PERRCR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	P[1]	0	Clears the interrupt status bit (P) set in the power error pending register (PERRPR) and the interrupt status bits set in the power domain 1 status register (PDSTAT1). A write of 0 has no effect.
		1	A write of 1 clears the P bit in PERRPR and the interrupt status bits in PDSTAT1.
0	Reserved	0	Reserved

### 8.6.9 Power Domain Transition Command Register (PTCMD)

The power domain transition command register (PTCMD) is shown in [Figure 8-9](#) and described in [Table 8-14](#).

**Figure 8-9. Power Domain Transition Command Register (PTCMD)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 8-14. Power Domain Transition Command Register (PTCMD) Field Descriptions**

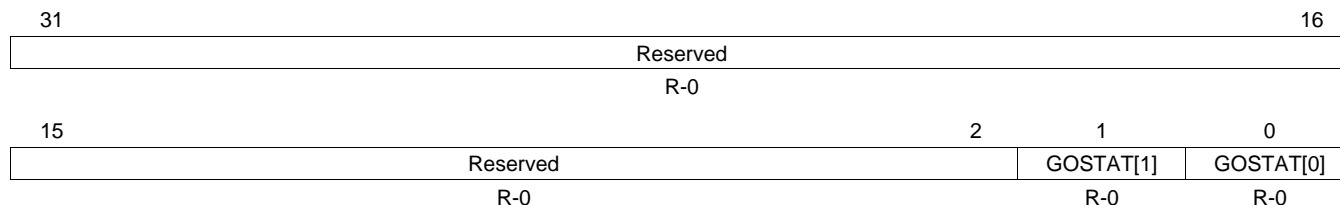
Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GO[1]	0 1	RAM/Pseudo (PD1) power domain GO transition command. A write of 0 has no effect. A write of 1 causes the PSC to evaluate all the NEXT fields relevant to this power domain (including PDCTL.NEXT for this domain, and MDCTL.NEXT for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (PDSTAT.STATE, MDSTAT.STATE), the PSC will transition those respective domain/modules to the new NEXT state.
0	GO[0]	0 1	Always ON (PD0) power domain GO transition command. A write of 0 has no effect. A write of 1 causes the PSC to evaluate all the NEXT fields relevant to this power domain (including MDCTL.NEXT for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (MDSTAT.STATE), the PSC will transition those respective domain/modules to the new NEXT state.



### 8.6.10 Power Domain Transition Status Register (PTSTAT)

The power domain transition status register (PTSTAT) is shown in [Figure 8-10](#) and described in [Table 8-15](#).

**Figure 8-10. Power Domain Transition Status Register (PTSTAT)**



LEGEND: R = Read only; -n = value after reset

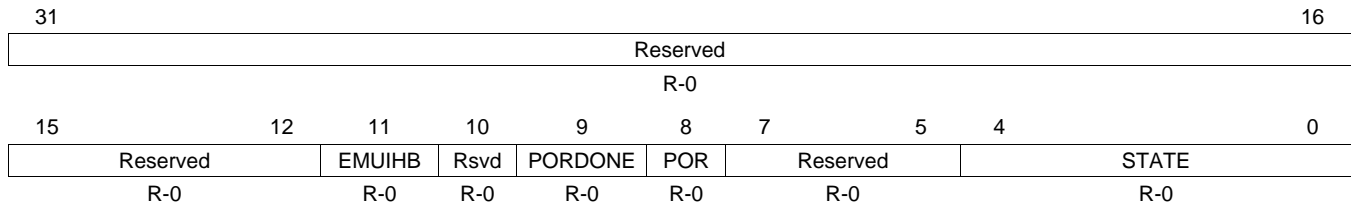
**Table 8-15. Power Domain Transition Status Register (PTSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GOSTAT[1]	0	RAM/Pseudo (PD1) power domain transition status. No transition in progress.
		1	RAM/Pseudo power domain is transitioning (that is, either the power domain is transitioning or modules in this power domain are transitioning).
0	GOSTAT[0]	0	Always ON (PD0) power domain transition status. No transition in progress.
		1	Modules in Always ON power domain are transitioning. Always On power domain is transitioning.

### 8.6.11 Power Domain 0 Status Register (PDSTAT0)

The power domain 0 status register (PDSTAT0) is shown in [Figure 8-11](#) and described in [Table 8-16](#).

**Figure 8-11. Power Domain 0 Status Register (PDSTAT0)**



LEGEND: R = Read only; -n = value after reset

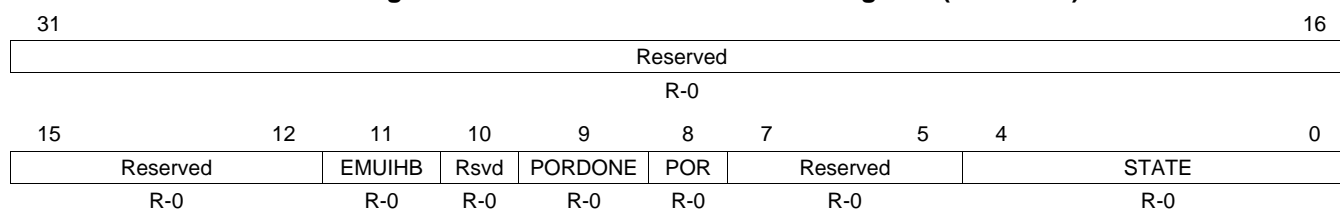
**Table 8-16. Power Domain 0 Status Register (PDSTAT0) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	EMUIHB	0	Emulation alters domain state. Interrupt is not active. No emulation altering user-desired power domain states.
		1	Interrupt is active. Emulation alters user-desired power domain state.
10	Reserved	0	Reserved
9	PORDONE	0	Power_On_Reset (POR) Done status Power domain POR is not done.
		1	Power domain POR is done.
8	POR	0	Power Domain Power_On_Reset (POR) status. This bit reflects the POR status for this power domain including all modules in the domain. Power domain POR is asserted.
		1	Power domain POR is de-asserted.
7-5	Reserved	0	Reserved
4-0	STATE	0-1Fh	Power Domain Status.
		0	Power domain is in the off state.
		1h	Power domain is in the on state.
		2h-Fh	Reserved
		10h-1Ah	Power domain is in transition.
		1Bh-1Fh	Reserved

### 8.6.12 Power Domain 1 Status Register (PDSTAT1)

The power domain 1 status register (PDSTAT1) is shown in [Figure 8-12](#) and described in [Table 8-17](#).

**Figure 8-12. Power Domain 1 Status Register (PDSTAT1)**



LEGEND: R = Read only; -n = value after reset

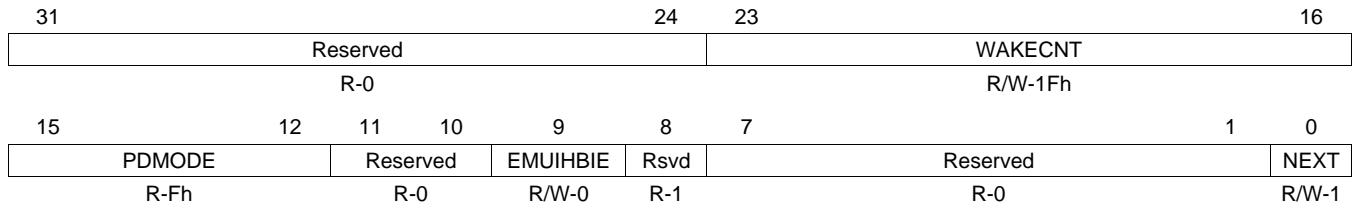
**Table 8-17. Power Domain 1 Status Register (PDSTAT1) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	EMUIHB	0	Emulation alters domain state. Interrupt is not active. No emulation altering user-desired power domain states.
		1	Interrupt is active. Emulation alters user-desired power domain state.
10	Reserved	0	Reserved
9	PORDONE	0	Power_On_Reset (POR) Done status Power domain POR is not done.
		1	Power domain POR is done.
8	POR	0	Power Domain Power_On_Reset (POR) status. This bit reflects the POR status for this power domain including all modules in the domain. Power domain POR is asserted.
		1	Power domain POR is de-asserted.
7-5	Reserved	0	Reserved
4-0	STATE	0-1Fh	Power Domain Status.
		0	Power domain is in the off state.
		1h	Power domain is in the on state.
		2h-Fh	Reserved
		10h-1Ah	Power domain is in transition.
		1Bh-1Fh	Reserved

### 8.6.13 Power Domain 0 Control Register (PDCTL0)

The power domain 0 control register (PDCTL0) is shown in [Figure 8-13](#) and described in [Table 8-18](#).

**Figure 8-13. Power Domain 0 Control Register (PDCTL0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

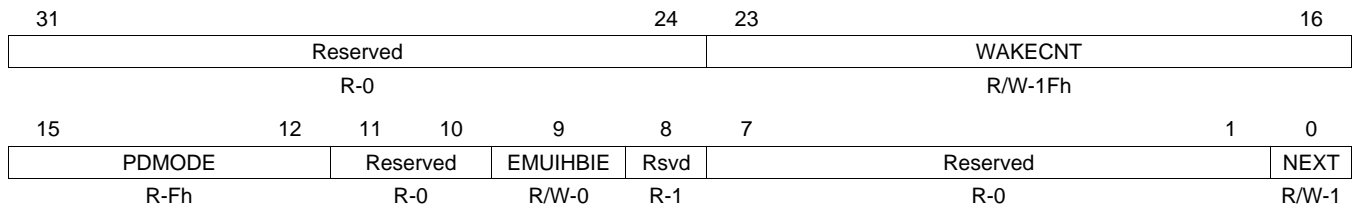
**Table 8-18. Power Domain 0 Control Register (PDCTL0) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	WAKECNT	0-FFh	RAM wake count delay value. Not recommended to change the default value (1Fh). Bits 23-30: GOOD2ACCESS wake delay. Bits 19-16: ON2GOOD wake delay.
15-12	PDMODE	0-Fh 0-Eh Fh	Power down mode. Reserved Core on, RAM array on, RAM periphery on.
11-10	Reserved	0	Reserved
9	EMUIHBIE	0 1	Emulation alters power domain state interrupt enable. Disable interrupt. Enable interrupt.
8	Reserved	1	Reserved
7-1	Reserved	0	Reserved
0	NEXT	0 1	Power domain next state. For Always ON power domain this bit is read/write, but writes have no effect since internally this power domain always remains in the on state. Power domain off. Power domain on.

### 8.6.14 Power Domain 1 Control Register (PDCTL1)

The power domain 1 control register (PDCTL1) is shown in [Figure 8-14](#) and described in [Table 8-19](#).

**Figure 8-14. Power Domain 1 Control Register (PDCTL1)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

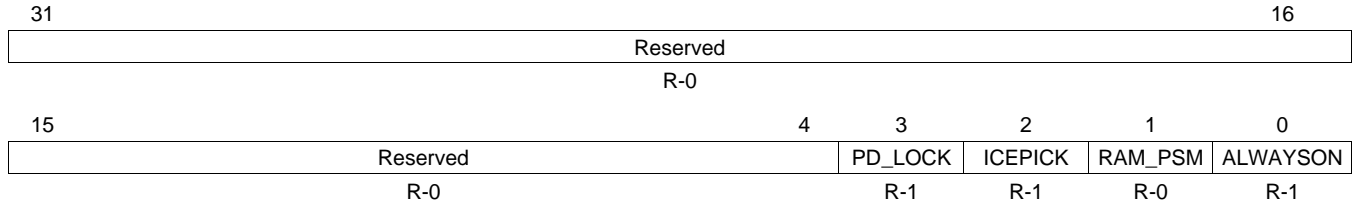
**Table 8-19. Power Domain 1 Control Register (PDCTL1) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	WAKECNT	0-FFh	RAM wake count delay value. Not recommended to change the default value (1Fh). Bits 23-30: GOOD2ACCESS wake delay. Bits 19-16: ON2GOOD wake delay.
15-12	PDMODE	0-Fh	Power down mode. 0 Core off, RAM array off, RAM periphery off. 1h Core off, RAM array retention, RAM periphery off (deep sleep). 2h-3h Reserved 4h Core retention, RAM array off, RAM periphery off. 5h Core retention, RAM array retention, RAM periphery off (deep sleep). 6h-7h Reserved 8h Core on, RAM array off, RAM periphery off. 9h Core on, RAM array retention, RAM periphery off (deep sleep). Ah Core on, RAM array retention, RAM periphery off (light sleep). Bh Core on, RAM array retention, RAM periphery on. Ch-Eh Reserved Fh Core on, RAM array on, RAM periphery on.
11-10	Reserved	0	Reserved
9	EMUIHBIE	0 1	Emulation alters power domain state interrupt enable. 0 Disable interrupt. 1 Enable interrupt.
8	Reserved	1	Reserved
7-1	Reserved	0	Reserved
0	NEXT	0 1	User-desired power domain next state. 0 Power domain off. 1 Power domain on.

### 8.6.15 Power Domain 0 Configuration Register (PDCFG0)

The power domain 0 configuration register (PDCFG0) is shown in [Figure 8-15](#) and described in [Table 8-20](#).

**Figure 8-15. Power Domain 0 Configuration Register (PDCFG0)**



LEGEND: R = Read only; -n = value after reset

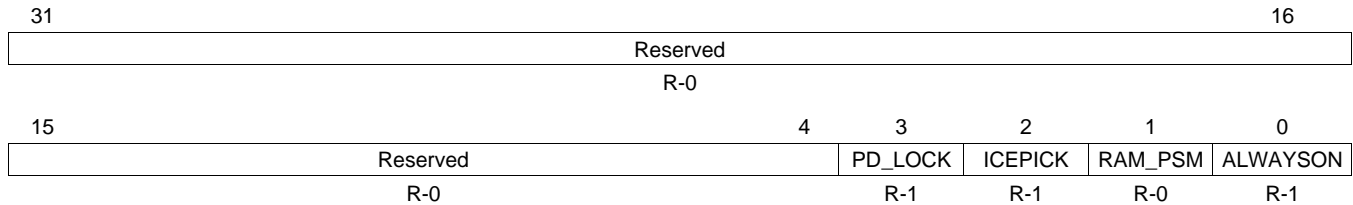
**Table 8-20. Power Domain 0 Configuration Register (PDCFG0) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	PD_LOCK	0	PDCTL.NEXT lock. For Always ON power domain this bit is a don't care.
		1	PDCTL.NEXT bit is locked and cannot be changed in software.
		1	PDCTL.NEXT bit is not locked.
2	ICEPICK	0	IcePick support.
		1	Not present
		1	Present
1	RAM_PSM	0	RAM power domain.
		1	Not a RAM power domain.
		1	RAM power domain.
0	ALWAYSON	0	Always ON power domain.
		1	Not an Always ON power domain.
		1	Always ON power domain.

### 8.6.16 Power Domain 1 Configuration Register (PDCFG1)

The power domain 1 configuration register (PDCFG1) is shown in [Figure 8-16](#) and described in [Table 8-21](#).

**Figure 8-16. Power Domain 1 Configuration Register (PDCFG1)**



LEGEND: R = Read only; -n = value after reset

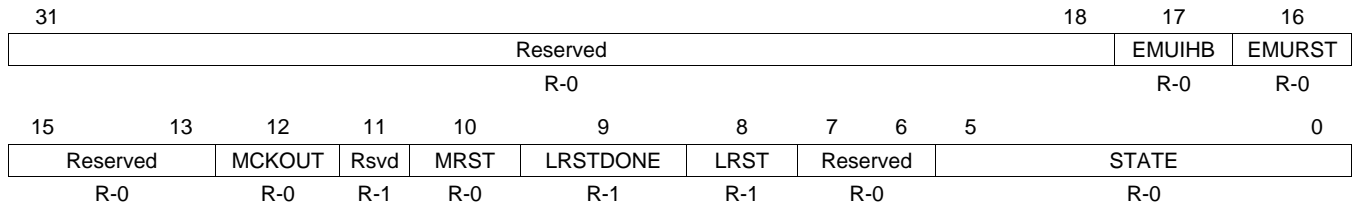
**Table 8-21. Power Domain 1 Configuration Register (PDCFG1) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	PD_LOCK	0	PDCTL.NEXT lock. For Always ON power domain this bit is a don't care.
		1	PDCTL.NEXT bit is locked and cannot be changed in software.
		1	PDCTL.NEXT bit is not locked.
2	ICEPICK	0	IcePick support.
		1	Not present
		1	Present
1	RAM_PSM	0	RAM power domain.
		1	Not a RAM power domain.
		1	RAM power domain.
0	ALWAYSON	0	Always ON power domain.
		1	Not an Always ON power domain.
		1	Always ON power domain.

### 8.6.17 Module Status *n* Register (MDSTAT<sub>*n*</sub>)

The module status *n* register (MDSTAT<sub>*n*</sub>) is shown in [Figure 8-17](#) and described in [Table 8-22](#).

**Figure 8-17. Module Status *n* Register (MDSTAT<sub>*n*</sub>)**



LEGEND: R = Read only; -*n* = value after reset

**Table 8-22. Module Status *n* Register (MDSTAT<sub>*n*</sub>) Field Descriptions**

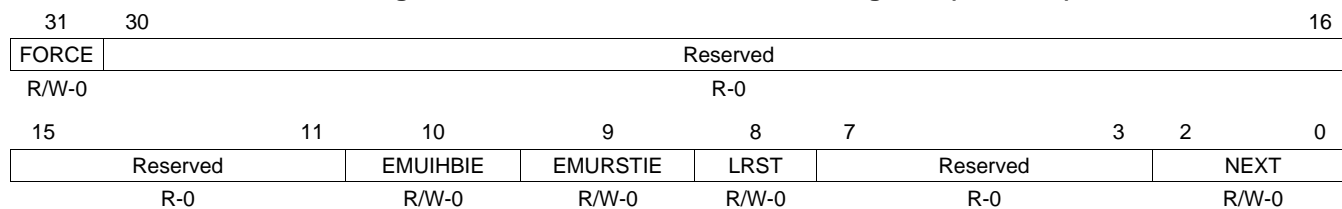
Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	EMUIHB	0 1	Emulation alters module state. This bit applies to DSP module (module 15). This field is 0 for all other modules. 0 No emulation altering user-desired module state programmed in the NEXT bit in the module control 15 register (MDCTL15). 1 Emulation altered user-desired state programmed in the NEXT bit in MDCTL15. If you desire to generate a PSCINT upon this event, you must set the EMUIHBIE bit in MDCTL15.
16	EMURST	0 1	Emulation alters module reset. This bit applies to DSP module (module 15). This field is 0 for all other modules. 0 No emulation altering user-desired module reset state. 1 Emulation altered user-desired module reset state. If you desire to generate a PSCINT upon this event, you must set the EMURSTIE bit in the module control 15 register (MDCTL15).
15-13	Reserved	0	Reserved
12	MCKOUT	0 1	Module clock output status. Shows status of module clock. 0 Module clock is off. 1 Module clock is on.
11	Reserved	1	Reserved
10	MRST	0 1	Module reset status. Reflects actual state of module reset. 0 Module reset is asserted. 1 Module reset is de-asserted.
9	LRSTDONE	0 1	Local reset done. Software is responsible for checking if local reset is done before accessing this module. This bit applies to DSP module (module 15). This field is 1 for all other modules. 0 Local reset is not done. 1 Local reset is done.
8	LRST	0 1	Module local reset status. This bit applies to DSP module (module 15). 0 Local reset is asserted. 1 Local reset is de-asserted.
7-6	Reserved	0	Reserved
5-0	STATE	0-3Fh 0 1h 2h 3h 4h-3Fh	Module state status: indicates current module status. 0 SwRstDisable state 1h SyncReset state 2h Disable state 3h Enable state 4h-3Fh Indicates transition



### 8.6.18 PSC0 Module Control $n$ Register (modules 0-15) (MDCTL $n$ )

The PSC0 module control  $n$  register (MDCTL $n$ ) is shown in [Figure 8-18](#) and described in [Table 8-23](#).

**Figure 8-18. PSC0 Module Control  $n$  Register (MDCTL $n$ )**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

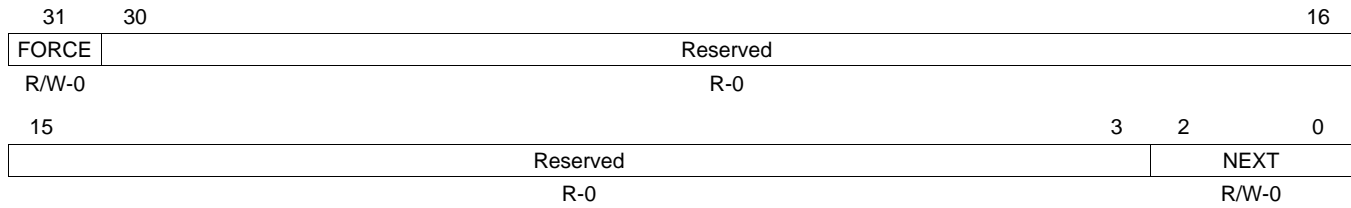
**Table 8-23. PSC0 Module Control  $n$  Register (MDCTL $n$ ) Field Descriptions**

Bit	Field	Value	Description
31	FORCE	0 1	Force enable. This bit forces the module state programmed in the NEXT bit in the module control 15 register (MDCTL15), ignoring and bypassing all the clock stop request handshakes managed by the PSC to change the state of the clocks to the module.  Note: It is <b>not</b> recommended to use the FORCE bit to disable the module clock, unless specified.  Force is disabled. Force is enabled.
30-11	Reserved	0	Reserved
10	EMUIHBIE	0 1	Interrupt enable for emulation alters module state. This bit applies to DSP module (module 15).  Disable interrupt. Enable interrupt.
9	EMURSTIE	0 1	Interrupt enable for emulation alters reset. This bit applies to DSP module (module 15).  Disable interrupt. Enable interrupt.
8	LRST	0 1	Module local reset control. This bit applies to DSP module (module 15).  Assert local reset De-assert local reset
7-3	Reserved	0	Reserved
2-0	NEXT	0-3h 0 1h 2h 3h	Module next state. SwRstDisable state SyncReset state Disable state Enable state

### 8.6.19 PSC1 Module Control *n* Register (modules 0-31) (MDCTL*n*)

The PSC1 module control *n* register (MDCTL*n*) is shown in [Figure 8-19](#) and described in [Table 8-24](#).

**Figure 8-19. PSC1 Module Control *n* Register (MDCTL*n*)**



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 8-24. PSC1 Module Control *n* Register (MDCTL*n*) Field Descriptions**

Bit	Field	Value	Description
31	FORCE	0 1	Force enable. This bit forces the module state programmed in the NEXT bit in the module control 15 register (MDCTL15), ignoring and bypassing all the clock stop request handshakes managed by the PSC to change the state of the clocks to the module.  Note: It is <b>not</b> recommended to use the FORCE bit to disable the module clock, unless specified.  Force is disabled. Force is enabled.
30-3	Reserved	0	Reserved
2-0	NEXT	0-3h 0 1h 2h 3h	Module next state. SwRstDisable state SyncReset state Disable state Enable state

## *Power Management*

---



---



---

Topic	Page
9.1 Introduction .....	153
9.2 Power Consumption Overview .....	153
9.3 PSC and PLLC Overview .....	153
9.4 Features .....	154
9.5 Clock Management .....	155
9.6 DSP Sleep Mode Management.....	156
9.7 RTC-Only Mode .....	156
9.8 Dynamic Voltage and Frequency Scaling (DVFS).....	156
9.9 Deep Sleep Mode.....	158
9.10 Additional Peripheral Power Management Considerations.....	161

## 9.1 Introduction

Power management is an important aspect for most embedded applications. For several applications and target markets, there may be a specific power budget and requirements to minimize power consumption for both power supply sizing and battery life considerations. Additionally, lower power consumption results in more optimal and efficient designs from cost, design, and energy perspectives. This device has several means of managing the power consumption. This chapter discusses the various power management features.

## 9.2 Power Consumption Overview

Power consumed by semiconductor devices has two components: dynamic and static. This can be shown as:

$$P_{total} = P_{dynamic} + P_{static}$$

The dynamic power is the power consumed to perform work when the device is in active modes (clocks applied, busses, and I/O switching), that is, analog circuits changing states. The dynamic power is defined by:

$$P_{dynamic} = \text{Capacitance} \times \text{Voltage}^2 \times \text{Frequency}$$

From the above formula, the dynamic power scales with the clock frequency (device/module frequency for core operations and switching frequency for I/O). Dynamic power can be reduced by controlling the clocks in such a way as to either operate at a clock setting just high enough to complete the required operation in the required timeline or to run at a clock setting until the work is complete and then drastically reduce the clock frequency or cut off the clocks until additional work must be performed.

In the formula, the dynamic power varies with the voltage squared, so the voltage of operations has significant impact on overall power consumption and, thus, on the battery life. Dynamic power can be reduced by scaling the operating voltage, when the performance requirements are not that high and the device can be operated at a corresponding lower frequency.

The capacitance is the capacitance of the switching nodes, or the load capacitances on the switching I/O pins.

The static power, as the name suggests, is independent of the switching frequency of the logic. It can be shown as:

$$P_{static} = f_{(leakage\ current)}$$

It is essentially a function of the “leakage”, or the power consumed by the logic when it is not switching or is not performing any work. Leakage current is dependent mostly on the manufacturing process used, the size of the die, etc. Leakage current is unavoidable while power is applied and scales roughly with the operating junction temperatures. Leakage power can only be avoided by removing power completely from a device or subsystem. The static power consumption plays a significant role in the Standby Modes (when the application is not running and in a dormant state) and plays an important role in the battery life for portable applications, etc.

## 9.3 PSC and PLLC Overview

The power and sleep controller (PSC) module plays an important role in managing the enabling/disabling of the clocks to the core and various peripheral modules. The PSC provides a granular support to turn on/off clocks on a module by module basis. Similarly, the two PLL controllers (PLL0 and PLL1) play an important role in device and module clock generation, and manage the frequency scaling operations for the device. Together these modules play a significant role in managing the clocks from a power management feature standpoint. For detailed information on the PSC, see the *Power and Sleep Controller (PSC)* chapter. For detailed information on the PLL0 and PLL1, see the *Device Clocking* chapter and the *Phase-Locked Loop Controller (PLLC)* chapter.

## 9.4 Features

This device has several means of managing power consumption, as detailed in the subsequent sections. This device uses the state-of-the-art 65 nm process, which provides a good balance on power and performance, providing high-performance transistors with relatively less leakage current and, thereby, low standby-power consumption modes.

There are several features in design as well as user driven software control to reduce dynamic power consumption. The design features (not under user control) include a power optimized clock tree design to reduce overall clock tree power consumption and automatic clock gating in several modules when the logic in the modules is not active.

The on-chip power and sleep controller (PSC) module provides granular software controlled module level clock gating, which reduces both clock tree and module power by basically disabling the clocks when the modules are not being used. Clock management also allows you to slow down the clocks, to reduce the dynamic power.

Table 9-1 describes the power management features.

**Table 9-1. Power Management Features**

Power Management	Description	Features
<b>Clock Management</b>		
PLL bypass and power-down	Both PLLs can be powered-down and run in bypass mode when not in use.	Reduces the dynamic power consumption of the core.
Module clock ON	Module clocks can be turned on/off without requiring reconfiguring the registers.	Reduces the dynamic power consumption of the core and I/O (if any free running I/O clocks).
<b>Core Sleep Management</b>		
DSP subsystem sleep mode	The DSP CPU can be put in sleep (IDLE) mode.	Reduces the dynamic power consumption.
<b>Voltage Management</b>		
RTC-only mode	Allows removing power from all core and I/O supply and just have the real-time clock (RTC) running.	Reduces the dynamic and static power for standby modes that require only the RTC to be functional.
<b>Dynamic Voltage and Frequency Scaling</b>		
Dynamic Voltage and Frequency Scaling (DVFS)	The operating voltage and frequency of the device can be dynamically scaled to meet the requirements of the application.	Reduces the dynamic power consumption of the core and I/O as well as standby power
<b>System/Device Sleep Management</b>		
Deep Sleep Mode	All internal clocks of the device can be turned on/off at the OSCIN level. The deep sleep function can be controlled externally through the DEESLEEP pin or internally through the RTC_ALARM pin.	Reduces the dynamic power consumption of the core and I/O.
<b>Peripheral I/O Power Management</b>		
DDR2/mDDR self-refresh mode	Allows memory to retain its contents while the rest of the system is powered down.	mDDR and DDR2 can be clock gated to reduce the dynamic power consumption or the entire device can be powered down to reduce the static power consumption.
LVC MOS I/O buffer receiver disable	LVC MOS I/O buffer receivers are disabled.	Minimizes the I/O power consumption.
Internal pull-up and pull-down resistor control	The internal pull-ups and pull-downs are enabled/disabled by groups.	Reduces the I/O leakage power.

## 9.5 Clock Management

### 9.5.1 Module Clock ON/OFF

The module clock on/off feature allows software to disable clocks to module individually, in order to reduce the module's dynamic/switching power consumption down to zero. This device is designed in full static CMOS; thus, when a module clock stops, the module's state is preserved and retained. When the clock is restarted, the module resumes operating from the stopping point.

---

**NOTE:** Stopping clocks to a module only affects dynamic power consumption, it does not affect static power consumption of the module or the device.

---

The power and sleep controller (PSC) module controls module clock gating. If a module's clock(s) is stopped while being accessed, the access may not occur, and it can potentially result in unexpected behavior. The PSC provides some protection against such erroneous conditions by monitoring the internal bus activity to ensure there are no accesses to the module from the internal bus, before allowing module's internal clock to be gated. However, it is still recommended that software must ensure that all of the transactions to the module are finished prior to disabling the clocks.

The procedure to turn module clocks on/off using the PSC is described in the *Power and Sleep Controller (PSC)* chapter.

---

**NOTE:** To preserve the state of the module, the module state in the PSC must be set to Disable. In this state, the module reset is not asserted and only the module clock is turned off.

---

Furthermore, special consideration must be given to DSP clock on/off. The procedure to turn the core clock on/off is further described in .

Additionally some peripherals implement additional power saving features by automatically shutting of clock to components within the module, when the logic is not active. This is transparent to you, but reduces overall dynamic power consumption when modules are not active.

### 9.5.2 Module Clock Frequency Scaling

Module clock frequency is scalable by programming the PLL multiply and divide parameters. Additionally, some modules might also have internal clock dividers. Reducing the clock frequency reduces the dynamic/switching power consumption, which scales linearly with frequency.

The *Device Clocking* chapter details the clocking structure of the device. The *Phase-Locked Loop Controller (PLL)* chapter describes how to program the PLL0 and PLL1 frequency and the frequency constraints.

### 9.5.3 PLL Bypass and Power Down

You can bypass each PLL in this device. Bypassing the PLL sends a bypass clock instead of the PLL VCO output (PLLOUT) to the system clocks of the PLLC. For PLLC0, the bypass clock is selected from either the PLL reference clock (OSCIN) or PLL1\_SYSCLK3. For PLLC1, the bypass clock is always OSCIN. The OSCIN frequency is typically, at most, up to 50 MHz.

You can use the OSCIN bypass mode to reduce the core and module clock frequencies to very low maintenance levels without using the PLL during periods of very low system activity. This can lower the overall dynamic power consumption, which is linearly proportional to the frequency.

When the PLL controller is placed in bypass mode, the PLL retains its frequency lock. This allows you to switch between bypass mode and PLL mode without having to wait for the PLL to relock. However, keeping the PLL locked consumes power. You can also power-down the PLL when bypassing it to minimize the overall power consumed by the PLL module. The advantage of bypassing the PLL without powering it down is that you do not have to incur the PLL lock time when switching back to a normal operating level.

The *Device Clocking* chapter and the *Phase-Locked Loop Controller (PLL)* chapter describe PLL bypass and PLL power down.

## 9.6 DSP Sleep Mode Management

### 9.6.1 C674x DSP CPU Sleep Mode

The DSP CPU can be put in a low-power state by executing the IDLE instruction. For information on the IDLE instruction, see the *TMS320C674x DSP CPU and Instruction Set Reference Guide* ([SPRUFE8](#)).

### 9.6.2 C674x Megamodule Sleep Mode

The IDLE instruction is used as part of the procedure for shutting down the entire C674x megamodule, by the power-down controller (PDC) module. In shutting down the entire C674x megamodule, the PDC can internally clock gate off the following components of the megamodule and internal memories of the DSP subsystem:

- C674x CPU
- Level 1 Program Memory Controller (PMC)
- Level 1 Data Memory Controller (DMC)
- Level 2 Unified Memory Controller (UMC)
- Extended Memory Controller (EMC)
- L1P Memory
- L1D Memory
- L2 Memory

Putting the entire C674x megamodule into the low-power sleep mode is typically more useful and saves a lot more power, as compared to just executing the IDLE instruction to put only the CPU in idle mode.

For information on putting the C674x megamodule in the low-power mode using the PDC, see the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

## 9.7 RTC-Only Mode

In real-time clock (RTC)-only mode, the RTC is powered on and the rest of the device is completely powered off (all supplies except the RTC supply are removed). In this mode, the RTC is fully functional and keeps track of date, hours, minutes, and seconds. In this mode, the overall power consumption would be significantly lower, as voltage from the rest of the core and I/O logic can be completely removed, eliminating most of the active and static power of the device, except for what is consumed by the RTC module, running at 32 kHz.

---

**NOTE:** To put the device in RTC-only mode, there is no software control sequence. You can put the device in the RTC-only mode by removing the power supply from all core and I/O logic, except for the RTC core logic supply (RTC\_CVDD). During wake up, all power sequencing requirements described in the device-specific data manual must be followed.

---

Some limitations apply in the RTC-only mode. First, the RTC\_ALARM pin is not available as an option for use as a control to signal an external power supply to reapply power to the rest of the device. This is because the RTC\_ALARM pin is powered by the I/O supply that is powered down in RTC-only mode. Second, in RTC-only mode, only the RTC register contents are preserved, all other internal memory and register contents are lost. Mobile DDR and DDR2 contents can be preserved through the use of self-refresh (see [Section 9.9.2](#)). However, software must be in place to restore the context of the device, for example, reinitialize internal registers, setup cache memory configurations, interrupt vectors, etc.

## 9.8 Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic voltage and frequency scaling (DVFS) consists of minimizing the idle time of the system. The DVFS technique uses dynamic selection of the optimal frequency and voltage to allow a task to be performed in the required amount of time. This reduces the total power consumption of the device while still meeting task requirements. DVFS requires control over the clock frequency and the operating voltage of the device elements. By intelligently switching these elements to their optimal operating points, it is possible to minimize the power consumption of the device for a given task.



For reasons related to the device (clock architecture, process, etc.), DVFS is used only for a few discrete steps, not over a continuum of voltage and frequency values. Each step, or operating performance point (OPP), is composed of a voltage and frequency pair. For an OPP, the frequency corresponds to the maximum frequency allowed at a voltage, or reciprocally; the voltage corresponds to the minimum voltage allowed for a frequency. See your device data manual for a list of the OPPs supported by the device.

When applying DVFS, a processor or system always runs at the lowest OPP that meets the performance requirement at a given time. You determine the optimal OPP for a given task and then switch to that OPP to save power.

### 9.8.1 Frequency Scaling Considerations

The operating frequency of the device is controlled through its two PLL controllers (PLLC0 and PLLC1). Through a series of multipliers and dividers you can change the frequencies of various clocks throughout the device. See the *Device Clocking* chapter for information on the clock architecture of the device and see the *Phase-Locked Loop Controller (PLL)* chapter for information on the PLL controllers. A few things must be noted when changing the various internal frequencies of the device:

- Changing the SYSCLK frequency

The PLL\_VCO (PLLOUT) frequency can be programmed through a PLL multiplier. A series of dividers divide PLLOUT to generate the various device SYSCLKs.

To change the SYSCLK frequency you can change the PLL multiplier or you can change the SYSCLK divider ratio. When changing the PLL multiplier, you must put the PLL controller in bypass mode while the PLL multiplier value is modified and a lock on the new frequency is reached. The lock time is given in the device data manual. When changing the divider ratios it is not required to put the PLL controller in bypass mode.

Changing the SYSCLK frequency through the dividers is faster as there is no need to reprogram the PLL. However, the SYSCLK frequency will depend solely on the divider ratios used.

- SYSCLK domain fixed ratios

Certain SYSCLK domains need to operate at a fixed ratio with respect to the CPU clock. Care should be taken to ensure that these fixed ratios are maintained. For additional details, see the *Device Clocking* chapter.

- PLLC0 bypass clock

When switching the PLL multiplier, the PLL controller must be placed in bypass mode. Bypassing the PLL sends a bypass clock instead of the PLL VCO output (PLLOUT) to the system clock dividers of the PLL controller.

For PLLC0 the bypass clock is selected from either the PLL reference clock (OSCIN) or PLL1\_SYSCLK3. For PLLC1, the bypass clock is always OSCIN. The OSCIN frequency is typically, at most, up to 50 MHz.

You can use the OSCIN bypass mode to reduce the core and module clock frequencies to very low maintenance levels without using the PLL during periods of very low system activity.

It may be desirable for the bypass clock to not revert to OSCIN in some situations to preserve bandwidth during frequency scaling transitions. For this reason, the PLLC0 bypass clock can be set to PLL1\_SYSCLK3. This selection is made through the EXTCLKSRC bit in the PLLCTL register of PLLC0.

- Peripheral immunity from CPU clock frequency changes

Peripherals that are clocked by the PLL0\_AUXCLK are immune to changes in the PLL0 frequency. The PLL0\_AUXCLK is derived from OSCIN.

Peripherals in the ASYNC3 domain are clocked off from either PLL1\_SYSCLK2 or PLL0\_SYSCLK2. Furthermore, PLL0\_SYSCLK2 must always be /2 of the CPU clock frequency. To keep these peripherals immune from changes in PLL0 frequency (such as when the CPU frequency is modified), you can configure the ASYNC3 domain to be clocked from PLL1\_SYSCLK2. PLL1 is mainly used to clock the DDR2/mDDR memory controller.

When peripherals are immune to changes in the CPU clock frequency, their internal clock dividers do not have to be adjusted for changes in their input clock frequencies.



## 9.8.2 Voltage Scaling Considerations

The operating voltage of the device must be totally controlled through mechanisms outside the device. I<sup>2</sup>C ports on the device can be used to communicate with external power management chips. A few things must be noted when changing the operating voltage of the device:

- Voltage ramp rate: The ramp rate of the operating voltage must be observed during operating performance point (OPP) transitions. See the device data manual for ramp rate specifications.
- Switching to a lower voltage: When switching to a lower voltage, the maximum operating frequency changes. Care must be taken such that the maximum operating frequency supported at the new voltage is not violated. For this reason, it is recommended to change the operating frequency before switching the operating voltage.

## 9.9 Deep Sleep Mode

This device supports a Deep Sleep mode where all device clocks are stopped and the on-chip oscillator is shut down to save power. Registers and memory contents are preserved, thus, upon recovery, the program may continue from where it left off with minimal overhead involved.

The Deep Sleep mode is initiated when the  $\overline{\text{DEEPSLEEP}}$  pin is driven low. The device wakes up from Deep Sleep mode when the  $\overline{\text{DEEPSLEEP}}$  pin is driven high. The  $\overline{\text{DEEPSLEEP}}$  pin can be driven by an external controller or it can be driven internally by the real-time clock (RTC). The RTC method allows for automatic wake-up at a programmed time.

---

**NOTE:** Due to pin multiplexing, the  $\overline{\text{DEEPSLEEP}}$  pin can only be driven by an external controller or its internal real-time clock (RTC). The  $\overline{\text{DEEPSLEEP}}$  pin cannot be driven by both an external controller and its internal real-time clock at the same time.

---

### 9.9.1 Entering/Exiting Deep Sleep Mode Using Externally Controlled Wake-Up

#### 9.9.1.1 Entering Deep Sleep Mode

Use the following procedure to enter the Deep Sleep mode if an external signal is used to wake-up the device:

1. To preserve DDR2/mDDR memory contents, activate the self-refresh mode and gate the clocks to the DDR2/mDDR memory controller. You can use partial array self-refresh (PASR) for additional power savings for mDDR memory.
2. PLL/PLLC0 and PLL/PLLC1 should be placed in bypass mode (clear the PLEN bit in the PLL control register (PLLCTL) of each PLLC to 0).
3. PLL/PLLC0 and PLL/PLLC1 should be powered down (set the PLLPWRDN bit in PLLCTL of each PLLC to 1).
4. Configure the  $\overline{\text{DEEPSLEEP}}$  pin as input-only using the PINMUX0\_31\_28 bits in the PINMUX0 register in the *System Configuration (SYSCFG) Module* chapter.
5. The external controller should drive the  $\overline{\text{DEEPSLEEP}}$  pin high (not in Deep Sleep).
6. Configure the desired delay in the SLEEP\_COUNT bit field in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter. This count determines the delay before the Deep Sleep logic releases the clocks to the device during wake up (allowing the oscillator to stabilize).
7. Set the SLEEPENABLE bit in DEEPSLEEP to 1. This automatically clears the SLEEPCOMPLETE bit.
8. Begin polling the SLEEPCOMPLETE bit until it is set to 1. This bit is set once the device is woken up from Deep Sleep mode.
9. The external controller drives the  $\overline{\text{DEEPSLEEP}}$  pin low to initiate Deep Sleep mode.

For more details on the clock stop procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

### 9.9.1.2 Exiting Deep Sleep Mode

Use the following procedure to exit the Deep Sleep state if an external signal is used to wake-up the device:

1. The external controller drives the  $\overline{\text{DEEPSLEEP}}$  pin high.
2. When the SLEEPDELAY delay is complete, the Deep Sleep logic releases the clock to the device and sets the SLEEPCOMPLETE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
3. Clear the SLEEPENABLE bit in DEEPSLEEP to 0. This automatically clears the SLEEPCOMPLETE bit.
4. Initialize the PLL controllers as described in [Section 7.2.2.2](#). Note that the state of the PLL controller registers is preserved during Deep Sleep mode. Therefore, it is not necessary to reprogram all the PLL controller registers unless a new setting is desired. At minimum, steps 3, 4, and 7-10 of the PLL initialization procedure must be followed.
5. Enable the clocks to the DDR2/mDDR memory controller, reset the DDR PHY, and then take the DDR2/mDDR out of self-refresh mode.
6. Configure the desired states to the peripherals and enable as required.

For more details on the clock enable procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

## 9.9.2 Entering/Exiting Deep Sleep Mode Using RTC Controlled Wake-Up

### 9.9.2.1 Entering Deep Sleep Mode

Use the following procedure to enter the Deep Sleep state if the RTC is used to wake-up the device:

1. To preserve DDR2/mDDR memory contents, activate the self-refresh mode and gate the clocks to the DDR2/mDDR memory controller. You can use partial array self-refresh (PASR) for additional power savings for mDDR memory.
2. PLL/PLLC0 and PLL/PLLC1 should be placed in bypass mode (clear the PLEN bit in the PLL control register (PLLCTL) of each PLLC to 0).
3. PLL/PLLC0 and PLL/PLLC1 should be powered down (set the PLLPWRDN bit in PLLCTL of each PLLC to 1).
4. Configure the desired wake-up time as an alarm in the RTC.
5. Configure the  $\overline{\text{DEEPSLEEP}}/\text{RTC\_ALARM}$  pin to output RTC\_ALARM using the PINMUX0\_31\_28 bits in the PINMUX0 register in the *System Configuration (SYSCFG) Module* chapter. The pin is driven low since the alarm has not yet occurred.
6. Configure the desired delay in the SLEEPDELAY bit field in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter. This count determines the delay before the Deep Sleep logic releases the clocks to the device during wake up (allowing the oscillator to stabilize).
7. Set the SLEEPENABLE bit in DEEPSLEEP to 1. This automatically clears the SLEEPCOMPLETE bit. Also, the device now enters the Deep Sleep mode since the  $\overline{\text{DEEPSLEEP}}$  pin is low.

For more details on the clock stop procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

### 9.9.2.2 Exiting Deep Sleep Mode

Use the following procedure to exit the Deep Sleep state if the RTC is used to wake-up the device:

1. The RTC alarm occurs and the RTC\_ALARM pin is driven high (which is internally connected to the  $\overline{\text{DEEPSLEEP}}$  pin). This causes the Deep Sleep logic to exit the Deep Sleep mode.
2. When the SLEEPDELAY delay is complete, the Deep Sleep logic releases the clock to the device and sets the SLEEPCOMPLETE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
3. Clear the SLEEPENABLE bit in DEEPSLEEP to 0. This automatically clears the SLEEPCOMPLETE bit.

4. Initialize the PLL controllers as described in [Section 7.2.2.2](#). Note that the state of the PLL controller registers is preserved during Deep Sleep mode. Therefore, it is not necessary to reprogram all the PLL controller registers unless a new setting is desired. At minimum, steps 3, 4, and 7-10 of the PLL initialization procedure must be followed.
5. Enable the clocks to the DDR2/mDDR memory controller, reset the DDR PHY, and then take the DDR2/mDDR out of self-refresh mode.
6. Configure the desired states to the peripherals and enable as required.

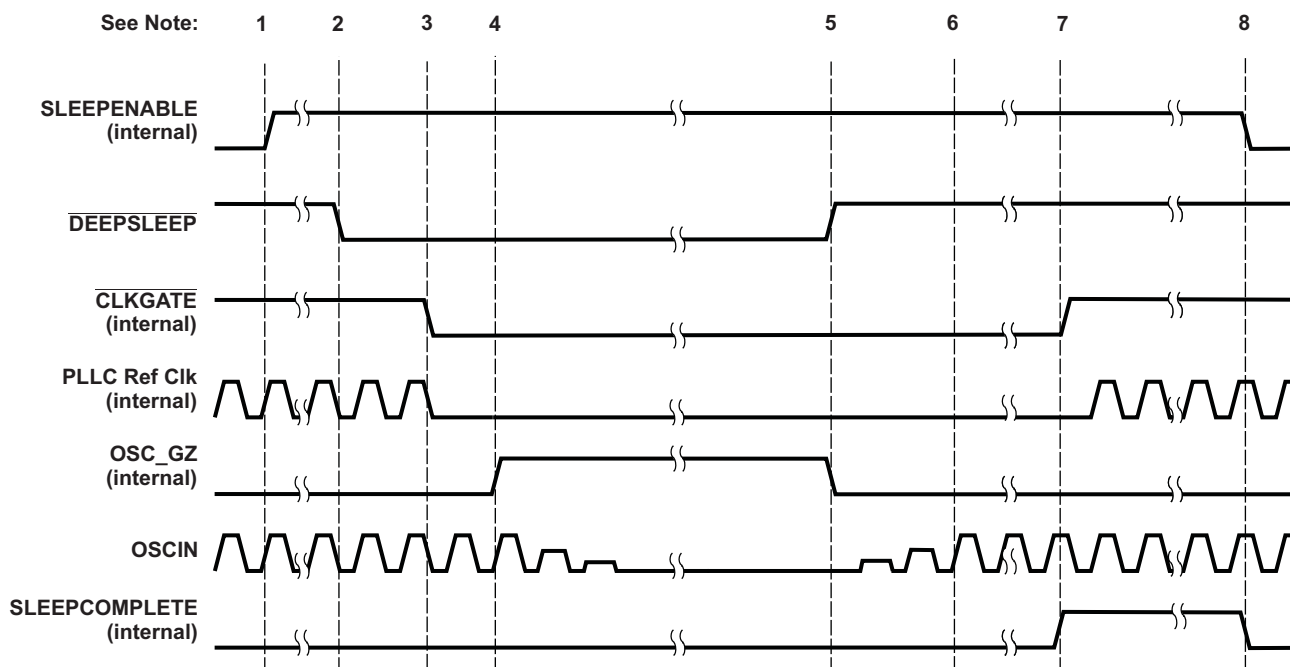
For more details on the clock enable procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

### 9.9.3 Deep Sleep Sequence

Figure 9-1 illustrates the Deep Sleep sequence:

1. Software sets the SLEEPENABLE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
2. The  $\overline{\text{DEEPSLEEP}}$  pin is driven low by either an external device or the RTC\_ALARM pin. The Deep Sleep mode begins.
3. The PLL controller reference clock is gated.
4. The on-chip oscillator is disabled. If the device is being clocked by an external source, this clock may stay enabled; the power savings from turning off this clock is minimal.
5. The  $\overline{\text{DEEPSLEEP}}$  pin is driven high and the on-chip oscillator is enabled.
6. The Deep Sleep counter begins counting valid clock cycles.
7. The count has reached the number specified in the SLEEP\_COUNT bit field and the SLEEP\_COMPLETE bit is set. The PLL reference clock is enabled and the Deep Sleep mode ends.
8. Software clears the SLEEPENABLE bit. The SLEEP\_COMPLETE bit is automatically cleared.

**Figure 9-1. Deep Sleep Mode Sequence**



## 9.9.4 Entering/Exiting Deep Sleep Mode Using Software Handshaking

Entering the Deep Sleep mode stops all of the clocks to the device so it is the responsibility of the software to ensure that all peripheral accesses have been completed and peripheral interfaces appropriately configured for clocks to stop. Therefore, before an external controller drives the DEEPSLEEP pin, a handshaking mechanism must be in place to give software time to prepare the device for Deep Sleep mode. The implementation of the handshake mechanism is up to the system designer.

### 9.9.4.1 Entering Deep Sleep Mode

The following example sequence can be used to activate the Deep Sleep mode using a handshaking mechanism between your device and an external device:

1. Clear the SLEEPENABLE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter to 0. The DEEPSLEEP pin has no effect until software running on the device sets this bit.
2. Configure the GP0[8]/DEEPSLEEP/RTC\_ALARM pin to output GP0[8] using the PINMUX0\_31\_28 bits in the PINMUX0 register in the *System Configuration (SYSCFG) Module* chapter. When the pin is configured for GPIO functionality, the internal DEEPSLEEP signal is still driven by the value on the pin.
3. Configure the GP0[8] pin to generate interrupts on the falling edge of the GPIO signal.
4. An external device drives the GP0[8] pin low.
5. Software prepares the device for Deep Sleep mode.
6. Set the SLEEPENABLE bit in DEEPSLEEP to 1. The Deep Sleep mode is immediately started and all device clocks are stopped. Also, the SLEEPCOMPLETE bit is automatically cleared.

### 9.9.4.2 Exiting Deep Sleep Mode

To exit the Deep Sleep mode, follow this sequence:

1. An external device drives the GP0[8] pin high.
2. The device exits the Deep Sleep mode. When the SLEEPDELAY delay is complete, the Deep Sleep logic releases the clock to the device and sets the SLEEPCOMPLETE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
3. Clear the SLEEPENABLE bit in DEEPSLEEP to 0.

## 9.10 Additional Peripheral Power Management Considerations

This section lists additional power management features and considerations that might be part of other chip-level or peripheral logic, apart from the features supported by the core, PLL controller (PLL), and power and sleep controller (PSC).

### 9.10.1 DDR2/mDDR Memory Controller Clock Gating and Self-Refresh Mode

The DDR2/mDDR memory controller supports different methods for reducing its power consumption including self-refresh mode, power-down mode, and clock gating. Additionally, the DDR2/mDDR memory controller DLL, PHY, and the receivers at the I/O pins can be disabled. Even if the PHY is active, the receivers can be configured to disable whenever writes are in progress and the receivers are not needed.

Self-refresh mode can be used to preserve the contents of DDR2/mDDR memory when the DDR2/mDDR memory controller is clock gated or when the device is placed in RTC-only mode. However, in the RTC-only mode, care must be taken to correctly take the DDR2/mDDR out of self-refresh mode.

---

**NOTE:** To preserve the contents of the external memory while the DDR2/mDDR memory controller is clock gated, its self-refresh mode must be enabled before the DDR2/mDDR memory controller clock is turned off.

---

In RTC-only mode, all portions of the device except for the RTC are powered down, including the DDR2/mDDR memory controller. During power-up, the DDR2/mDDR memory controller defaults to its reset state. When the DDR2/mDDR memory controller is taken out of reset, it automatically runs its memory initialization routine; the self-refresh state of the memory is ignored. This hardware sequence cannot be stopped by software running on the device.

To correctly take the memory out of self-refresh after coming back from RTC-only mode, follow these steps:

1. Before going into RTC-only mode, disconnect the DDR2/mDDR memory controller CKE output pin from the memory; ensure the memory's CKE input pin continues to be driven low.
2. After coming back from RTC-only mode, configure the device to the desired operating state.
3. Program the DDR2/mDDR memory controller following the normal sequence.
4. Enable the self-refresh mode of the DDR2/mDDR memory controller.
5. Connect the DDR2/mDDR memory controller CKE output pin to the memory.
6. Disable the self-refresh mode of the DDR2/mDDR memory controller.

After this sequence, the DDR2/mDDR memory controller is ready for use. Note that hardware logic is needed to disconnect the CKE output pin from the memory and to drive the memory's CKE input pin low.

For more details on the power management features of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

### 9.10.2 LVC MOS I/O Buffer Receiver Disable

This device supports two types of LVC MOS I/Os: 1.8V I/Os and low-static current dual-voltage I/Os that operate at either 1.8V or 3.3V. The receivers on the LVC MOS I/Os are enabled and disabled by software (see the RXACTIVE Control Register (RXACTIVE) in the *System Configuration (SYSCFG) Module* chapter). In the event that certain receivers are not used (such as in a low-power state), they can be disabled to conserve power.

### 9.10.3 Pull-Up/Pull-Down Disable

In general, you must ensure that all input pins are always pulled to a logic-high or a logic-low voltage level. A floating input pin can consume a small amount of I/O leakage current. The I/O leakage current can be greatly multiplied in the case of several floating inputs pins.

This device includes internal pull-up and pull-down resistors that prevent floating input pins. These internal resistors are generally very weak and their use is intended for pins that are not connected on the board design. For pins that are connected, external pull-up and pull-down resistors are recommended.

When an input pin is externally driven to a valid logic level, through an external pull-up resistor or by an external device for example, it is recommended to disable the internal resistor. Opposing an internal pull-up or pull-down resistor can consume a small amount of current. Internal resistors are disabled through the pullup/pulldown enable register (PUPD\_ENA) in the *System Configuration (SYSCFG) Module* chapter.

---

---

## System Configuration (SYSCFG) Module

---

---

Topic	Page
10.1 Introduction .....	164
10.2 Protection .....	164
10.3 Master Priority Control .....	165
10.4 Interrupt Support .....	166
10.5 SYSCFG Registers .....	167

## 10.1 Introduction

The system configuration (SYSCFG) module is a system-level module containing status and top level control logic required by the device. The system configuration module consists of a set of memory-mapped status and control registers, accessible by the CPU, supporting all of the following system features, and miscellaneous functions and operations.

- Device Identification
- Device Configuration
  - Pin multiplexing control
  - Device Boot Configuration Status
- Master Priority Control
  - Controls the system priority for all master peripherals (including EDMA3TC)
- Emulation Control
  - Emulation suspend control for peripherals that support the feature
- Special Peripheral Status and Control
  - Locking of PLL control settings
  - Default burst size configuration for EDMA3 transfer controllers
  - Event source selection for the eCAP peripheral input capture
  - McASP0 AMUTEIN selection and clearing of AMUTE
  - Clock source selection for EMIFA and DDR2/mDDR
  - HPI Control

The system configuration module controls several global operations of the device; therefore, the module supports protection against erroneous and illegal accesses to the registers in its memory-map. The protection mechanisms that are present in the module are:

- A special key sequence that needs to be written into a set of registers in the system configuration module, to allow write ability to the rest of registers in the system configuration module.
- Several registers in the module are only accessible when the CPU requesting read/write access is in privileged mode.

## 10.2 Protection

The SYSCFG module controls several global operations of the device; therefore, it has a protection mechanism that prevents spurious and illegal accesses to the registers in its memory map. The protection mechanism enables accesses to these registers only if certain conditions are met.

### 10.2.1 Privilege Mode Protection

The CPU supports two privilege levels: Supervisor and User. Several registers in the SYSCFG memory-map can only be accessed when the accessing host (CPU or master peripheral) is operating in privileged mode, that is, in Supervisor mode. The registers that can only be accessed in privileged mode are listed in [Section 10.5](#). See the *TMS320C674x DSP CPU and Instruction Set Reference Guide* ([SPRUFE8](#)) for details on privilege levels.



## 10.2.2 Kicker Mechanism Protection

---

**NOTE:** The Kick registers are disabled in silicon revision 2 and later. The SYSCFG registers are always unlocked and writes to the Kick registers have no functional effect.

The Kick registers (KICK0R and KICK1R) can only be accessed in privileged mode (the host needs to be in Supervisor mode). Any number of accesses may be performed to the SYSCFG module, while the module is unlocked.

The SYSCFG module remains unlocked after the unlock sequence, until locked again. Locking the module is accomplished by writing any value other than the key values to either KICK0R or KICK1R.

---

To access any registers in the SYSCFG module, it is required to follow a special sequence of writes to the Kick registers (KICK0R and KICK1R) with correct key values. Writing the correct key value to the kick registers unlocks the registers in the SYSCFG memory-map. In order to access the SYSCFG registers, the following unlock sequence needs to be executed in software:

1. Write the key value of 83E7 0B13h to KICK0R.
2. Write the key value of 95A4 F1E0h to KICK1R.

After steps 1 and 2, the SYSCFG module registers are accessible and can be configured as per the application requirements.

## 10.3 Master Priority Control

The on-chip peripherals/modules are essentially divided into two broad categories, masters and slaves. The master peripherals are typically capable of initiating their own read/write data access requests, this includes the DSP, EDMA3 transfer controllers, and peripherals that do not rely on the CPU or EDMA3 for initiating the data transfer to/from them. In order to determine allowed connection between masters and slave, each master request source must have a unique master ID (mstid) associated with it. The master ID is shown in [Table 10-1](#). See the device-specific data manual to determine the masters present on your device.

Each switched central resource (SCR) performs prioritization based on priority level of the master that sends the read/write requests. For all peripherals/ports classified as masters on the device, the priority is programmed in the master priority registers (MSTPRI0-3) in the SYSCFG modules. The default priority levels for each bus master is shown in [Table 10-2](#). Application software is expected to modify these values to obtain the desired performance.



**Table 10-1. Master IDs**

Master ID	Peripheral
0-1	Reserved
2	DSP MDMA
3	DSP CFG
4-9	Reserved
10	EDMA3_0_CC0
11	EDMA3_1_CC0
12-15	Reserved
16	EDMA3_0_TC0 - read
17	EDMA3_0_TC0 - write
18	EDMA3_0_TC1 - read
19	EDMA3_0_TC1 - write
20	EDMA3_1_TC0 – read
21	EDMA3_1_TC0 – write
22-36	Reserved
37	HPI
38-255	Reserved

**Table 10-2. Default Master Priority**

Master	Default Priority <sup>(1)</sup>	Master Priority Register
EDMA3_0_TC0 <sup>(2)</sup>	0	MSTPRI1
EDMA3_0_TC1 <sup>(2)</sup>	0	MSTPRI1
DSP MDMA <sup>(3)</sup>	2	MSTPRI0
DSP CFG <sup>(3)</sup>	2	MSTPRI0
EDMA3_1_TC0 <sup>(2)</sup>	4	MSTPRI1
HPI	6	MSTPRI2

<sup>(1)</sup> The default priority settings might not be optimal for all applications. The master priority should be changed from default based on application specific requirement, in order to get optimal performance and prioritization for masters moving data that is real time sensitive.

<sup>(2)</sup> The priority for EDMA3\_0\_TC0, EDMA3\_0\_TC1, and EDMA3\_1\_TC0 is configurable through fields in the master priority 1 register (MSTPRI1), not the EDMA3CC QUEPRI register.

<sup>(3)</sup> The priority for DSP MDMA and DSP CFG is controlled by fields in the master priority 0 register (MSTPRI0) and not DSP.MDMAARBE.PRI (DSP Bandwidth manager module).

## 10.4 Interrupt Support

### 10.4.1 Interrupt Events and Requests

The SYSCFG module generates two interrupts: an address error interrupt (BOOTCFG\_ADDR\_ERR) and a protection interrupt (BOOTCFG\_PROT\_ERR). The BOOTCFG\_ADDR\_ERR is generated when there is an addressing violation due to an access to a non-existent location in the SYSCFG register space. The BOOTCFG\_PROT\_ERR interrupt is generated when there is a protection violation of either in the defined ranges or to the SYSCFG registers. It is required to write a value of 0 to the end of interrupt register (EOI) after the software has processed the SYSCFG interrupt, this acts as an acknowledgement of completion of the SYSCFG interrupt so that the module can reliably generate subsequent interrupts.

The transfer parameters that caused the violation are saved in the fault address register (FLTADDR) and the fault status register (FLTSTAT).

## 10.4.2 Interrupt Multiplexing

The interrupts from the SYSCFG module are combined with the interrupts from the MPU module into a single interrupt called MPU\_BOOTCFG\_ERR. The combined interrupt is routed to the DSP interrupt controller.

## 10.4.3 Host-DSP Communication Interrupts

The SYSCFG module also has a set of registers, the chip signal register (CHIPSIG) and the chip signal clear register (CHIPSIG\_CLR), to facilitate host-to-processor communication. This is generally used to allow an external host and the DSP to coordinate.

Either of the processors can set specific bits in this SYSCFG register, which in turn can interrupt the other processor, if the interrupts have been appropriately enabled in the processor's interrupt controller.

## 10.5 SYSCFG Registers

Table 10-3 lists the memory-mapped registers for the system configuration module 0 (SYSCFG0) and Table 10-4 lists the memory-mapped registers for the system configuration module 1 (SYSCFG1). These tables also indicate whether a particular register can be accessed only when the CPU is in privileged mode.

**Table 10-3. System Configuration Module 0 (SYSCFG0) Registers**

Address	Acronym	Register Description	Access	Section
01C1 4000h	REVID	Revision Identification Register	—	<a href="#">Section 10.5.1</a>
01C1 4008h	DIEIDR0 <sup>(1)</sup>	Die Identification Register 0	—	—
01C1 400Ch	DIEIDR1 <sup>(1)</sup>	Die Identification Register 1	—	—
01C1 4010h	DIEIDR2 <sup>(1)</sup>	Die Identification Register 2	—	—
01C1 4014h	DIEIDR3 <sup>(1)</sup>	Die Identification Register 3	—	—
01C1 4018h	DEVIDR0	Device Identification Register 0	Privileged mode	<a href="#">Section 10.5.2</a>
01C1 4020h	BOOTCFG	Boot Configuration Register	Privileged mode	<a href="#">Section 10.5.3</a>
01C1 4024h	CHIPREVIDR	Chip Revision Identification Register	Privileged mode	<a href="#">Section 10.5.4</a>
01C1 4038h	KICK0R	Kick 0 Register	Privileged mode	<a href="#">Section 10.5.5.1</a>
01C1 403Ch	KICK1R	Kick 1 Register	Privileged mode	<a href="#">Section 10.5.5.2</a>
01C1 4044h	HOST1CFG	Host 1 Configuration Register	—	<a href="#">Section 10.5.6</a>
01C1 40E0h	IRAWSTAT	Interrupt Raw Status/Set Register	Privileged mode	<a href="#">Section 10.5.7.1</a>
01C1 40E4h	IENSTAT	Interrupt Enable Status/Clear Register	Privileged mode	<a href="#">Section 10.5.7.2</a>
01C1 40E8h	IENSET	Interrupt Enable Register	Privileged mode	<a href="#">Section 10.5.7.3</a>
01C1 40ECh	IENCLR	Interrupt Enable Clear Register	Privileged mode	<a href="#">Section 10.5.7.4</a>
01C1 40F0h	EOI	End of Interrupt Register	Privileged mode	<a href="#">Section 10.5.7.5</a>
01C1 40F4h	FLTADDRR	Fault Address Register	Privileged mode	<a href="#">Section 10.5.8.1</a>
01C1 40F8h	FLTSTAT	Fault Status Register	—	<a href="#">Section 10.5.8.2</a>
01C1 4110h	MSTPRI0	Master Priority 0 Register	Privileged mode	<a href="#">Section 10.5.9.1</a>
01C1 4114h	MSTPRI1	Master Priority 1 Register	Privileged mode	<a href="#">Section 10.5.9.2</a>
01C1 4118h	MSTPRI2	Master Priority 2 Register	Privileged mode	<a href="#">Section 10.5.9.3</a>
01C1 4120h	PINMUX0	Pin Multiplexing Control 0 Register	Privileged mode	<a href="#">Section 10.5.10.1</a>
01C1 4124h	PINMUX1	Pin Multiplexing Control 1 Register	Privileged mode	<a href="#">Section 10.5.10.2</a>
01C1 4128h	PINMUX2	Pin Multiplexing Control 2 Register	Privileged mode	<a href="#">Section 10.5.10.3</a>
01C1 412Ch	PINMUX3	Pin Multiplexing Control 3 Register	Privileged mode	<a href="#">Section 10.5.10.4</a>
01C1 4130h	PINMUX4	Pin Multiplexing Control 4 Register	Privileged mode	<a href="#">Section 10.5.10.5</a>
01C1 4134h	PINMUX5	Pin Multiplexing Control 5 Register	Privileged mode	<a href="#">Section 10.5.10.6</a>
01C1 4138h	PINMUX6	Pin Multiplexing Control 6 Register	Privileged mode	<a href="#">Section 10.5.10.7</a>

<sup>(1)</sup> This register is for internal-use only.

**Table 10-3. System Configuration Module 0 (SYSCFG0) Registers (continued)**

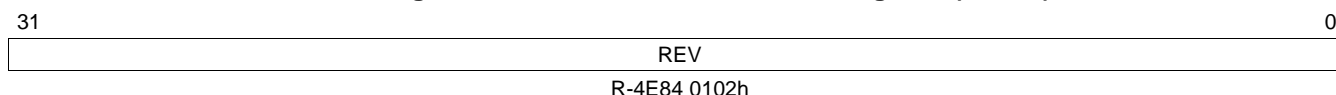
Address	Acronym	Register Description	Access	Section
01C1 413Ch	PINMUX7	Pin Multiplexing Control 7 Register	Privileged mode	<a href="#">Section 10.5.10.8</a>
01C1 4140h	PINMUX8	Pin Multiplexing Control 8 Register	Privileged mode	<a href="#">Section 10.5.10.9</a>
01C1 4144h	PINMUX9	Pin Multiplexing Control 9 Register	Privileged mode	<a href="#">Section 10.5.10.10</a>
01C1 4148h	PINMUX10	Pin Multiplexing Control 10 Register	Privileged mode	<a href="#">Section 10.5.10.11</a>
01C1 414Ch	PINMUX11	Pin Multiplexing Control 11 Register	Privileged mode	<a href="#">Section 10.5.10.12</a>
01C1 4150h	PINMUX12	Pin Multiplexing Control 12 Register	Privileged mode	<a href="#">Section 10.5.10.13</a>
01C1 4154h	PINMUX13	Pin Multiplexing Control 13 Register	Privileged mode	<a href="#">Section 10.5.10.14</a>
01C1 4158h	PINMUX14	Pin Multiplexing Control 14 Register	Privileged mode	<a href="#">Section 10.5.10.15</a>
01C1 415Ch	PINMUX15	Pin Multiplexing Control 15 Register	Privileged mode	<a href="#">Section 10.5.10.16</a>
01C1 4160h	PINMUX16	Pin Multiplexing Control 16 Register	Privileged mode	<a href="#">Section 10.5.10.17</a>
01C1 4164h	PINMUX17	Pin Multiplexing Control 17 Register	Privileged mode	<a href="#">Section 10.5.10.18</a>
01C1 4168h	PINMUX18	Pin Multiplexing Control 18 Register	Privileged mode	<a href="#">Section 10.5.10.19</a>
01C1 416Ch	PINMUX19	Pin Multiplexing Control 19 Register	Privileged mode	<a href="#">Section 10.5.10.20</a>
01C1 4170h	SUSPSRC	Suspend Source Register	Privileged mode	<a href="#">Section 10.5.11</a>
01C1 4174h	CHIPSIG	Chip Signal Register	—	<a href="#">Section 10.5.12</a>
01C1 4178h	CHIPSIG_CLR	Chip Signal Clear Register	—	<a href="#">Section 10.5.13</a>
01C1 417Ch	CFGCHIP0	Chip Configuration 0 Register	Privileged mode	<a href="#">Section 10.5.14</a>
01C1 4180h	CFGCHIP1	Chip Configuration 1 Register	Privileged mode	<a href="#">Section 10.5.15</a>
01C1 4188h	CFGCHIP3	Chip Configuration 3 Register	Privileged mode	<a href="#">Section 10.5.16</a>
01C1 418Ch	CFGCHIP4	Chip Configuration 4 Register	Privileged mode	<a href="#">Section 10.5.17</a>

**Table 10-4. System Configuration Module 1 (SYSCFG1) Registers**

Address	Acronym	Register Description	Access	Section
01E2 C000h	VTPIO_CTL	VTP I/O Control Register	Privileged mode	<a href="#">Section 10.5.18</a>
01E2 C004h	DDR_SLEW	DDR Slew Register	Privileged mode	<a href="#">Section 10.5.19</a>
01E2 C008h	DEEPSLEEP	Deep Sleep Register	Privileged mode	<a href="#">Section 10.5.20</a>
01E2 C00Ch	PUPD_ENA	Pullup/Pulldown Enable Register	Privileged mode	<a href="#">Section 10.5.21</a>
01E2 C010h	PUPD_SEL	Pullup/Pulldown Selection Register	Privileged mode	<a href="#">Section 10.5.22</a>
01E2 C014h	RXACTIVE	RXACTIVE Control Register	Privileged mode	<a href="#">Section 10.5.23</a>

### 10.5.1 Revision Identification Register (REVID)

The revision identification register (REVID) provides the revision information for the SYSCFG module. The REVID is shown in [Figure 10-1](#) and described in [Table 10-5](#).

**Figure 10-1. Revision Identification Register (REVID)**


LEGEND: R = Read only; -n = value after reset

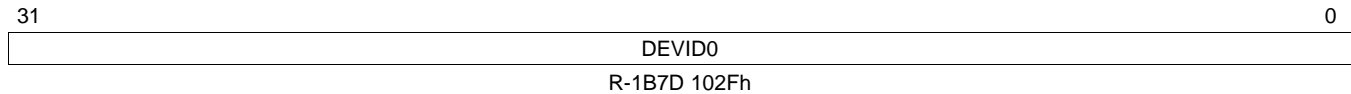
**Table 10-5. Revision Identification Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4E84 0102h	<b>Revision ID.</b> Revision information for the SYSCFG module.

### 10.5.2 Device Identification Register 0 (DEVIDR0)

The device identification register 0 (DEVIDR0) contains a software readable version of the JTAG ID device. Software can use this register to determine the version of the device on which it is executing. The DEVIDR0 is shown in [Figure 10-2](#) and described in [Table 10-6](#).

**Figure 10-2. Device Identification Register 0 (DEVIDR0)**



LEGEND: R = Read only; -n = value after reset

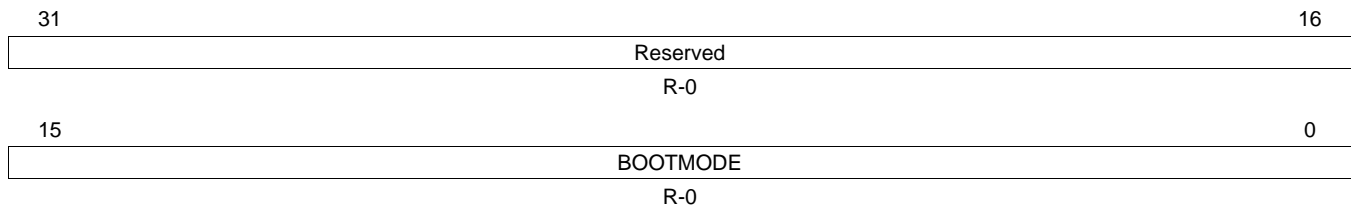
**Table 10-6. Device Identification Register 0 (DEVIDR0) Field Descriptions**

Bit	Field	Value	Description
31-0	DEVID0	1B7D 102Fh	Device identification.

### 10.5.3 Boot Configuration Register (BOOTCFG)

The device boot and configuration settings are latched at device reset, and captured in the boot configuration register (BOOTCFG). See your device-specific data manual and the *Boot Considerations* chapter for details on boot and configuration settings. The BOOTCFG is shown in [Figure 10-3](#) and described in [Table 10-7](#).

**Figure 10-3. Boot Configuration Register (BOOTCFG)**



LEGEND: R = Read only; -n = value after reset

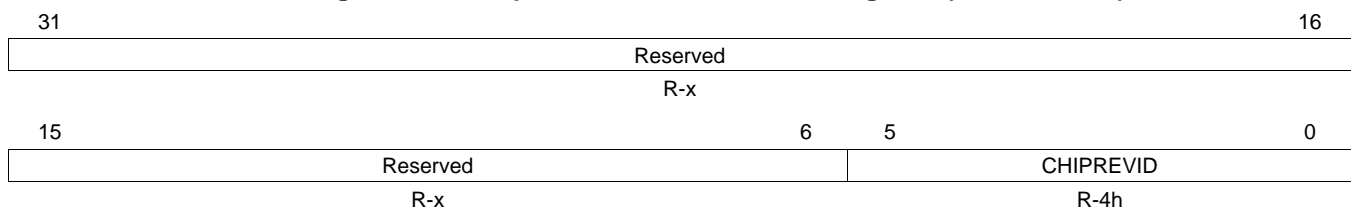
**Table 10-7. Boot Configuration Register (BOOTCFG) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	BOOTMODE	0-FFFFh	<b>Boot Mode.</b> This reflects the state of the boot mode pins.

### 10.5.4 Chip Revision Identification Register (CHIPREVIDR)

The chip revision identification register (CHIPREVIDR) provides the software-readable silicon revision information for the device. The CHIPREVID is shown in [Figure 10-4](#) and described in [Table 10-8](#).

**Figure 10-4. Chip Revision Identification Register (CHIPREVIDR)**



LEGEND: R = Read only; -n = value after reset; x = value is indeterminate after reset

**Table 10-8. Chip Revision Identification Register (CHIPREVIDR) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	CHIPREVID	0-3h 4h	<b>Identifies silicon revision of device.</b> Older silicon revision Silicon revision 2.2

## 10.5.5 Kick Registers (KICK0R-KICK1R)

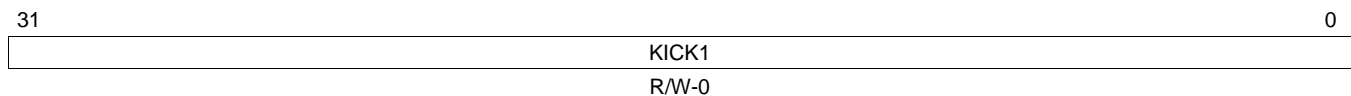
**NOTE:** The kick registers are disabled in silicon revision 2 and later. The SYSCFG registers are always unlocked and writes to the kick registers have no functional effect.

The SYSCFG module has a protection mechanism to prevent any spurious writes from changing any of the modules memory-mapped registers. At power-on reset, none of the SYSCFG module registers are writeable (they are readable). To allow writing to the registers in the module, it is required to “unlock” the registers by writing to two memory-mapped registers in the SYSCFG module, Kick0 and Kick1, with exact data values. Once these values are written, then all the registers in the SYSCFG module that are writeable can be written to. See [Section 10.2.2](#) for the exact key values and sequence of steps. Writing any other data value to either of these kick registers will cause the memory mapped registers to be “locked” again and block out any write accesses to registers in the SYSCFG module.

### 10.5.5.1 Kick 0 Register (KICK0R)

The KICK0R is shown in [Figure 10-5](#) and described in [Table 10-9](#).

**Figure 10-5. Kick 0 Register (KICK0R)**



LEGEND: R/W = Read/Write; -n = value after reset

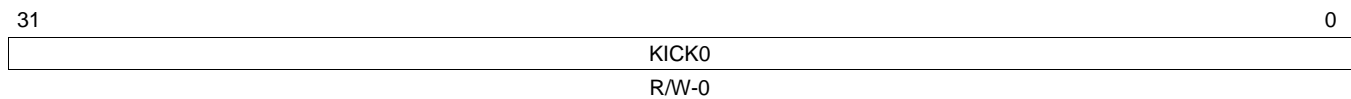
**Table 10-9. Kick 0 Register (KICK0R) Field Descriptions**

Bit	Field	Value	Description
31-0	KICK0	0-FFFF FFFFh	<b>KICK0R allows writing to unlock the kick0 data.</b> The written data must be 83E7 0B13h to unlock this register. It must be written before writing to the kick1 register. Writing any other value will lock the other MMRs.

### 10.5.5.2 Kick 1 Register (KICK1R)

The KICK1R is shown in [Figure 10-6](#) and described in [Table 10-10](#).

**Figure 10-6. Kick 1 Register (KICK1R)**



LEGEND: R/W = Read/Write; -n = value after reset

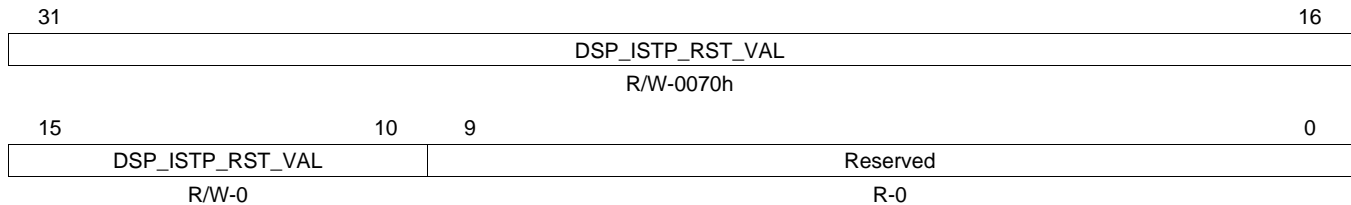
**Table 10-10. Kick 1 Register (KICK1R) Field Descriptions**

Bit	Field	Value	Description
31-0	KICK1	0-FFFF FFFFh	<b>KICK1R allows writing to unlock the kick1 data and the kicker mechanism to write to other MMRs.</b> The written data must be 95A4 F1E0h to unlock this register. KICK0R must be written before writing to the kick1 register. Writing any other value will lock the other MMRs.

### 10.5.6 Host 1 Configuration Register (HOST1CFG)

The host 1 configuration register (HOST1CFG) provides information on the DSP boot address value at power-on reset. The boot address defaults to 0070 0000h (DSP ROM) on power-up. The address field is read/writeable after reset and can be modified to allow execution from an alternate location after a module level or local reset on the DSP. The HOST1CFG is shown in [Figure 10-7](#) and described in [Table 10-11](#).

**Figure 10-7. Host 1 Configuration Register (HOST1CFG)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-11. Host 1 Configuration Register (HOST1CFG) Field Descriptions**

Bit	Field	Value	Description
31-10	DSP_ISTP_RST_VAL	0-3F FFFFh	DSP boot address vector.
9-0	Reserved	0	Reserved

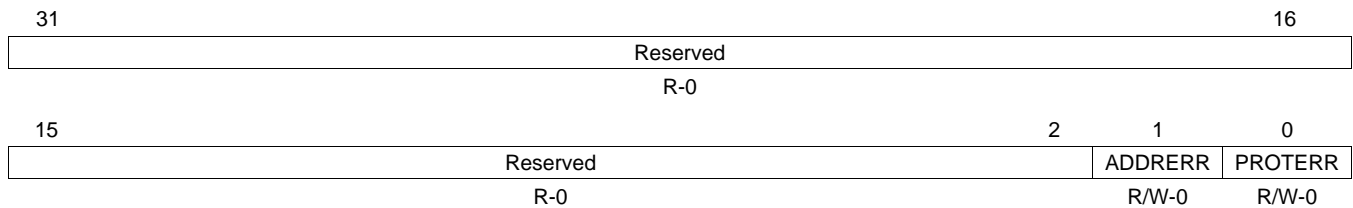
### 10.5.7 Interrupt Registers

The interrupt registers are a set of registers that provide control for the address and protection violation error interrupt generated by the SYSCFG module when there is an address or protection violation to the module's memory-mapped register address space. This includes enable control, interrupt set and clear control, and end of interrupt (EOI) control.

#### 10.5.7.1 Interrupt Raw Status/Set Register (IRAWSTAT)

The interrupt raw status/set register (IRAWSTAT) shows the interrupt status before enabling the interrupt and allows setting of the interrupt status. The IRAWSTAT is shown in [Figure 10-8](#) and described in [Table 10-12](#).

**Figure 10-8. Interrupt Raw Status/Set Register (IRAWSTAT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-12. Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions**

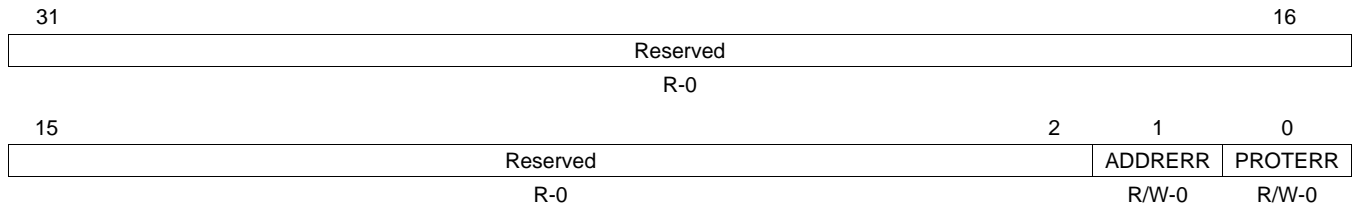
Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR	0 1	<b>Addressing violation error.</b> Reading this bit field reflects the raw status of the interrupt before enabling. 0 Indicates the interrupt is not set. Writing 0 has no effect. 1 Indicates the interrupt is set. Writing 1 sets the status.
0	PROTERR	0 1	<b>Protection violation error.</b> Reading this bit field reflects the raw status of the interrupt before enabling. 0 Indicates the interrupt is not set. Writing 0 has no effect. 1 Indicates the interrupt is set. Writing 1 sets the status.



### 10.5.7.2 Interrupt Enable Status/Clear Register (IENSTAT)

The interrupt enable status/clear register (IENSTAT) shows the status of enabled interrupt and allows clearing of the interrupt status. The IENSTAT is shown in [Figure 10-9](#) and described in [Table 10-13](#).

**Figure 10-9. Interrupt Enable Status/Clear Register (IENSTAT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

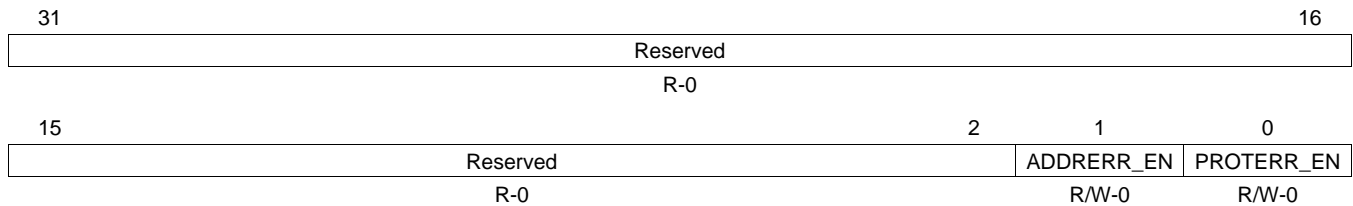
**Table 10-13. Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR	0	<b>Addressing violation error.</b> Reading this bit field reflects the interrupt enabled status. Indicates the interrupt is not set. Writing 0 has no effect.
		1	Indicates the interrupt is set. Writing 1 clears the status.
0	PROTERR	0	<b>Protection violation error.</b> Reading this bit field reflects the interrupt enabled status. Indicates the interrupt is not set. Writing 0 has no effect.
		1	Indicates the interrupt is set. Writing 1 clears the status.

### 10.5.7.3 Interrupt Enable Register (IENSET)

The interrupt enable register (IENSET) allows setting/enabling the interrupt for address and/or protection violation condition. It also shows the value of the register (whether or not interrupt is enabled). The IENSET is shown in [Figure 10-10](#) and described in [Table 10-14](#).

**Figure 10-10. Interrupt Enable Register (IENSET)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

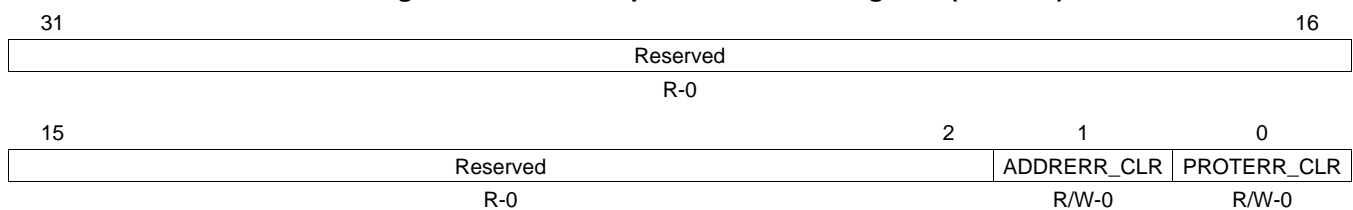
**Table 10-14. Interrupt Enable Register (IENSET) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR_EN	0	<b>Addressing violation error.</b> Writing a 0 has not effect.
		1	Writing a 1 enables this interrupt.
0	PROTERR_EN	0	<b>Protection violation error.</b> Writing a 0 has not effect.
		1	Writing a 1 enables this interrupt.

### 10.5.7.4 Interrupt Enable Clear Register (IENCLR)

The interrupt enable clear register (IENCLR) allows clearing/disable the interrupt for address and/or protection violation condition. It also shows the value of the interrupt enable register (IENSET). The IENCLR is shown in [Figure 10-11](#) and described in [Table 10-15](#).

**Figure 10-11. Interrupt Enable Clear Register (IENCLR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

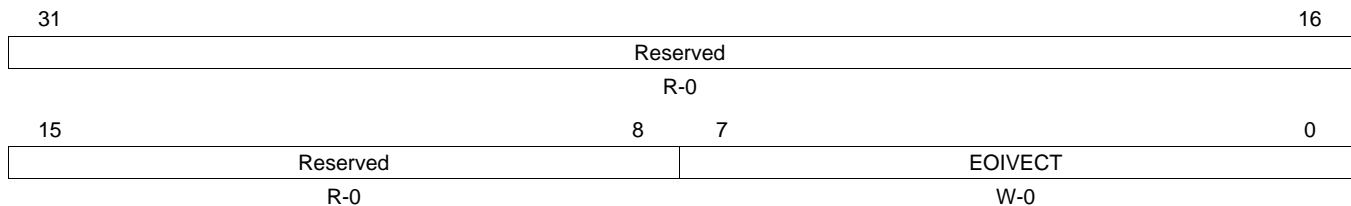
**Table 10-15. Interrupt Enable Clear Register (IENCLR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR_CLR	0	<b>Addressing violation error.</b> Writing a 0 has not effect.
		1	Writing a 1 clears/disables this interrupt.
0	PROTERR_CLR	0	<b>Protection violation error.</b> Writing a 0 has not effect.
		1	Writing a 1 clears/disables this interrupt.

### 10.5.7.5 End of Interrupt Register (EOI)

The end of interrupt register (EOI) is used in software to indicate completion of the interrupt servicing of the SYSCFG interrupt (for address/protection violation). It is required to write a value of 0 to the EOI register after the software has processed the SYSCFG interrupt, this acts as an acknowledgement of completion of the SYSCFG interrupt so that the module can reliably generate the subsequent interrupts. The EOI is shown in [Figure 10-12](#) and described in [Table 10-16](#).

**Figure 10-12. End of Interrupt Register (EOI)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 10-16. End of Interrupt Register (EOI) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always read 0.
7-0	EOI ECT	0-FFh	<b>EOI vector value.</b> Write the interrupt distribution value of the chip.

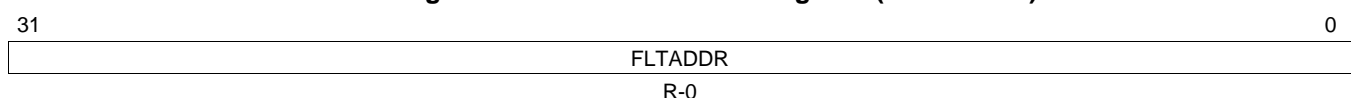
### 10.5.8 Fault Registers

The fault registers are a group of registers responsible for capturing the details on the faulty (address/protection violation errors) accesses, such as address and type of error.

#### 10.5.8.1 Fault Address Register (FLTADDR)

The fault address register (FLTADDR) captures the address of the first transfer that causes the address or memory violation error. The FLTADDR is shown in [Figure 10-13](#) and described in [Table 10-17](#).

**Figure 10-13. Fault Address Register (FLTADDR)**



LEGEND: R = Read only; -n = value after reset

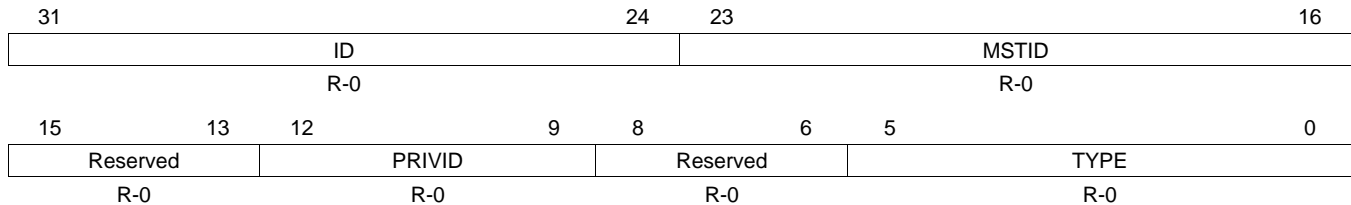
**Table 10-17. Fault Address Register (FLTADDR) Field Descriptions**

Bit	Field	Value	Description
31-0	FLTADDR	0-FFFF FFFFh	<b>Fault address for the first fault transfer.</b>

### 10.5.8.2 Fault Status Register (FLTSTAT)

The fault status register (FLTSTAT) holds/captures additional attributes and status of the first erroneous transaction. This includes things like the master id for the master that caused the address/memory violation error, details on whether it is a user or supervisor level read/write or execute fault. The FLTSTAT is shown in [Figure 10-14](#) and described in [Table 10-18](#).

**Figure 10-14. Fault Status Register (FLTSTAT)**



LEGEND: R = Read only; -n = value after reset

**Table 10-18. Fault Status Register (FLTSTAT) Field Descriptions**

Bit	Field	Value	Description
31-24	ID	0-FFh	<b>Transfer ID of the first fault transfer.</b>
23-16	MSTID	0-FFh	<b>Master ID of the first fault transfer.</b>
15-13	Reserved	0	Reserved. Always read 0
12-9	PRIVID	0-Fh	<b>Privilege ID of the first fault transfer.</b>
8-6	Reserved	0	Reserved. Always read 0
5-0	TYPE		<b>Fault type of first fault transfer.</b>
		0	No transfer fault
		1h	User execute fault
		2h	User write fault
		3h	<i>Reserved</i>
		4h	User read fault
		5h-7h	<i>Reserved</i>
		8h	Supervisor execute fault
		9h-Fh	<i>Reserved</i>
		10h	Supervisor write fault
		11h-1Fh	<i>Reserved</i>
		20h	Supervisor read fault
		21h-3Fh	<i>Reserved</i>

## 10.5.9 Master Priority Registers (MSTPRI0-MSTPRI2)

### 10.5.9.1 Master Priority 0 Register (MSTPRI0)

The master priority 0 register (MSTPRI0) is shown in [Figure 10-15](#) and described in [Table 10-19](#).

**Figure 10-15. Master Priority 0 Register (MSTPRI0)**

31	30	28	27	26	24	23	22	20	19	18	16
Rsvd	Reserved		Rsvd	Reserved		Rsvd	Reserved		Rsvd	Reserved	
R/W-0	R/W-4h		R/W-0	R/W-4h		R/W-0	R/W-4h		R/W-0	R/W-4h	
15	14	12	11	10	8	7	6	4	3	2	0
Rsvd	DSP_CFG		Rsvd	DSP_MDMA		Rsvd	Reserved		Rsvd	Reserved	
R/W-0	R/W-2h		R-0	R/W-2h		R-0	R/W-2h		R-0	R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-19. Master Priority 0 Register (MSTPRI0) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reserved. Write the default value when modifying this register.
30-28	Reserved	4h	Reserved. Write the default value when modifying this register.
27	Reserved	0	Reserved. Write the default value when modifying this register.
26-24	Reserved	4h	Reserved. Write the default value when modifying this register.
23	Reserved	0	Reserved. Write the default value when modifying this register.
22-20	Reserved	4h	Reserved. Write the default value when modifying this register.
19	Reserved	0	Reserved. Write the default value when modifying this register.
18-16	Reserved	4h	Reserved. Write the default value when modifying this register.
15	Reserved	0	Reserved. Write the default value when modifying this register.
14-12	DSP_CFG	0-7h	<b>DSP CFG port priority.</b> Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
11	Reserved	0	Reserved. Always read as 0.
10-8	DSP_MDMA	0-7h	<b>DSP DMA port priority.</b> Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
7	Reserved	0	Reserved. Always read as 0.
6-4	Reserved	2h	Reserved. Write the default value when modifying this register.
3	Reserved	0	Reserved. Always read as 0.
2-0	Reserved	2h	Reserved. Write the default value when modifying this register.

### 10.5.9.2 Master Priority 1 Register (MSTPRI1)

The master priority 1 register (MSTPRI1) is shown in [Figure 10-16](#) and described in [Table 10-20](#).

**Figure 10-16. Master Priority 1 Register (MSTPRI1)**

31	30	28	27	26	24	23	22	20	19	18	16
Rsvd	Reserved		Rsvd	Reserved		Rsvd	Reserved		Rsvd	EDMA31TC0	
R/W-0	R/W-4h		R/W-0	R/W-4h		R/W-0	R/W-4h		R/W-0	R/W-4h	
15	14	12	11	10	8	7	6	4	3	2	0
Rsvd	EDMA30TC1		Rsvd	EDMA30TC0		Rsvd	Reserved		Rsvd	Reserved	
R/W-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-20. Master Priority 1 Register (MSTPRI1) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reserved. Write the default value when modifying this register.
30-28	Reserved	4h	Reserved. Write the default value when modifying this register.
27	Reserved	0	Reserved. Write the default value when modifying this register.
26-24	Reserved	4h	Reserved. Write the default value when modifying this register.
23	Reserved	0	Reserved. Write the default value when modifying this register.
22-20	Reserved	4h	Reserved. Write the default value when modifying this register.
19	Reserved	0	Reserved. Write the default value when modifying this register.
18-16	EDMA31TC0	0-7h	<b>EDMA3_1_TC0 port priority.</b> Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
15	Reserved	0	Reserved. Write the default value when modifying this register.
14-12	EDMA30TC1	0-7h	<b>EDMA3_0_TC1 port priority.</b> Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
11	Reserved	0	Reserved. Always read as 0.
10-8	EDMA30TC0	0-7h	<b>EDMA3_0_TC0 port priority.</b> Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
7	Reserved	0	Reserved. Always read as 0.
6-4	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
3	Reserved	0	Reserved. Always read as 0.
2-0	Reserved	0	Reserved. Write the default value to all bits when modifying this register.

### 10.5.9.3 Master Priority 2 Register (MSTPRI2)

The master priority 2 register (MSTPRI2) is shown in [Figure 10-17](#) and described in [Table 10-21](#).

**Figure 10-17. Master Priority 2 Register (MSTPRI2)**

31	30	28	27	26	24	23	22	20	19	18	16
Rsvd	Reserved		Rsvd	Reserved		Rsvd	UHPI		Rsvd	Reserved	
R/W-0	R/W-5h		R/W-0	R/W-4h		R/W-0	R/W-6h		R/W-0	R/W-0	
15	14	12	11	10	8	7	6	4	3	2	0
Rsvd	Reserved		Rsvd	Reserved		Rsvd	Reserved		Rsvd	Reserved	
R/W-0	R/W-4h		R/W-0	R/W-4h		R/W-0	R/W-0		R/W-0	R/W-4h	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-21. Master Priority 2 Register (MSTPRI2) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reserved. Write the default value when modifying this register.
30-28	Reserved	5h	Reserved. Write the default value when modifying this register.
27	Reserved	0	Reserved. Write the default value when modifying this register.
26-24	Reserved	4h	Reserved. Write the default value when modifying this register.
23	Reserved	0	Reserved. Write the default value when modifying this register.
22-20	UHPI	0-7h	<b>HPI port priority.</b> Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
19	Reserved	0	Reserved. Write the default value when modifying this register.
18-16	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
15	Reserved	0	Reserved. Write the default value when modifying this register.
14-12	Reserved	4h	Reserved. Write the default value when modifying this register.
11	Reserved	0	Reserved. Write the default value when modifying this register.
10-8	Reserved	4h	Reserved. Write the default value when modifying this register.
7	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
6-4	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
3	Reserved	0	Reserved. Write the default value when modifying this register.
2-0	Reserved	4h	Reserved. Write the default value when modifying this register.

### 10.5.10 Pin Multiplexing Control Registers (PINMUX0-PINMUX19)

Extensive use of pin multiplexing is used to accommodate the large number of peripheral functions in the smallest possible package. On the device, pin multiplexing can be controlled on a pin by pin basis. This is done by the pin multiplexing registers (PINMUX0-PINMUX19). Each pin that is multiplexed with several different functions has a corresponding 4-bit field in PINMUX $n$ . Pin multiplexing selects which of several peripheral pin functions control the pins I/O buffer output data and output enable values only. Note that the input from each pin is always routed to all of the peripherals that share the pin; the PINMUX registers have no effect on input from a pin. Hardware does not attempt to ensure that the proper pin multiplexing is selected for the peripherals or that interface mode is being used. Detailed information about the pin multiplexing and control is covered in the device-specific data manual. Access to the pin multiplexing utility is available in *OMAP-L132/L138, TMS320C6742/6/8 Pin Multiplexing Utility Application Report (SPRAB63)*.

#### 10.5.10.1 Pin Multiplexing Control 0 Register (PINMUX0)

**Figure 10-18. Pin Multiplexing Control 0 Register (PINMUX0)**

31	28	27	24	23	20	19	16
PINMUX0_31_28		PINMUX0_27_24		PINMUX0_23_20		PINMUX0_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX0_15_12		PINMUX0_11_8		PINMUX0_7_4		PINMUX0_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Table 10-22. Pin Multiplexing Control 0 Register (PINMUX0) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX0_31_28		<b>RTC_ALARM/GP0[8]/DEEPSLEEP Control</b>	
		0	Selects Function DEEPSLEEP	I
		1h	Reserved	X
		2h	Selects Function RTC_ALARM	O
		3h-7h	Reserved	X
		8h	Selects Function GP0[8]	I/O
27-24	PINMUX0_27_24	9h-Fh	Reserved	X
			<b>AMUTE/GP0[9] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AMUTE	I/O
		2h-7h	Reserved	X
23-20	PINMUX0_23_20	8h	Selects Function GP0[9]	I/O
		9h-Fh	Reserved	X
			<b>AHCLKX/GP0[10] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AHCLKX	I/O
		2h-7h	Reserved	X
		8h	Selects Function GP0[10]	I/O
		9h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state



**Table 10-22. Pin Multiplexing Control 0 Register (PINMUX0) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
19-16	PINMUX0_19_16	0 1h 2h-7h 8h 9h-Fh	<b>AHCLKR/GP0[11] Control</b> Pin is 3-stated. Selects Function AHCLKR <i>Reserved</i> Selects Function GP0[11] <i>Reserved</i>	Z I/O X I/O X
15-12	PINMUX0_15_12	0 1h 2h-7h 8h 9h-Fh	<b>AFSX/GP0[12] Control</b> Pin is 3-stated. Selects Function AFSX <i>Reserved</i> Selects Function GP0[12] <i>Reserved</i>	Z I/O X I/O X
11-8	PINMUX0_11_8	0 1h 2h-7h 8h 9h-Fh	<b>AFSR/GP0[13] Control</b> Pin is 3-stated. Selects Function AFSR <i>Reserved</i> Selects Function GP0[13] <i>Reserved</i>	Z I/O X I/O X
7-4	PINMUX0_7_4	0 1h 2h-7h 8h 9h-Fh	<b>ACLKX/GP0[14] Control</b> Pin is 3-stated. Selects Function ACLKX <i>Reserved</i> Selects Function GP0[14] <i>Reserved</i>	Z I/O X I/O X
3-0	PINMUX0_3_0	0 1h 2h-7h 8h 9h-Fh	<b>ACLKR/GP0[15] Control</b> Pin is 3-stated. Selects Function ACLKR <i>Reserved</i> Selects Function GP0[15] <i>Reserved</i>	Z I/O X I/O X

### 10.5.10.2 Pin Multiplexing Control 1 Register (PINMUX1)

**Figure 10-19. Pin Multiplexing Control 1 Register (PINMUX1)**

31	28	27	24	23	20	19	16
PINMUX1_31_28		PINMUX1_27_24		PINMUX1_23_20		PINMUX1_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX1_15_12		PINMUX1_11_8		PINMUX1_7_4		PINMUX1_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-23. Pin Multiplexing Control 1 Register (PINMUX1) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX1_31_28		<b>AXR8/CLKS1/ECAP1_APWM1/GP0[0] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR8	I/O
		2h	Selects Function CLKS1	I
		3h	<i>Reserved</i>	X
		4h	Selects Function ECAP1_APWM1	I/O
		5h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[0]	I/O
9h-Fh	<i>Reserved</i>	X		
27-24	PINMUX1_27_24		<b>AXR9/DX1/GP0[1] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR9	I/O
		2h	Selects Function DX1	O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[1]	I/O
9h-Fh	<i>Reserved</i>	X		
23-20	PINMUX1_23_20		<b>AXR10/DR1/GP0[2] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR10	I/O
		2h	Selects Function DR1	I
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[2]	I/O
9h-Fh	<i>Reserved</i>	X		
19-16	PINMUX1_19_16		<b>AXR11/FSX1/GP0[3] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR11	I/O
		2h	Selects Function FSX1	I/O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[3]	I/O
9h-Fh	<i>Reserved</i>	X		

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-23. Pin Multiplexing Control 1 Register (PINMUX1) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
15-12	PINMUX1_15_12		<b>AXR12/FSR1/GP0[4] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR12	I/O
		2h	Selects Function FSR1	I/O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[4]	I/O
		9h-Fh	<i>Reserved</i>	X
11-8	PINMUX1_11_8		<b>AXR13/CLKX1/GP0[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR13	I/O
		2h	Selects Function CLKX1	I/O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[5]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX1_7_4		<b>AXR14/CLKR1/GP0[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR14	I/O
		2h	Selects Function CLKR1	I/O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[6]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX1_3_0		<b>AXR15/EPWM0TZ[0]/ECAP2_APWM2/GP0[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR15	I/O
		2h	Selects Function EPWM0TZ[0]	I
		3h	<i>Reserved</i>	X
		4h	Selects Function ECAP2_APWM2	I/O
		5h-7h	<i>Reserved</i>	X
		8h	Selects Function GP0[7]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.3 Pin Multiplexing Control 2 Register (PINMUX2)

**Figure 10-20. Pin Multiplexing Control 2 Register (PINMUX2)**

31	28	27	24	23	20	19	16
PINMUX2_31_28		PINMUX2_27_24		PINMUX2_23_20		PINMUX2_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX2_15_12		PINMUX2_11_8		PINMUX2_7_4		PINMUX2_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-24. Pin Multiplexing Control 2 Register (PINMUX2) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX2_31_28		<b>AXR0/ECAP0_APWM0/GP8[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR0	I/O
		2h	Selects Function ECAP0_APWM0	I/O
		3h	<i>Reserved</i>	X
		4h	Selects Function GP8[7]	I/O
27-24	PINMUX2_27_24	5h-Fh	<i>Reserved</i>	X
			<b>AXR1/GP1[9] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR1	I/O
		2h-3h	<i>Reserved</i>	X
23-20	PINMUX2_23_20	4h	Selects Function GP1[9]	I/O
		5h-Fh	<i>Reserved</i>	X
			<b>AXR2/GP1[10] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR2	I/O
19-16	PINMUX2_19_16	2h-3h	<i>Reserved</i>	X
		4h	Selects Function GP1[10]	I/O
		5h-Fh	<i>Reserved</i>	X
			<b>AXR3/GP1[11] Control</b>	
		0	Pin is 3-stated.	Z
15-12	PINMUX2_15_12	1h	Selects Function AXR3	I/O
		2h-3h	<i>Reserved</i>	X
		4h	Selects Function GP1[11]	I/O
		5h-Fh	<i>Reserved</i>	X
			<b>AXR4/GP1[12] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR4	I/O
		2h-3h	<i>Reserved</i>	X
		4h	Selects Function GP1[12]	I/O
		5h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-24. Pin Multiplexing Control 2 Register (PINMUX2) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX2_11_8		<b>AXR5/GP1[13] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR5	I/O
		2h-3h	<i>Reserved</i>	X
		4h	Selects Function GP1[13]	I/O
	5h-Fh	<i>Reserved</i>	X	
7-4	PINMUX2_7_4		<b>AXR6/GP1[14] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR6	I/O
		2h-3h	<i>Reserved</i>	X
		4h	Selects Function GP1[14]	I/O
	5h-Fh	<i>Reserved</i>	X	
3-0	PINMUX2_3_0		<b>AXR7/EPWM1TZ[0]/GP1[15] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR7	I/O
		2h	Selects Function EPWM1TZ[0]	I
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP1[15]	I/O
	9h-Fh	<i>Reserved</i>	X	

### 10.5.10.4 Pin Multiplexing Control 3 Register (PINMUX3)

**Figure 10-21. Pin Multiplexing Control 3 Register (PINMUX3)**

31	28	27	24	23	20	19	16
PINMUX3_31_28		PINMUX3_27_24		PINMUX3_23_20		PINMUX3_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX3_15_12		PINMUX3_11_8		PINMUX3_7_4		PINMUX3_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-25. Pin Multiplexing Control 3 Register (PINMUX3) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX3_31_28		<b>UART0_RTS/GP8[1] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function <u>UART0_RTS</u>	O
		3h	Reserved	X
		4h	Selects Function GP8[1]	I/O
5h-Fh	Reserved	X		
27-24	PINMUX3_27_24		<b>UART0_CTS/GP8[2] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function <u>UART0_CTS</u>	I
		3h	Reserved	X
		4h	Selects Function GP8[2]	I/O
5h-Fh	Reserved	X		
23-20	PINMUX3_23_20		<b>UART0_TXD/GP8[3] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function <u>UART0_TXD</u>	O
		3h	Reserved	X
		4h	Selects Function GP8[3]	I/O
5h-Fh	Reserved	X		
19-16	PINMUX3_19_16		<b>UART0_RXD/GP8[4] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function <u>UART0_RXD</u>	I
		3h	Reserved	X
		4h	Selects Function GP8[4]	I/O
5h-Fh	Reserved	X		
15-12	PINMUX3_15_12		<b>EPWMSYNCO/GP8[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function <u>EPWMSYNCO</u>	O
		3h	Reserved	X
		4h	Selects Function GP8[5]	I/O
5h-Fh	Reserved	X		

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-25. Pin Multiplexing Control 3 Register (PINMUX3) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX3_11_8		<b>EPWMSYNCI/GP8[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function EPWMSYNCI	I
		3h	Reserved	X
		4h	Selects Function GP8[6]	I/O
	5h-Fh	Reserved	X	
7-4	PINMUX3_7_4		<b>EPWM0B Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function EPWM0B	I/O
	3h-Fh	Reserved	X	
3-0	PINMUX3_3_0		<b>EPWM0A/GP1[8] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function EPWM0A	I/O
		3h	Reserved	X
		4h	Selects Function GP1[8]	I/O
	5h-Fh	Reserved	X	

### 10.5.10.5 Pin Multiplexing Control 4 Register (PINMUX4)

**Figure 10-22. Pin Multiplexing Control 4 Register (PINMUX4)**

31	28	27	24	23	20	19	16
PINMUX4_31_28		PINMUX4_27_24		PINMUX4_23_20		PINMUX4_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX4_15_12		PINMUX4_11_8		PINMUX4_7_4		PINMUX4_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-26. Pin Multiplexing Control 4 Register (PINMUX4) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX4_31_28		<b>SPI1_SCS[2]/GP1[0] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1\_SCS}}[2]$	I/O
		2h-7h	Reserved	X
		8h	Selects Function GP1[0]	I/O
9h-Fh	Reserved	X		
27-24	PINMUX4_27_24		<b>SPI1_SCS[3]/GP1[1] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1\_SCS}}[3]$	I/O
		2h-7h	Reserved	X
		8h	Selects Function GP1[1]	I/O
9h-Fh	Reserved	X		
23-20	PINMUX4_23_20		<b>SPI1_SCS[4]/GP1[2] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1\_SCS}}[4]$	I/O
		2h-7h	Reserved	X
		8h	Selects Function GP1[2]	I/O
9h-Fh	Reserved	X		
19-16	PINMUX4_19_16		<b>SPI1_SCS[5]/GP1[3] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1\_SCS}}[5]$	I/O
		2h-7h	Reserved	X
		8h	Selects Function GP1[3]	I/O
9h-Fh	Reserved	X		
15-12	PINMUX4_15_12		<b>SPI1_SCS[6]/I2C0_SDA/GP1[4] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1\_SCS}}[6]$	I/O
		2h	Selects Function I2C0_SDA	I/O
		3h-7h	Reserved	X
		8h	Selects Function GP1[4]	I/O
9h-Fh	Reserved	X		

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state



**Table 10-26. Pin Multiplexing Control 4 Register (PINMUX4) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX4_11_8		<b>SPI1_SCS[7]/I2C0_SCL/GP1[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function <u>SPI1_SCS[7]</u>	I/O
		2h	Selects Function I2C0_SCL	I/O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP1[5]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX4_7_4		<b>TM64P1_OUT12/GP1[6]/TM64P1_IN12 Control</b>	
		0	Selects Function TM64P1_IN12	I
		1h	<i>Reserved</i>	X
		2h	Selects Function TM64P1_OUT12	O
		3h	<i>Reserved</i>	X
		4h	Selects Function GP1[6]	I/O
		5h-Fh	<i>Reserved</i>	X
3-0	PINMUX4_3_0		<b>TM64P0_OUT12/GP1[7]/TM64P0_IN12 Control</b>	
		0	Selects Function TM64P0_IN12	I
		1h	<i>Reserved</i>	X
		2h	Selects Function TM64P0_OUT12	O
		3h	<i>Reserved</i>	X
		4h	Selects Function GP1[7]	I/O
		5h-Fh	<i>Reserved</i>	X

### 10.5.10.6 Pin Multiplexing Control 5 Register (PINMUX5)

**Figure 10-23. Pin Multiplexing Control 5 Register (PINMUX5)**

31	28	27	24	23	20	19	16
PINMUX5_31_28		PINMUX5_27_24		PINMUX5_23_20		PINMUX5_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX5_15_12		PINMUX5_11_8		PINMUX5_7_4		PINMUX5_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-27. Pin Multiplexing Control 5 Register (PINMUX5) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX5_31_28	0	<b>EMA_BA[0]/GP2[8] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_BA[0]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[8]	I/O
		9h-Fh	<i>Reserved</i>	X
27-24	PINMUX5_27_24	0	<b>EMA_BA[1]/GP2[9] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_BA[1]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[9]	I/O
		9h-Fh	<i>Reserved</i>	X
23-20	PINMUX5_23_20	0	<b>SPI1_SIMO/GP2[10] Control</b> Pin is 3-stated.	Z
		1h	Selects Function SPI1_SIMO	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[10]	I/O
		9h-Fh	<i>Reserved</i>	X
19-16	PINMUX5_19_16	0	<b>SPI1_SOMI/GP2[11] Control</b> Pin is 3-stated.	Z
		1h	Selects Function SPI1_SOMI	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[11]	I/O
		9h-Fh	<i>Reserved</i>	X
15-12	PINMUX5_15_12	0	<b>SPI1_ENA/GP2[12] Control</b> Pin is 3-stated.	Z
		1h	Selects Function SPI1_ENA	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[12]	I/O
		9h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-27. Pin Multiplexing Control 5 Register (PINMUX5) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX5_11_8	0 1h 2h-7h 8h 9h-Fh	<b>SPI1_CLK/GP2[13] Control</b> Pin is 3-stated. Selects Function SPI1_CLK <i>Reserved</i> Selects Function GP2[13] <i>Reserved</i>	Z I/O X I/O X
7-4	PINMUX5_7_4	0 1h 2h 3h-7h 8h 9h-Fh	<b>SPI1_SCS[0]/EPWM1B/GP2[14] Control</b> Pin is 3-stated. Selects Function SPI1_SCS[0] Selects Function EPWM1B <i>Reserved</i> Selects Function GP2[14] <i>Reserved</i>	Z I/O I/O X I/O X
3-0	PINMUX5_3_0	0 1h 2h 3h-7h 8h 9h-Fh	<b>SPI1_SCS[1]/EPWM1A/GP2[15] Control</b> Pin is 3-stated. Selects Function SPI1_SCS[1] Selects Function EPWM1A <i>Reserved</i> Selects Function GP2[15] <i>Reserved</i>	Z I/O I/O X I/O X

### 10.5.10.7 Pin Multiplexing Control 6 Register (PINMUX6)

**Figure 10-24. Pin Multiplexing Control 6 Register (PINMUX6)**

31	28	27	24	23	20	19	16
PINMUX6_31_28		PINMUX6_27_24		PINMUX6_23_20		PINMUX6_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX6_15_12		PINMUX6_11_8		PINMUX6_7_4		PINMUX6_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-28. Pin Multiplexing Control 6 Register (PINMUX6) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX6_31_28		<b>EMA_CS[0]/GP2[0] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA\_CS}}[0]$	O
		2h-7h	Reserved	X
		8h	Selects Function GP2[0]	I/O
9h-Fh	Reserved	X		
27-24	PINMUX6_27_24		<b>EMA_WAIT[1]/GP2[1] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA\_WAIT}}[1]$	I
		2h-7h	Reserved	X
		8h	Selects Function GP2[1]	I/O
9h-Fh	Reserved	X		
23-20	PINMUX6_23_20		<b>EMA_WE_DQM[1]/GP2[2] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA\_WE\_DQM}}[1]$	O
		2h-7h	Reserved	X
		8h	Selects Function GP2[2]	I/O
9h-Fh	Reserved	X		
19-16	PINMUX6_19_16		<b>EMA_WE_DQM[0]/GP2[3] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA\_WE\_DQM}}[0]$	O
		2h-7h	Reserved	X
		8h	Selects Function GP2[3]	I/O
9h-Fh	Reserved	X		
15-12	PINMUX6_15_12		<b>EMA_CAS/GP2[4] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA\_CAS}}$	O
		2h-7h	Reserved	X
		8h	Selects Function GP2[4]	I/O
9h-Fh	Reserved	X		

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-28. Pin Multiplexing Control 6 Register (PINMUX6) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX6_11_8		<b>EMA_RAS/GP2[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function <u>EMA_RAS</u>	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[5]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX6_7_4		<b>EMA_SDCKE/GP2[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_SDCKE	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[6]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX6_3_0		<b>EMA_CLK/GP2[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_CLK	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP2[7]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.8 Pin Multiplexing Control 7 Register (PINMUX7)

**Figure 10-25. Pin Multiplexing Control 7 Register (PINMUX7)**

31	28	27	24	23	20	19	16
PINMUX7_31_28		PINMUX7_27_24		PINMUX7_23_20		PINMUX7_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX7_15_12		PINMUX7_11_8		PINMUX7_7_4		PINMUX7_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-29. Pin Multiplexing Control 7 Register (PINMUX7) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX7_31_28	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_WAIT[0]	I
		2h-7h	Reserved	X
		8h	Selects Function GP3[8]	I/O
		9h-Fh	Reserved	X
27-24	PINMUX7_27_24	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A_R $\bar{W}$	O
		2h-7h	Reserved	X
		8h	Selects Function GP3[9]	I/O
		9h-Fh	Reserved	X
23-20	PINMUX7_23_20	0	Pin is 3-stated.	Z
		1h	Selects Function $\bar{E}MA\_OE$	O
		2h-7h	Reserved	X
		8h	Selects Function GP3[10]	I/O
		9h-Fh	Reserved	X
19-16	PINMUX7_19_16	0	Pin is 3-stated.	Z
		1h	Selects Function $\bar{E}MA\_WE$	O
		2h-7h	Reserved	X
		8h	Selects Function GP3[11]	I/O
		9h-Fh	Reserved	X
15-12	PINMUX7_15_12	0	Pin is 3-stated.	Z
		1h	Selects Function $\bar{E}MA\_CS[5]$	O
		2h-7h	Reserved	X
		8h	Selects Function GP3[12]	I/O
		9h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-29. Pin Multiplexing Control 7 Register (PINMUX7) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX7_11_8		<b>EMA_CS[4]/GP3[13] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function <u>EMA_CS[4]</u>	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[13]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX7_7_4		<b>EMA_CS[3]/GP3[14] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function <u>EMA_CS[3]</u>	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[14]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX7_3_0		<b>EMA_CS[2]/GP3[15] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function <u>EMA_CS[2]</u>	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[15]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.9 Pin Multiplexing Control 8 Register (PINMUX8)

**Figure 10-26. Pin Multiplexing Control 8 Register (PINMUX8)**

31	28	27	24	23	20	19	16
PINMUX8_31_28		PINMUX8_27_24		PINMUX8_23_20		PINMUX8_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX8_15_12		PINMUX8_11_8		PINMUX8_7_4		PINMUX8_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-30. Pin Multiplexing Control 8 Register (PINMUX8) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX8_31_28	0	<b>EMA_D[8]/GP3[0] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[8]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[0]	I/O
		9h-Fh	<i>Reserved</i>	X
27-24	PINMUX8_27_24	0	<b>EMA_D[9]/GP3[1] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[9]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[1]	I/O
		9h-Fh	<i>Reserved</i>	X
23-20	PINMUX8_23_20	0	<b>EMA_D[10]/GP3[2] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[10]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[2]	I/O
		9h-Fh	<i>Reserved</i>	X
19-16	PINMUX8_19_16	0	<b>EMA_D[11]/GP3[3] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[11]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[3]	I/O
		9h-Fh	<i>Reserved</i>	X
15-12	PINMUX8_15_12	0	<b>EMA_D[12]/GP3[4] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[12]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[4]	I/O
		9h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state



**Table 10-30. Pin Multiplexing Control 8 Register (PINMUX8) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX8_11_8		<b>EMA_D[13]/GP3[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[13]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[5]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX8_7_4		<b>EMA_D[14]/GP3[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[14]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[6]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX8_3_0		<b>EMA_D[15]/GP3[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[15]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP3[7]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.10 Pin Multiplexing Control 9 Register (PINMUX9)

**Figure 10-27. Pin Multiplexing Control 9 Register (PINMUX9)**

31	28	27	24	23	20	19	16
PINMUX9_31_28		PINMUX9_27_24		PINMUX9_23_20		PINMUX9_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX9_15_12		PINMUX9_11_8		PINMUX9_7_4		PINMUX9_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-31. Pin Multiplexing Control 9 Register (PINMUX9) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX9_31_28	0	<b>EMA_D[0]/GP4[8] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[0]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[8]	I/O
		9h-Fh	<i>Reserved</i>	X
27-24	PINMUX9_27_24	0	<b>EMA_D[1]/GP4[9] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[1]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[9]	I/O
		9h-Fh	<i>Reserved</i>	X
23-20	PINMUX9_23_20	0	<b>EMA_D[2]/GP4[10] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[2]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[10]	I/O
		9h-Fh	<i>Reserved</i>	X
19-16	PINMUX9_19_16	0	<b>EMA_D[3]/GP4[11] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[3]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[11]	I/O
		9h-Fh	<i>Reserved</i>	X
15-12	PINMUX9_15_12	0	<b>EMA_D[4]/GP4[12] Control</b> Pin is 3-stated.	Z
		1h	Selects Function EMA_D[4]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[12]	I/O
		9h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-31. Pin Multiplexing Control 9 Register (PINMUX9) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX9_11_8		<b>EMA_D[5]/GP4[13] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[5]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[13]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX9_7_4		<b>EMA_D[6]/GP4[14] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[6]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[14]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX9_3_0		<b>EMA_D[7]/GP4[15] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[7]	I/O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[15]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.11 Pin Multiplexing Control 10 Register (PINMUX10)

**Figure 10-28. Pin Multiplexing Control 10 Register (PINMUX10)**

31	28	27	24	23	20	19	16
PINMUX10_31_28		PINMUX10_27_24		PINMUX10_23_20		PINMUX10_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX10_15_12		PINMUX10_11_8		PINMUX10_7_4		PINMUX10_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-32. Pin Multiplexing Control 10 Register (PINMUX10) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX10_31_28	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[16]	O
		2h-7h	Reserved	X
		8h	Selects Function GP4[0]	I/O
		9h-Fh	Reserved	X
27-24	PINMUX10_27_24	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[17]	O
		2h-7h	Reserved	X
		8h	Selects Function GP4[1]	I/O
		9h-Fh	Reserved	X
23-20	PINMUX10_23_20	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[18]	O
		2h-7h	Reserved	X
		8h	Selects Function GP4[2]	I/O
		9h-Fh	Reserved	X
19-16	PINMUX10_19_16	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[19]	O
		2h-7h	Reserved	X
		8h	Selects Function GP4[3]	I/O
		9h-Fh	Reserved	X
15-12	PINMUX10_15_12	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[20]	O
		2h-7h	Reserved	X
		8h	Selects Function GP4[4]	I/O
		9h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-32. Pin Multiplexing Control 10 Register (PINMUX10) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX10_11_8		<b>EMA_A[21]/GP4[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[21]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[5]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX10_7_4		<b>EMA_A[22]/GP4[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[22]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[6]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX10_3_0		<b>GP4[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	<i>Reserved</i>	X
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP4[7]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.12 Pin Multiplexing Control 11 Register (PINMUX11)

**Figure 10-29. Pin Multiplexing Control 11 Register (PINMUX11)**

31	28	27	24	23	20	19	16
PINMUX11_31_28		PINMUX11_27_24		PINMUX11_23_20		PINMUX11_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX11_15_12		PINMUX11_11_8		PINMUX11_7_4		PINMUX11_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-33. Pin Multiplexing Control 11 Register (PINMUX11) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX11_31_28	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[8]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[8]	I/O
		9h-Fh	Reserved	X
27-24	PINMUX11_27_24	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[9]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[9]	I/O
		9h-Fh	Reserved	X
23-20	PINMUX11_23_20	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[10]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[10]	I/O
		9h-Fh	Reserved	X
19-16	PINMUX11_19_16	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[11]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[11]	I/O
		9h-Fh	Reserved	X
15-12	PINMUX11_15_12	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[12]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[12]	I/O
		9h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-33. Pin Multiplexing Control 11 Register (PINMUX11) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX11_11_8		<b>EMA_A[13]/GP5[13] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[13]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP5[13]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX11_7_4		<b>EMA_A[14]/GP5[14] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[14]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP5[14]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX11_3_0		<b>EMA_A[15]/GP5[15] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[15]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP5[15]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.13 Pin Multiplexing Control 12 Register (PINMUX12)

**Figure 10-30. Pin Multiplexing Control 12 Register (PINMUX12)**

31	28	27	24	23	20	19	16
PINMUX12_31_28		PINMUX12_27_24		PINMUX12_23_20		PINMUX12_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX12_15_12		PINMUX12_11_8		PINMUX12_7_4		PINMUX12_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-34. Pin Multiplexing Control 12 Register (PINMUX12) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX12_31_28	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[0]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[0]	I/O
		9h-Fh	Reserved	X
27-24	PINMUX12_27_24	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[1]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[1]	I/O
		9h-Fh	Reserved	X
23-20	PINMUX12_23_20	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[2]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[2]	I/O
		9h-Fh	Reserved	X
19-16	PINMUX12_19_16	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[3]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[3]	I/O
		9h-Fh	Reserved	X
15-12	PINMUX12_15_12	0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[4]	O
		2h-7h	Reserved	X
		8h	Selects Function GP5[4]	I/O
		9h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state



**Table 10-34. Pin Multiplexing Control 12 Register (PINMUX12) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX12_11_8		<b>EMA_A[5]/GP5[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[5]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP5[5]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX12_7_4		<b>EMA_A[6]/GP5[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[6]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP5[6]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX12_3_0		<b>EMA_A[7]/GP5[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[7]	O
		2h-7h	<i>Reserved</i>	X
		8h	Selects Function GP5[7]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.14 Pin Multiplexing Control 13 Register (PINMUX13)

**Figure 10-31. Pin Multiplexing Control 13 Register (PINMUX13)**

31	28	27	24	23	20	19	16
PINMUX13_31_28		PINMUX13_27_24		PINMUX13_23_20		PINMUX13_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX13_15_12		PINMUX13_11_8		PINMUX13_7_4		PINMUX13_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-35. Pin Multiplexing Control 13 Register (PINMUX13) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX13_31_28		<b>UHPI_HR<math>\bar{W}</math>/GP6[8] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HR $\bar{W}$	I
		3h-7h	Reserved	X
		8h	Selects Function GP6[8]	I/O
27-24	PINMUX13_27_24		<b>UHPI_HHWIL/GP6[9] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HHWIL	I
		3h-7h	Reserved	X
		8h	Selects Function GP6[9]	I/O
23-20	PINMUX13_23_20		<b>UHPI_HCNTL1/GP6[10] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HCNTL1	I
		3h-7h	Reserved	X
		8h	Selects Function GP6[10]	I/O
19-16	PINMUX13_19_16		<b>UHPI_HCNTL0/GP6[11] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HCNTL0	I
		3h-7h	Reserved	X
		8h	Selects Function GP6[11]	I/O
15-12	PINMUX13_15_12		<b>UHPI_HINT/GP6[12] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HINT	O
		3h-7h	Reserved	X
		8h	Selects Function GP6[12]	I/O
		9h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-35. Pin Multiplexing Control 13 Register (PINMUX13) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
11-8	PINMUX13_11_8		<b>UHPI_HRDY/GP6[13] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function $\overline{\text{UHPI\_HRDY}}$	O
		3h-7h	Reserved	X
		8h	Selects Function GP6[13]	I/O
7-4	PINMUX13_7_4		<b>CLKOUT/UHPI_HDS2/GP6[14] Control</b>	
		0	Pin is 3-stated.	Z
		1h	Selects Function CLKOUT	O
		2h	Selects Function $\overline{\text{UHPI\_HDS2}}$	I
		3h-7h	Reserved	X
		8h	Selects Function GP6[14]	I/O
3-0	PINMUX13_3_0		<b>RESETOUT/UHPI_HAS/GP6[15] Control</b>	
		0	Selects Function $\overline{\text{RESETOUT}}$	O
		1h	Selects Function $\overline{\text{RESETOUT}}$	O
		2h	Selects Function $\overline{\text{UHPI\_HAS}}$	I
		3h-7h	Reserved	X
		8h	Selects Function GP6[15]	I/O
		9h-Fh	Reserved	X

### 10.5.10.15 Pin Multiplexing Control 14 Register (PINMUX14)

**Figure 10-32. Pin Multiplexing Control 14 Register (PINMUX14)**

31	28	27	24	23	20	19	16
PINMUX14_31_28		PINMUX14_27_24		PINMUX14_23_20		PINMUX14_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX14_15_12		PINMUX14_11_8		PINMUX14_7_4		PINMUX14_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-36. Pin Multiplexing Control 14 Register (PINMUX14) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX14_31_28	0	UHPI_HD[10] Control Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[10]	I/O
		3h-Fh	Reserved	X
27-24	PINMUX14_27_24	0	UHPI_HD[11] Control Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[11]	I/O
		3h-Fh	Reserved	X
23-20	PINMUX14_23_20	0	UHPI_HD[12] Control Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[12]	I/O
		3h-Fh	Reserved	X
19-16	PINMUX14_19_16	0	UHPI_HD[13] Control Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[13]	I/O
		3h-Fh	Reserved	X
15-12	PINMUX14_15_12	0	UHPI_HD[14] Control Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[14]	I/O
		3h-Fh	Reserved	X
11-8	PINMUX14_11_8	0	UHPI_HD[15] Control Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[15]	I/O
		3h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-36. Pin Multiplexing Control 14 Register (PINMUX14) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
7-4	PINMUX14_7_4		<b>UHPI_HDS1/GP6[6] Control</b>	
		0	Pin is 3-stated.	Z
		1h	<i>Reserved</i>	X
		2h	Selects Function $\overline{\text{UHPI\_HDS1}}$	I
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP6[6]	I/O
3-0	PINMUX14_3_0		<b>UHPI_HCS/GP6[7] Control</b>	
		0	Pin is 3-stated.	Z
		1h	<i>Reserved</i>	X
		2h	Selects Function $\overline{\text{UHPI\_HCS}}$	I
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP6[7]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.16 Pin Multiplexing Control 15 Register (PINMUX15)

**Figure 10-33. Pin Multiplexing Control 15 Register (PINMUX15)**

31	28	27	24	23	20	19	16
PINMUX15_31_28		PINMUX15_27_24		PINMUX15_23_20		PINMUX15_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX15_15_12		PINMUX15_11_8		PINMUX15_7_4		PINMUX15_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-37. Pin Multiplexing Control 15 Register (PINMUX15) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX15_31_28	0	<b>UHPI_HD[2] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[2]	I/O
		3h-Fh	Reserved	X
27-24	PINMUX15_27_24	0	<b>UHPI_HD[3] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[3]	I/O
		3h-Fh	Reserved	X
23-20	PINMUX15_23_20	0	<b>UHPI_HD[4] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[4]	I/O
		3h-Fh	Reserved	X
19-16	PINMUX15_19_16	0	<b>UHPI_HD[5] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[5]	I/O
		3h-Fh	Reserved	X
15-12	PINMUX15_15_12	0	<b>UHPI_HD[6] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[6]	I/O
		3h-Fh	Reserved	X
11-8	PINMUX15_11_8	0	<b>UHPI_HD[7] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[7]	I/O
		3h-Fh	Reserved	X
7-4	PINMUX15_7_4	0	<b>UHPI_HD[8] Control</b> Pin is 3-stated.	Z
		1h	Reserved	X
		2h	Selects Function UHPI_HD[8]	I/O
		3h-Fh	Reserved	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-37. Pin Multiplexing Control 15 Register (PINMUX15) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
3-0	PINMUX15_3_0		<b>UHPI_HD[9] Control</b>	
		0	Pin is 3-stated.	Z
		1h	<i>Reserved</i>	X
		2h	Selects Function UHPI_HD[9]	I/O
		3h-Fh	<i>Reserved</i>	X

### 10.5.10.17 Pin Multiplexing Control 16 Register (PINMUX16)

**Figure 10-34. Pin Multiplexing Control 16 Register (PINMUX16)**

31	28	27	24	23	20	19	16
PINMUX16_31_28		PINMUX16_27_24		PINMUX16_23_20		PINMUX16_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX16_15_12		PINMUX16_11_8		PINMUX16_7_4		PINMUX16_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-38. Pin Multiplexing Control 16 Register (PINMUX16) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX16_31_28	0	<b>GP7[10] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[10]	I/O
		9h-Fh	<i>Reserved</i>	X
27-24	PINMUX16_27_24	0	<b>GP7[11] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[11]	I/O
		9h-Fh	<i>Reserved</i>	X
23-20	PINMUX16_23_20	0	<b>GP7[12] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[12]	I/O
		9h-Fh	<i>Reserved</i>	X
19-16	PINMUX16_19_16	0	<b>GP7[13] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[13]	I/O
		9h-Fh	<i>Reserved</i>	X
15-12	PINMUX16_15_12	0	<b>GP7[14] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[14]	I/O
		9h-Fh	<i>Reserved</i>	X
11-8	PINMUX16_11_8	0	<b>GP7[15] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[15]	I/O
		9h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state



**Table 10-38. Pin Multiplexing Control 16 Register (PINMUX16) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
7-4	PINMUX16_7_4		<b>UHPI_HD[0]/GP6[5] Control</b>	
		0	Pin is 3-stated.	Z
		1h	<i>Reserved</i>	X
		2h	Selects Function UHPI_HD[0]	I/O
		3h-7h	<i>Reserved</i>	X
		8h	Selects Function GP6[5]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX16_3_0		<b>UHPI_HD[1] Control</b>	
		0	Pin is 3-stated.	Z
		1h	<i>Reserved</i>	X
		2h	Selects Function UHPI_HD[1]	I/O
		3h-Fh	<i>Reserved</i>	X

### 10.5.10.18 Pin Multiplexing Control 17 Register (PINMUX17)

**Figure 10-35. Pin Multiplexing Control 17 Register (PINMUX17)**

31	28	27	24	23	20	19	16
PINMUX17_31_28		PINMUX17_27_24		PINMUX17_23_20		PINMUX17_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX17_15_12		PINMUX17_11_8		PINMUX17_7_4		PINMUX17_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-39. Pin Multiplexing Control 17 Register (PINMUX17) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX17_31_28	0	<b>GP7[2]/BOOT[2] Control</b> Selects Function BOOT[2]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[2]	I/O
		9h-Fh	<i>Reserved</i>	X
27-24	PINMUX17_27_24	0	<b>GP7[3]/BOOT[3] Control</b> Selects Function BOOT[3]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[3]	I/O
		9h-Fh	<i>Reserved</i>	X
23-20	PINMUX17_23_20	0	<b>GP7[4]/BOOT[4] Control</b> Selects Function BOOT[4]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[4]	I/O
		9h-Fh	<i>Reserved</i>	X
19-16	PINMUX17_19_16	0	<b>GP7[5]/BOOT[5] Control</b> Selects Function BOOT[5]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[5]	I/O
		9h-Fh	<i>Reserved</i>	X
15-12	PINMUX17_15_12	0	<b>GP7[6]/BOOT[6] Control</b> Selects Function BOOT[6]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[6]	I/O
		9h-Fh	<i>Reserved</i>	X
11-8	PINMUX17_11_8	0	<b>GP7[7]/BOOT[7] Control</b> Selects Function BOOT[7]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[7]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX17_7_4	0	<b>GP7[8] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[8]	I/O
		9h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-39. Pin Multiplexing Control 17 Register (PINMUX17) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
3-0	PINMUX17_3_0		<b>GP7[9] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[9]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.19 Pin Multiplexing Control 18 Register (PINMUX18)

**Figure 10-36. Pin Multiplexing Control 18 Register (PINMUX18)**

31	28	27	24	23	20	19	16
PINMUX18_31_28		PINMUX18_27_24		PINMUX18_23_20		PINMUX18_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX18_15_12		PINMUX18_11_8		PINMUX18_7_4		PINMUX18_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-40. Pin Multiplexing Control 18 Register (PINMUX18) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX18_31_28	0	<b>GP8[10] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[10]	I/O
		9h-Fh	<i>Reserved</i>	X
27-24	PINMUX18_27_24	0	<b>GP8[11] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[11]	I/O
		9h-Fh	<i>Reserved</i>	X
23-20	PINMUX18_23_20	0	<b>GP8[12] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[12]	I/O
		9h-Fh	<i>Reserved</i>	X
19-16	PINMUX18_19_16	0	<b>GP8[13] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[13]	I/O
		9h-Fh	<i>Reserved</i>	X
15-12	PINMUX18_15_12	0	<b>GP8[14] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[14]	I/O
		9h-Fh	<i>Reserved</i>	X
11-8	PINMUX18_11_8	0	<b>GP8[15] Control</b> Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[15]	I/O
		9h-Fh	<i>Reserved</i>	X
7-4	PINMUX18_7_4	0	<b>GP7[0]/BOOT[0] Control</b> Selects Function BOOT[0]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[0]	I/O
		9h-Fh	<i>Reserved</i>	X

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-40. Pin Multiplexing Control 18 Register (PINMUX18) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
3-0	PINMUX18_3_0		<b>GP7[1]/BOOT[1] Control</b>	
		0	Selects Function BOOT[1]	I
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP7[1]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.10.20 Pin Multiplexing Control 19 Register (PINMUX19)

**Figure 10-37. Pin Multiplexing Control 19 Register (PINMUX19)**

31	28	27	24	23	20	19	16
PINMUX19_31_28		PINMUX19_27_24		PINMUX19_23_20		PINMUX19_19_16	
R/W-0		R/W-0		R/W-0		R/W-0	
15	12	11	8	7	4	3	0
PINMUX19_15_12		PINMUX19_11_8		PINMUX19_7_4		PINMUX19_3_0	
R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-41. Pin Multiplexing Control 19 Register (PINMUX19) Field Descriptions**

Bit	Field	Value	Description	Type <sup>(1)</sup>
31-28	PINMUX19_31_28		<b>GP8[0] Control.</b> GP8[0] is initially configured as a reserved function after reset and will not be in a predictable state. This signal will only be stable after the GPIO configuration for this pin has been completed. You should carefully consider the system implications of this pin being in an unknown state after reset.	
		0-7h	Reserved	X
		8h	Selects Function GP8[0]	I/O
		9h-Fh	Reserved	X
27-24	PINMUX19_27_24		<b>GP6[0] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	Reserved	X
		8h	Selects Function GP6[0]	I/O
9h-Fh	Reserved	X		
23-20	PINMUX19_23_20		<b>GP6[1] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	Reserved	X
		8h	Selects Function GP6[1]	I/O
9h-Fh	Reserved	X		
19-16	PINMUX19_19_16		<b>GP6[2] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	Reserved	X
		8h	Selects Function GP6[2]	I/O
9h-Fh	Reserved	X		
15-12	PINMUX19_15_12		<b>GP6[3] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	Reserved	X
		8h	Selects Function GP6[3]	I/O
9h-Fh	Reserved	X		
11-8	PINMUX19_11_8		<b>GP6[4] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	Reserved	X
		8h	Selects Function GP6[4]	I/O
9h-Fh	Reserved	X		

<sup>(1)</sup> I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

**Table 10-41. Pin Multiplexing Control 19 Register (PINMUX19) Field Descriptions (continued)**

Bit	Field	Value	Description	Type <sup>(1)</sup>
7-4	PINMUX19_7_4		<b>GP8[8] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[8]	I/O
		9h-Fh	<i>Reserved</i>	X
3-0	PINMUX19_3_0		<b>GP8[9] Control</b>	
		0	Pin is 3-stated.	Z
		1h-7h	<i>Reserved</i>	X
		8h	Selects Function GP8[9]	I/O
		9h-Fh	<i>Reserved</i>	X

### 10.5.11 Suspend Source Register (SUSPSRC)

The suspend source register (SUSPSRC) indicates the emulation suspend source for those peripherals that support emulation suspend. A value of 1 (default) for a SUSPSRC bit corresponding to the peripheral, indicates that the DSP emulator controls the peripheral's emulation suspend signal. You should maintain this register with its default values.

The SUSPSRC is shown in [Figure 10-38](#) and described in [Table 10-42](#).

**Figure 10-38. Suspend Source Register (SUSPSRC)**

31	30	29	28	27	26	25	24
Reserved	Reserved	Reserved	TIMER64P_1SRC	TIMER64P_0SRC	Reserved	Reserved	EPWM1SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
23	22	21	20	19	18	17	16
EPWM0SRC	SPI1SRC	Reserved	Reserved	Reserved	UART0SRC	Reserved	I2C0SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
Reserved	Reserved	Reserved	HPI1SRC	Reserved	Reserved	Reserved	MCBSP1SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	ECAP2SRC	ECAP1SRC	ECAP0SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-42. Suspend Source Register (SUSPSRC) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
28	TIMER64P_1SRC	0	<b>Timer1 64 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.
27	TIMER64P_0SRC	0	<b>Timer0 64 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.
26-25	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
24	EPWM1SRC	0	<b>EPWM1 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.
23	EPWM0SRC	0	<b>EPWM0 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.
22	SPI1SRC	0	<b>SPI1 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.
21-19	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
18	UART0SRC	0	<b>UART0 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.
17	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
16	I2C0SRC	0	<b>I2C0 Emulation Suspend Source.</b> No emulation suspend.
		1	DSP is the source of the emulation suspend.



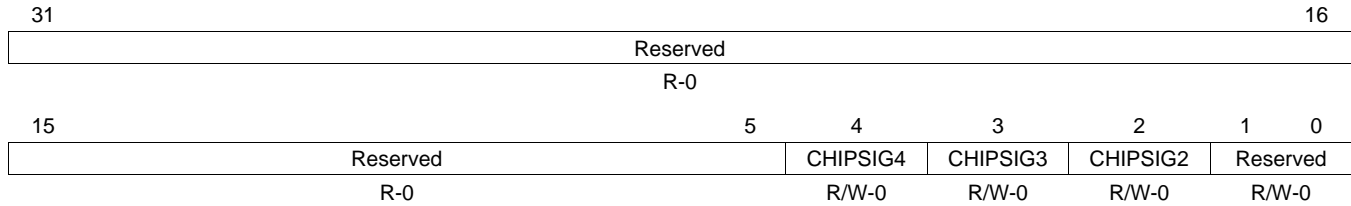
**Table 10-42. Suspend Source Register (SUSPSRC) Field Descriptions (continued)**

Bit	Field	Value	Description
15-13	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
12	HPISRC	0 1	<b>HPI Emulation Suspend Source.</b> 0 No emulation suspend. 1 DSP is the source of the emulation suspend.
11-9	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
8	MCBSP1SRC	0 1	<b>McBSP1 Emulation Suspend Source.</b> 0 No emulation suspend. 1 DSP is the source of the emulation suspend.
7-3	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
2	ECAP2SRC	0 1	<b>ECAP2 Emulation Suspend Source.</b> 0 No emulation suspend. 1 DSP is the source of the emulation suspend.
1	ECAP1SRC	0 1	<b>ECAP1 Emulation Suspend Source.</b> 0 No emulation suspend. 1 DSP is the source of the emulation suspend.
0	ECAP0SRC	0 1	<b>ECAP0 Emulation Suspend Source.</b> 0 No emulation suspend. 1 DSP is the source of the emulation suspend.

### 10.5.12 Chip Signal Register (CHIPSIG)

The interrupts to the DSP can be generated by setting one of the two CHIPSIG[3-2] bits or an NMI interrupt by setting the CHIPSIG[4] bit in the chip signal register (CHIPSIG). Writing a 1 to these bits sets the interrupts, writing a 0 has no effect. Reads return the value of these bits and can also be used as status bits. The CHIPSIG is shown in [Figure 10-39](#) and described in [Table 10-43](#).

**Figure 10-39. Chip Signal Register (CHIPSIG)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-43. Chip Signal Register (CHIPSIG) Field Descriptions**

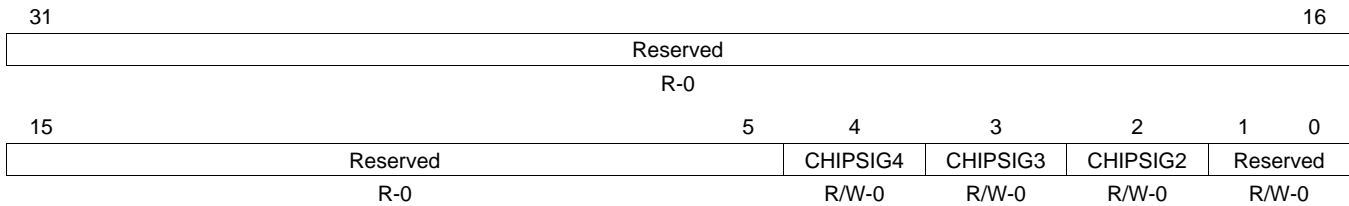
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	CHIPSIG4	0 1	<b>Asserts DSP NMI interrupt.</b> No effect Asserts interrupt
3	CHIPSIG3	0 1	<b>Asserts SYSCFG_CHIPINT3 interrupt.</b> No effect Asserts interrupt
2	CHIPSIG2	0 1	<b>Asserts SYSCFG_CHIPINT2 interrupt.</b> No effect Asserts interrupt
1-0	Reserved	0	Reserved. Write the default value to all bits when modifying this register.

### 10.5.13 Chip Signal Clear Register (CHIPSIG\_CLR)

The chip signal clear register (CHIPSIG\_CLR) is used to clear the bits set in the chip signal register (CHIPSIG). Writing a 1 to a CHIPSIG[*n*] bit in CHIPSIG\_CLR clears the corresponding CHIPSIG[*n*] bit in CHIPSIG; writing a 0 has no effect. After servicing the interrupt, the interrupted processor can clear the bits set in CHIPSIG by writing 1 to the corresponding bits in CHIPSIG\_CLR. The other processor may poll the CHIPSIG[*n*] bit to determine when the interrupted processor has completed the interrupt service. The CHIPSIG\_CLR is shown in Figure 10-40 and described in Table 10-44.

For more information on DSP interrupts, see the *DSP Subsystem* chapter.

**Figure 10-40. Chip Signal Clear Register (CHIPSIG\_CLR)**



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 10-44. Chip Signal Clear Register (CHIPSIG\_CLR) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	CHIPSIG4	0	No effect
		1	Clears interrupt
3	CHIPSIG3	0	No effect
		1	Clears interrupt
2	CHIPSIG2	0	No effect
		1	Clears interrupt
1-0	Reserved	0	Reserved. Write the default value to all bits when modifying this register.

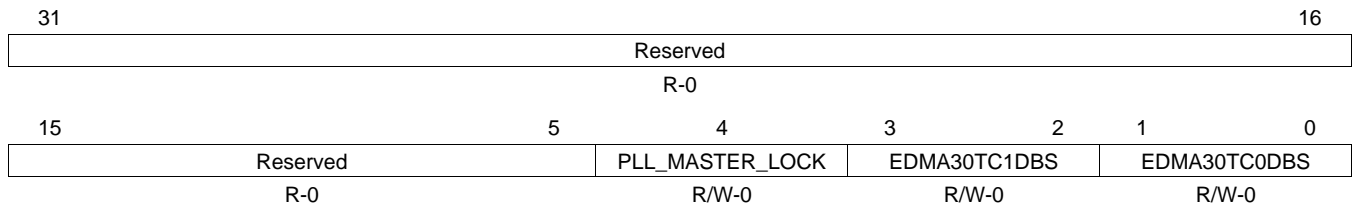
### 10.5.14 Chip Configuration 0 Register (CFGCHIP0)

The chip configuration 0 register (CFGCHIP0) controls the following functions:

- PLL Controller 0 memory-mapped register lock: Used to lock out writes to the PLLC0 memory-mapped registers (MMRs) to prevent any erroneous writes in software to the PLLC0 register space.
- EDMA3\_0 Transfer Controller Default Burst Size (DBS) Control: This controls the maximum number of bytes issued per read/write command or the burst size for the individual transfer controllers (TCs) on the device. By default for all transfer controllers, the burst size is set to 16 bytes. However, CFGCHIP0 allows configurability of this parameter so that the TC can have a burst size of 16, 32, or 64 bytes. The burst size determines the intra packet efficiency for the EDMA3\_0 transfers. Additionally, it also facilitates preemption at a system level, as all transfer requests are internally broken down by the transfer controller up to DBS size byte chunks and on a system level, each master's priority (configured by the MSTPRI register) is evaluated at burst size boundaries. The DBS value can significantly impact the standalone throughput performance depending on the source and destination (bus width/frequency/burst support etc) and the TC FIFO size, etc. Therefore, the DBS size configuration should be carefully analyzed to meet the system's throughput/performance requirements.

The CFGCHIP0 is shown in [Figure 10-41](#) and described in [Table 10-45](#).

**Figure 10-41. Chip Configuration 0 Register (CFGCHIP0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-45. Chip Configuration 0 Register (CFGCHIP0) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4	PLL_MASTER_LOCK	0	<b>PLLC0 MMRs lock.</b> PLLC0 MMRs are freely accessible.
		1	All PLLC0 MMRs are locked.
3-2	EDMA30TC1DBS	0	<b>EDMA3_0_TC1 Default Burst Size (DBS).</b> 16 bytes
		1h	32 bytes
		2h	64 bytes
		3h	Reserved
1-0	EDMA30TC0DBS	0	<b>EDMA3_0_TC0 Default Burst Size (DBS).</b> 16 bytes
		1h	32 bytes
		2h	64 bytes
		3h	Reserved

### 10.5.15 Chip Configuration 1 Register (CFGCHIP1)

The chip configuration 1 register (CFGCHIP1) controls the following functions:

- eCAP0/1/2 event input source: Allows using McASP0 TX/RX events as eCAP event input sources.
- EDMA3\_1 Transfer Controller Default Burst Size (DBS) Control: This controls the maximum number of bytes issued per read/write command or the burst size for the individual transfer controllers (TCs) on the device. By default for all transfer controllers, the burst size is set to 16 bytes. However, CFGCHIP1 allows configurability of this parameter so that the TC can have a burst size of 16, 32, or 64 bytes. The burst size determines the intra packet efficiency for the EDMA3\_1 transfers. Additionally, it also facilitates preemption at a system level, as all transfer requests are internally broken down by the transfer controller up to DBS size byte chunks and on a system level, each master's priority (configured by the MSTPRI register) is evaluated at burst size boundaries. The DBS value can significantly impact the standalone throughput performance depending on the source and destination (bus width/frequency/burst support etc) and the TC FIFO size, etc. Therefore, the DBS size configuration should be carefully analyzed to meet the system's throughput/performance requirements.
- eHRPWM Time Base Clock (TBCLK) Synchronization: Allows the software to globally synchronize all enabled eHRPWM modules to the time base clock (TBCLK).
- McASP0 AMUTEIN signal source control: Allows selecting GPIO interrupt from different banks as source for the McASP0 AMUTEIN signal.

The CFGCHIP1 is shown in [Figure 10-42](#) and described in [Table 10-46](#).

**Figure 10-42. Chip Configuration 1 Register (CFGCHIP1)**

31	27	26	22	21	17	16
CAP2SRC		CAP1SRC		CAP0SRC		HPIBYTEAD
R/W-0		R/W-0		R/W-0		R/W-0
15	14	13	12	11		
HPIENA	EDMA31TC0DBS		TBCLKSYNC	Reserved		
R/W-0	R/W-0		R/W-0	R/W-0		
7				4	3	0
Reserved				AMUTESEL0		
R/W-0				R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 10-46. Chip Configuration 1 Register (CFGCHIP1) Field Descriptions**

Bit	Field	Value	Description
31-27	CAP2SRC	0	Selects the eCAP2 module event input. eCAP2 Pin input
		1h	McASP0 TX DMA Event
		2h	McASP0 RX DMA Event
		3h-1Fh	Reserved
26-22	CAP1SRC	0	Selects the eCAP1 module event input. eCAP1 Pin input
		1h	McASP0 TX DMA Event
		2h	McASP0 RX DMA Event
		3h-1Fh	Reserved
21-17	CAP0SRC	0	Selects the eCAP0 module event input. eCAP0 Pin input
		1h	McASP0 TX DMA Event
		2h	McASP0 RX DMA Event
		3h-1Fh	Reserved

**Table 10-46. Chip Configuration 1 Register (CFGCHIP1) Field Descriptions (continued)**

Bit	Field	Value	Description
16	HPIBYTEAD	0	Host address is a word address.
		1	Host address is a byte address.
15	HPIENA	0	HPI is disabled.
		1	HPI is enabled.
14-13	EDMA31TC0DBS	0	16 bytes
		1h	32 bytes
		2h	64 bytes
		3h	<i>Reserved</i>
12	TBCLKSYNC	0	<b>eHRPWM Module Time Base Clock Synchronization.</b> Allows you to globally synchronize all enabled eHRPWM modules to the time base clock (TBCLK). Time base clock (TBCLK) within each enabled eHRPWM module is stopped.
		1	All enabled eHRPWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each eHRPWM module must be set identically.
11-4	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
3-0	AMUTESELO	0	Drive McASP0 AMUTEIN signal low.
		1h	GPIO Interrupt from Bank 0
		2h	GPIO Interrupt from Bank 1
		3h	GPIO Interrupt from Bank 2
		4h	GPIO Interrupt from Bank 3
		5h	GPIO Interrupt from Bank 4
		6h	GPIO Interrupt from Bank 5
		7h	GPIO Interrupt from Bank 6
		8h	GPIO Interrupt from Bank 7
		9h-Fh	<i>Reserved</i>

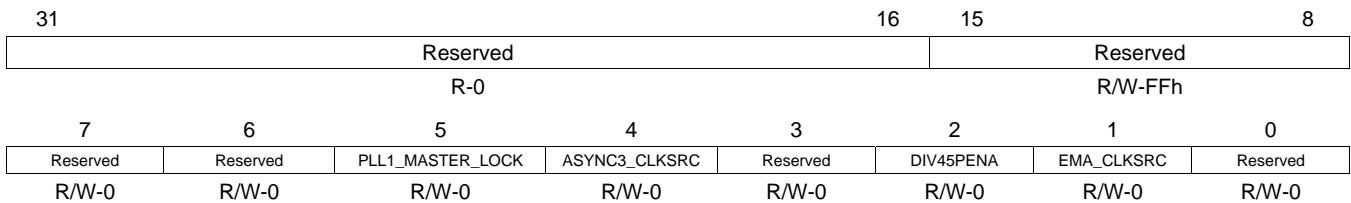
### 10.5.16 Chip Configuration 3 Register (CFGCHIP3)

The chip configuration 3 register (CFGCHIP3) controls the following peripheral/module functions:

- PLL Controller 1 memory-mapped register lock: Used to lock out writes to the PLLC1 memory-mapped registers (MMRs) to prevent any erroneous writes in software to the PLLC1 register space.
- ASYNC3 Clock Source Control: Allows control for the source of the ASYNC3 clock.
- DIV4p5 Clock Enable/Disable: The DIV4p5 (/4.5) hardware clock divider is provided to generate 133 MHz from the 600 MHz PLL clock for use as clocks to the EMIFs. Allows enabling/disabling this clock divider.
- EMIFA Module Clock Source Control: Allows control for the source of the EMIFA module clock.

The CFGCHIP3 is shown in [Figure 10-43](#) and described in [Table 10-47](#).

**Figure 10-43. Chip Configuration 3 Register (CFGCHIP3)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

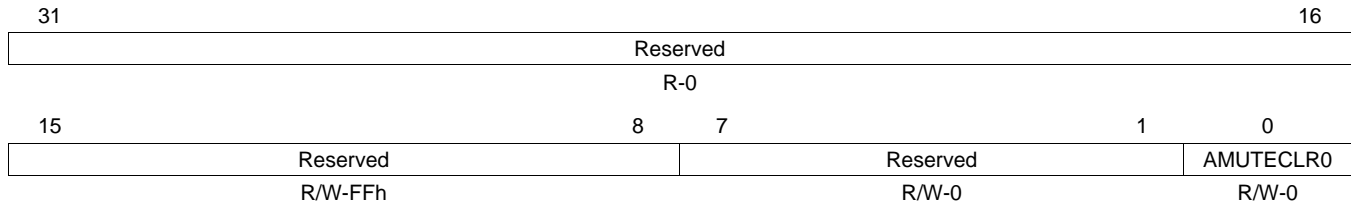
**Table 10-47. Chip Configuration 3 Register (CFGCHIP3) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	Reserved	FFh	Reserved. Write the default value to all bits when modifying this register.
7-6	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
5	PLL1_MASTER_LOCK	0	<b>PLLC1 MMRs lock.</b> PLLC1 MMRs are freely accessible.
		1	All PLLC1 MMRs are locked.
4	ASYNC3_CLKSRC	0	<b>Clock source for ASYNC3.</b> Clock driven by PLL0_SYSCLK2.
		1	Clock driven by PLL1_SYSCLK2.
3	Reserved	0	Reserved. Write the default value when modifying this register.
2	DIV45PENA	0	<b>Controls the fixed DIV4.5 divider in the PLL controller.</b> Divide by 4.5 is disabled.
		1	Divide by 4.5 is enabled.
1	EMA_CLKSRC	0	<b>Clock source for EMIFA clock domain.</b> Clock driven by PLL0_SYSCLK3
		1	Clock driven by DIV4.5 PLL output
0	Reserved	0	Reserved. Write the default value when modifying this register.

### 10.5.17 Chip Configuration 4 Register (CFGCHIP4)

The chip configuration 4 register (CFGCHIP4) is used for clearing the AMUNTEIN signal for McASP0. Writing a 1 causes a single pulse that clears the latched GPIO interrupt for AMUTEIN of McASP0, if it was previously set; reads always return a value of 0. The CFGCHIP4 is shown in Figure 10-44 and described in Table 10-48.

**Figure 10-44. Chip Configuration 4 Register (CFGCHIP4)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

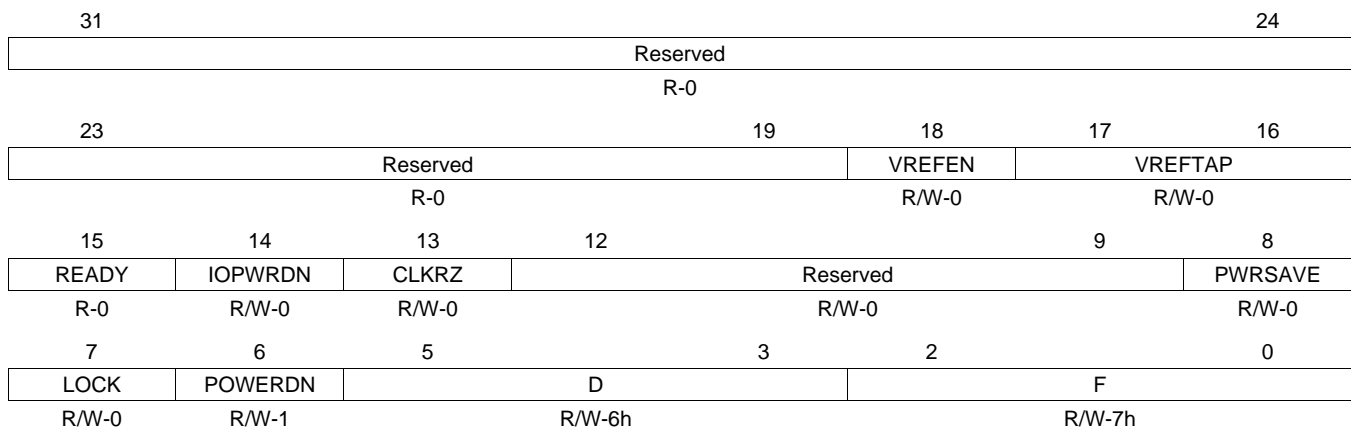
**Table 10-48. Chip Configuration 4 Register (CFGCHIP4) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	Reserved	FFh	Reserved. Write the default value to all bits when modifying this register.
7-1	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
0	AMUTECLR0	0	<b>Clears the latched GPIO interrupt for AMUTEIN of McASP0 when set to 1.</b> No effect
		1	Clears interrupt

### 10.5.18 VTP I/O Control Register (VTPIO\_CTL)

The VTP I/O control register (VTPIO\_CTL) is used to control the calibration of the DDR2/mDDR memory controller I/Os with respect to voltage, temperature, and process (VTP). The voltage, temperature, and process information is used to control the IO's output impedance. The VTPIO\_CTL is shown in Figure 10-45 and described in Table 10-49.

**Figure 10-45. VTP I/O Control Register (VTPIO\_CTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset



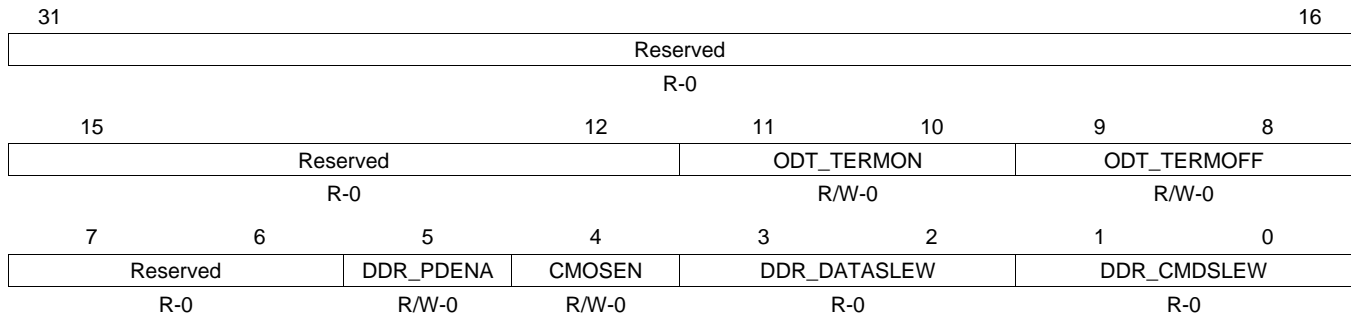
**Table 10-49. VTP I/O Control Register (VTPIO\_CTL) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	VREFEN	0 1	<b>Internal DDR I/O Vref enable.</b> 0 Connected to pad, external reference. 1 <i>Reserved</i>
17-16	VREFTAP	0 1h-3h	<b>Selection for internal reference voltage level.</b> 0 Vref = 50.0% of VDD5 1h-3h <i>Reserved</i>
15	READY	0 1	<b>VTP Ready status.</b> 0 VTP is not ready. 1 VTP is ready.
14	IOPWRDN	0 1	<b>Power down enable for DDR input buffer.</b> 0 Disable power down control by the PWRDNEN bit in the DDR PHY control register 1 (DRPYC1R). 1 Enable power down control by the PWRDNEN bit in the DDR PHY control register 1 (DRPYC1R).
13	CLKRZ	0	<b>VTP clear.</b> Write 0 to clear VTP flops.
12-9	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
8	PWRSAVE	0 1	<b>VTP power save mode.</b> Turn off power to the external resistor when it is not needed. The PWRSAVE bit setting is only valid when the POWERDN bit is cleared to 0. 0 Disable power save mode. 1 Enable power save mode.
7	LOCK	0 1	<b>VTP impedance lock.</b> Lock impedance value so that the VTP controller can be powered down. 0 Unlock impedance. 1 Lock impedance.
6	POWERDN	0 1	<b>VTP power down.</b> Power down the VTP controller. The PWRSAVE bit setting is only valid when the POWERDN bit is cleared to 0. 0 Disable power down. 1 Enable power down.
5-3	D	0-5h 6h 7h	<b>Drive strength control bit.</b> 0-5h <i>Reserved</i> 6h 100% drive strength 7h <i>Reserved</i>
2-0	F	0-6h 7h	<b>Digital filter control bit.</b> 0-6h <i>Reserved</i> 7h Digital filter is enabled.

### 10.5.19 DDR Slew Register (DDR\_SLEW)

The DDR slew register (DDR\_SLEW) reflects the DDR I/O timing as programmed in the device eFuse. The CMOSEN field configures the DDR I/O cells into an LVCMOS buffer (this makes it mDDR compatible). The DDR\_SLEW is shown in [Figure 10-46](#) and described in [Table 10-50](#).

**Figure 10-46. DDR Slew Register (DDR\_SLEW)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

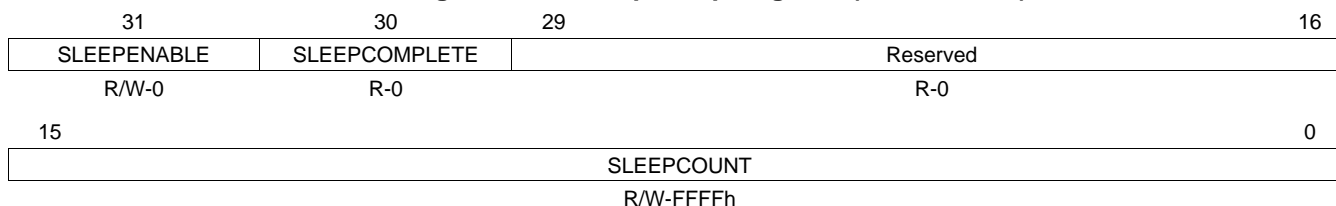
**Table 10-50. DDR Slew Register (DDR\_SLEW) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-10	ODT_TERMON	0 <i>1h-3h</i>	<b>Controls Thevenin termination mode while I/O is in read or write mode.</b> Termination is not supported on this device. No termination <i>Reserved</i>
9-8	ODT_TERMOFF	0 <i>1h-3h</i>	<b>Controls Thevenin termination mode while I/O is not in read or write mode.</b> Termination is not supported on this device. No termination <i>Reserved</i>
7-6	Reserved	0	Reserved
5	DDR_PDENA	0 1	<b>Enables pull downs for mDDR mode (should be disabled for DDR2).</b> 0 Pull downs are disabled. Disable pull downs when using DDR2. 1 Pull downs are enabled. Enable pull downs when using mDDR.
4	CMOSEN	0 1	<b>Selects mDDR LVCMOS RX / SSTL18 differential RX.</b> 0 SSTL Receiver. Select SSTL when using DDR2. 1 LVCMOS Receiver. Select LVCMOS when using mDDR.
3-2	DDR_DATASLEW	0 <i>1h-3h</i>	<b>Slew rate mode control status for data macro.</b> Slew rate control is not supported on this device. 0 Slew rate control is off. <i>Reserved</i>
1-0	DDR_CMDSLEW	0 <i>1h-3h</i>	<b>Slew rate mode control status for command macro.</b> Slew rate control is not supported on this device. 0 Slew rate control is off. <i>Reserved</i>

### 10.5.20 Deep Sleep Register (DEEPSLEEP)

The deep sleep register (DEEPSLEEP) control the Deep Sleep logic. See your device-specific data manual and the *Boot Considerations* chapter for details on boot and configuration settings. The DEEPSLEEP is shown in [Figure 10-47](#) and described in [Table 10-51](#).

**Figure 10-47. Deep Sleep Register (DEEPSLEEP)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

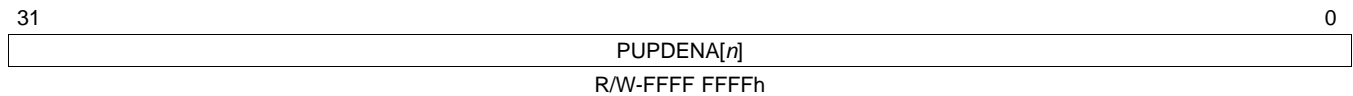
**Table 10-51. Deep Sleep Register (DEEPSLEEP) Field Descriptions**

Bit	Field	Value	Description
31	SLEEPENABLE	0	<b>Deep sleep enable.</b> The software must clear this bit to 0 when the device is awakened from deep sleep.
		1	Device is in normal operating mode; <u>DEEPSLEEP</u> pin has no effect.
			Deep sleep mode is enabled; setting <u>DEEPSLEEP</u> pin low initiates oscillator shut down.
30	SLEEPCOMPLETE	0	<b>Deep sleep complete.</b> Once the deep sleep process starts, the software must poll the SLEEPCOMPLETE bit; when the SLEEPCOMPLETE bit is read as 1, the software should clear the SLEEPENABLE bit and continue operation.
		1	SLEEPCOUNT delay is not complete.
			SLEEPCOUNT delay is complete.
29-16	Reserved	0	Reserved
15-0	SLEEPCOUNT	0-FFFFh	<b>Deep sleep counter.</b> Number of cycles to count prior to the oscillator being stable. All 16 bits are tied directly to the counter in the Deep Sleep logic.

### 10.5.21 Pullup/Pulldown Enable Register (PUPD\_ENA)

The pullup/pulldown enable register (PUPD\_ENA) enables the pull-up or pull-down functionality for the pin group *n* defined in your device-specific data manual. The PUPD\_ENA is shown in [Figure 10-48](#) and described in [Table 10-52](#).

**Figure 10-48. Pullup/Pulldown Enable Register (PUPD\_ENA)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-52. Pullup/Pulldown Enable Register (PUPD\_ENA) Field Descriptions**

Bit	Field	Value	Description
31-0	PUPDENA[n]		<b>Enables internal pull-up or pull-down functionality for pin group CP[n].</b> See your device-specific data manual for pin group information. The internal pull-up or pull-down functionality selection for bit position <i>n</i> in PUPD_ENA is set in the same bit position <i>n</i> of the pullup/pulldown select register (PUPD_SEL).
		0	Internal pull-up or pull-down functionality for pin group <i>n</i> is disabled.
		1	Internal pull-up or pull-down functionality for pin group <i>n</i> is enabled.

### 10.5.22 Pullup/Pulldown Select Register (PUPD\_SEL)

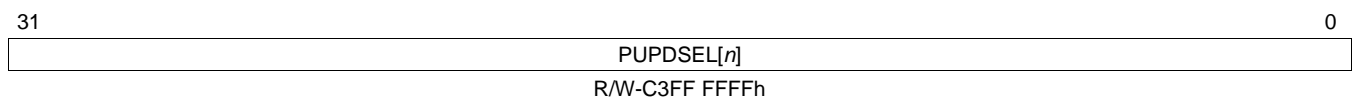
The pullup/pulldown select register (PUPD\_SEL) selects between the pull-up or pull-down functionality for the pin group *n* defined in your device-specific data manual. The PUPD\_SEL is shown in [Figure 10-49](#) and described in [Table 10-53](#) and [Table 10-54](#).

---

**NOTE:** The PUPD\_SEL settings are not active until the device is out of reset. During reset, all of the CP[n] pins are pulled down. If the application requires a pull-up during reset, an external pull-up should be used.

---

**Figure 10-49. Pullup/Pulldown Select Register (PUPD\_SEL)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 10-53. Pullup/Pulldown Select Register (PUPD\_SEL) Field Descriptions**

Bit	Field	Value	Description
31-0	PUPDSEL[n]		<b>Selects between the internal pull-up or pull-down functionality for pin group CP[n].</b> See your device-specific data manual for pin group information. The selection for bit position <i>n</i> in PUPD_SEL is only valid when the same bit position <i>n</i> is set in the pullup/pulldown enable register (PUPD_ENA).
		0	Internal pull-down functionality for pin group <i>n</i> is disabled.
		1	Internal pull-up functionality for pin group <i>n</i> is enabled.

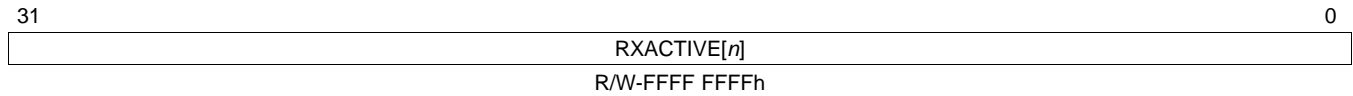
**Table 10-54. Pullup/Pulldown Select Register (PUPD\_SEL) Default Values**

Bit	Field	Default Value	Description
31	PUPDSEL[31]	1	Pin Group CP[31] is configured for pull-up by default.
30	PUPDSEL[30]	1	Pin Group CP[30] is configured for pull-up by default.
29	PUPDSEL[29]	0	Pin Group CP[29] is configured for pull-down by default.
28	PUPDSEL[28]	0	Pin Group CP[28] is configured for pull-down by default.
27	PUPDSEL[27]	0	Pin Group CP[27] is configured for pull-down by default.
26	PUPDSEL[26]	0	Pin Group CP[26] is configured for pull-down by default.
25	PUPDSEL[25]	1	Pin Group CP[25] is configured for pull-up by default.
24	PUPDSEL[24]	1	Pin Group CP[24] is configured for pull-up by default.
23	PUPDSEL[23]	1	Pin Group CP[23] is configured for pull-up by default.
22	PUPDSEL[22]	1	Pin Group CP[22] is configured for pull-up by default.
21	PUPDSEL[21]	1	Pin Group CP[21] is configured for pull-up by default.
20	PUPDSEL[20]	1	Pin Group CP[20] is configured for pull-up by default.
19	PUPDSEL[19]	1	Pin Group CP[19] is configured for pull-up by default.
18	PUPDSEL[18]	1	Pin Group CP[18] is configured for pull-up by default.
17	PUPDSEL[17]	1	Pin Group CP[17] is configured for pull-up by default.
16	PUPDSEL[16]	1	Pin Group CP[16] is configured for pull-up by default.
15	PUPDSEL[15]	1	Pin Group CP[15] is configured for pull-up by default.
14	PUPDSEL[14]	1	Pin Group CP[14] is configured for pull-up by default.
13	PUPDSEL[13]	1	Pin Group CP[13] is configured for pull-up by default.
12	PUPDSEL[12]	1	Pin Group CP[12] is configured for pull-up by default.
11	PUPDSEL[11]	1	Pin Group CP[11] is configured for pull-up by default.
10	PUPDSEL[10]	1	Pin Group CP[10] is configured for pull-up by default.
9	PUPDSEL[9]	1	Pin Group CP[9] is configured for pull-up by default.
8	PUPDSEL[8]	1	Pin Group CP[8] is configured for pull-up by default.
7	PUPDSEL[7]	1	Pin Group CP[7] is configured for pull-up by default.
6	PUPDSEL[6]	1	Pin Group CP[6] is configured for pull-up by default.
5	PUPDSEL[5]	1	Pin Group CP[5] is configured for pull-up by default.
4	PUPDSEL[4]	1	Pin Group CP[4] is configured for pull-up by default.
3	PUPDSEL[3]	1	Pin Group CP[3] is configured for pull-up by default.
2	PUPDSEL[2]	1	Pin Group CP[2] is configured for pull-up by default.
1	PUPDSEL[1]	1	Pin Group CP[1] is configured for pull-up by default.
0	PUPDSEL[0]	1	Pin Group CP[0] is configured for pull-up by default.

### 10.5.23 RXACTIVE Control Register (RXACTIVE)

The RXACTIVE control register (RXACTIVE) enables or disables the LVCMOS receivers for the pin group  $n$  defined in your device-specific data manual. The RXACTIVE is shown in [Figure 10-50](#) and described in [Table 10-55](#).

**Figure 10-50. RXACTIVE Control Register (RXACTIVE)**



LEGEND: R/W = Read/Write; - $n$  = value after reset

**Table 10-55. RXACTIVE Control Register (RXACTIVE) Field Descriptions**

Bit	Field	Value	Description
31-0	RXACTIVE[ $n$ ]		<b>Enables the LVCMOS receivers on pin group <math>n</math>.</b> See your device-specific data manual for pin group information. Receivers should only be disabled if the associated pin group is not being used.
		0	LVCMOS receivers for pin group $n$ are disabled.
		1	LVCMOS receivers for pin group $n$ are enabled.

## ***Boot Considerations***

---

---

---

Topic	Page
11.1 Introduction .....	237

## 11.1 Introduction

This device supports a variety of boot modes through an internal DSP ROM bootloader. This device does not support dedicated hardware boot modes; therefore, all boot modes utilize the internal DSP ROM. The input states of the BOOT pins are sampled and latched into the BOOTCFG register, which is part of the system configuration (SYSCFG) module, when device reset is deasserted. Boot mode selection is determined by the values of the BOOT pins.

The following boot modes are supported:

- NAND Flash boot
  - 8-bit NAND
  - 16-bit NAND
- NOR Flash boot
  - NOR Direct boot (8-bit or 16-bit)
  - NOR Legacy boot (8-bit or 16-bit)
  - NOR AIS boot (8-bit or 16-bit)
- HPI boot
- I2C0 boot
  - EEPROM (Master Mode)
  - External Host (Slave Mode)
- SPI1 boot
  - Serial Flash (Master Mode)
  - Serial EEPROM (Master Mode)
  - External Host (Slave Mode)
- UART0 boot
  - External Host

See *Using the TMS320C6748/C6746/C6742 Bootloader Application Report (SPRAAT2)* for more details on the ROM Boot Loader, a list of boot pins used, and the complete list of supported boot modes.



## DDR2/mDDR Memory Controller

---

---

---

This chapter describes the DDR2/mobile DDR (mDDR) memory controller.

Topic	Page
<b>12.1 Introduction</b> .....	<b>239</b>
<b>12.2 Architecture</b> .....	<b>241</b>
<b>12.3 Supported Use Cases</b> .....	<b>269</b>
<b>12.4 Registers</b> .....	<b>274</b>

## 12.1 Introduction

### 12.1.1 Purpose of the Peripheral

The DDR2/mDDR memory controller is used to interface with JESD79D-2 standard compliant DDR2 SDRAM devices and JESD209 standard mobile DDR (mDDR) SDRAM devices. Memory types such as DDR1 SDRAM, SDR SDRAM, SBSRAM, and asynchronous memories are not supported. The DDR2/mDDR memory is the major memory location for program and data storage.

### 12.1.2 Features

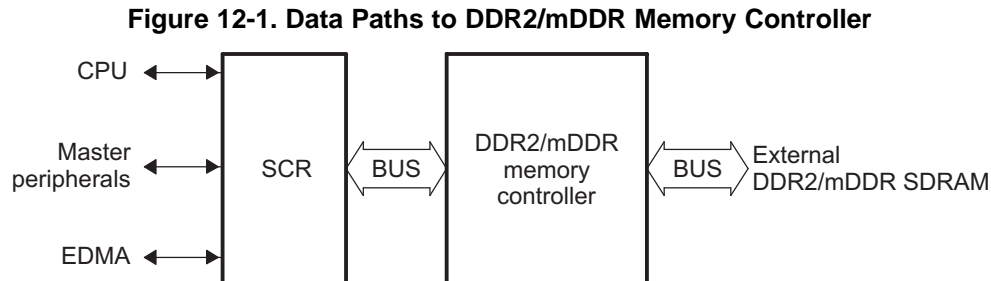
The DDR2/mDDR memory controller supports the following features:

- JESD79D-2 standard compliant DDR2 SDRAM
- JESD209 standard compliant mobile DDR (mDDR)
- Data bus width of 16 bits
- CAS latencies:
  - DDR2: 2, 3, 4, and 5
  - mDDR: 2 and 3
- Internal banks:
  - DDR2: 1, 2, 4, and 8
  - mDDR: 1, 2, and 4
- Burst length: 8
- Burst type: sequential
- 1 CS signal
- Page sizes: 256, 512, 1024, and 2048
- SDRAM auto-initialization
- Self-refresh mode
- Partial array self-refresh (for mDDR)
- Power-down mode
- Prioritized refresh
- Programmable refresh rate and backlog counter
- Programmable timing parameters
- Little-endian mode

### 12.1.3 Functional Block Diagram

The DDR2/mDDR memory controller is the main interface to external DDR2/mDDR memory. [Figure 12-1](#) displays the general data paths to on-chip peripherals and external DDR2/mDDR SDRAM.

Master peripherals, EDMA, and the CPU can access the DDR2/mDDR memory controller through the switched central resource (SCR).



### 12.1.4 Supported Use Case Statement

The DDR2/mDDR memory controller supports JESD79D-2 DDR2 SDRAM memories and the JESD209 mobile DDR (mDDR) SDRAM memories utilizing 16 bits of the DDR2/mDDR memory controller data bus. See [Section 12.3](#) for more details.

### 12.1.5 Industry Standard(s) Compliance Statement

The DDR2/mDDR memory controller is compliant with the JESD79D-2 DDR2 SDRAM standard and the JESD209 mobile DDR (mDDR) standard with the following exception:

- On-Die Termination (ODT). The DDR2/mDDR memory controller does not include any on-die terminating resistors. Furthermore, the on-die terminating resistors of the DDR2/mDDR SDRAM device must be disabled by tying the ODT input pin of the DDR2/mDDR SDRAM to ground.

## 12.2 Architecture

This section describes the architecture of the DDR2/mDDR memory controller as well as how it is structured and how it works within the context of the system-on-a-chip. The DDR2/mDDR memory controller can gluelessly interface to most standard DDR2/mDDR SDRAM devices and supports such features as self-refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to interface and properly configure the DDR2/mDDR memory controller to perform read and write operations to externally-connected DDR2/mDDR SDRAM devices. Also, [Section 12.3](#) provides a detailed example of interfacing the DDR2/mDDR memory controller to a common DDR2/mDDR SDRAM device.

### 12.2.1 Clock Control

The DDR2/mDDR memory controller receives two input clocks from internal clock sources, VCLK and 2X\_CLK ([Figure 12-2](#)). VCLK is a divided-down version of the PLL0 clock. 2X\_CLK is the PLL1 clock. 2X\_CLK should be configured to clock at the frequency of the desired data rate, or stated similarly, it should operate at twice the frequency of the desired DDR2/mDDR memory clock. DDR\_CLK and  $\overline{\text{DDR\_CLK}}$  are the two output clocks of the DDR2/mDDR memory controller providing the interface clock to the DDR2/mDDR SDRAM memory. These two clocks operate at a frequency of 2X\_CLK/2.

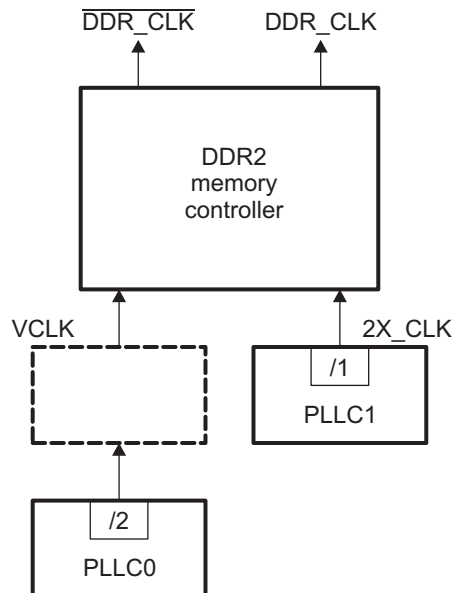
#### 12.2.1.1 Clock Source

VCLK and 2X\_CLK are sourced from two independent PLLs ([Figure 12-2](#)). VCLK is sourced from PLL controller 0 (PLL0) and 2X\_CLK is sourced from PLL controller 1 (PLL1).

VCLK is clocked at a fixed divider ratio of PLL0. This divider is fixed at 2, meaning VCLK is clocked at a frequency of PLL0/2.

The clock from PLL1 is not divided before reaching 2X\_CLK. PLL1 should be configured to supply 2X\_CLK at the desired frequency. For example, if a 138-MHz DDR2/mDDR interface clock (DDR\_CLK) is desired, then PLL1 must be configured to generate a 276-MHz clock on 2X\_CLK.

**Figure 12-2. DDR2/mDDR Memory Controller Clock Block Diagram**



### 12.2.1.2 Clock Configuration

The frequency of 2X\_CLK is configured by selecting the appropriate PLL multiplier. The PLL multiplier is selected by programming registers within PLLC1. The PLLC1 divider ration is fixed at 1. For information on programming the PLL controllers, see the *Phase-Locked Loop Controller (PLL)* chapter. For information on supported clock frequencies, see the *Device Clocking* chapter and your device-specific data manual.

---

**NOTE:** PLLC1 should be configured and a stable clock present on 2X\_CLK before releasing the DDR2/mDDR memory controller from reset.

---

### 12.2.1.3 DDR2/mDDR Memory Controller Internal Clock Domains

There are two clock domains within the DDR2/mDDR memory controller. The two clock domains are driven by VCLK and a divided-down by 2 version of 2X\_CLK called MCLK. The command FIFO, write FIFO, and read FIFO described in [Section 12.2.6](#) are all on the VCLK domain. From this, VCLK drives the interface to the peripheral bus.

The MCLK domain consists of the DDR2/mDDR memory controller state machine and memory-mapped registers. This clock domain is clocked at the rate of the external DDR2/mDDR memory, 2X\_CLK/2.

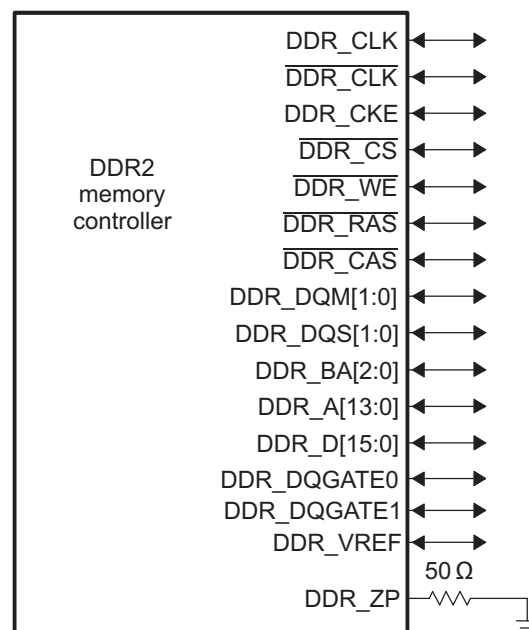
To conserve power within the DDR2/mDDR memory controller, VCLK, MCLK, and 2X\_CLK may be stopped. See [Section 12.2.16](#) for proper clock stop procedures.

## 12.2.2 Signal Descriptions

The DDR2/mDDR memory controller signals are shown in [Figure 12-3](#) and described in [DDR2/mDDR Memory Controller Signal Descriptions](#). The following features are included:

- The maximum data bus is 16-bits wide.
- The address bus is 14-bits wide with an additional three bank address pins.
- Two differential output clocks driven by internal clock sources.
- Command signals: Row and column address strobe, write enable strobe, data strobe, and data mask.
- One chip select signal and one clock enable signal.

**Figure 12-3. DDR2/mDDR Memory Controller Signals**



### DDR2/mDDR Memory Controller Signal Descriptions

Pin	Type <sup>(1)</sup>	Description
DDR_CLK, DDR_CLK	O/Z	<b>Clock:</b> Differential clock outputs.
DDR_CKE	O/Z	<b>Clock enable:</b> Active high.
DDR_CS	O/Z	<b>Chip select:</b> Active low.
DDR_WE	O/Z	<b>Write enable strobe:</b> Active low, command output.
DDR_RAS	O/Z	<b>Row address strobe:</b> Active low, command output.
DDR_CAS	O/Z	<b>Column address strobe:</b> Active low, command output.
DDR_DQM[1:0]	O/Z	<b>Data mask:</b> Active high, output mask signal for write data.
DDR_DQS[1:0]	I/O/Z	<b>Data strobe:</b> Active high, bi-directional signals. Output with write data, input with read data.
DDR_BA[2:0]	O/Z	<b>Bank select:</b> Output, defining which bank a given command is applied.
DDR_A[13:0]	O/Z	<b>Address:</b> Address bus.
DDR_D[15:0]	I/O/Z	<b>Data:</b> Bi-directional data bus. Input for read data, output for write data.
DDR_DQGATE0	O/Z	<b>Strobe Enable:</b> Active high.
DDR_DQGATE1	I/O/Z	<b>Strobe Enable Delay:</b> Loopback signal for timing adjustment (DQS gating). Route from DDR_DQGATE0 to DDR device and back to DDR_DQGATE1 with same constraints as used for DDR clock and data.
DDR_ZP	I/O/Z	<b>Output drive strength reference:</b> Reference output for drive strength calibration of N and P channel outputs. Tie to ground via 50 ohm .5% tolerance 1/16th watt resistor (49.9 ohm .5% tolerance is acceptable).
DDR_VREF	pwr	<b>Voltage reference input:</b> Voltage reference input for the SSTL_18 I/O buffers. Note even in the case of mDDR an external resistor divider connected to this pin is necessary.

<sup>(1)</sup> Legend: I = input, O = Output, Z = high impedance, pwr = power

### 12.2.3 Protocol Description(s)

The DDR2/mDDR memory controller supports the DDR2/mDDR SDRAM commands listed in [Table 12-1](#). [Table 12-2](#) shows the signal truth table for the DDR2/mDDR SDRAM commands.

**Table 12-1. DDR2/mDDR SDRAM Commands**

Command	Function
ACTV	Activates the selected bank and row.
DCAB	Precharge all command. Deactivates (precharges) all banks.
DEAC	Precharge single command. Deactivates (precharges) a single bank.
DESEL	Device Deselect.
EMRS	Extended Mode Register set. Allows altering the contents of the mode register.
MRS	Mode register set. Allows altering the contents of the mode register.
NOP	No operation.
Power Down	Power-down mode.
READ	Inputs the starting column address and begins the read operation.
READ with autoprecharge	Inputs the starting column address and begins the read operation. The read operation is followed by a precharge.
REFR	Autorefresh cycle.
SLFREFR	Self-refresh mode.
WRT	Inputs the starting column address and begins the write operation.
WRT with autoprecharge	Inputs the starting column address and begins the write operation. The write operation is followed by a precharge.

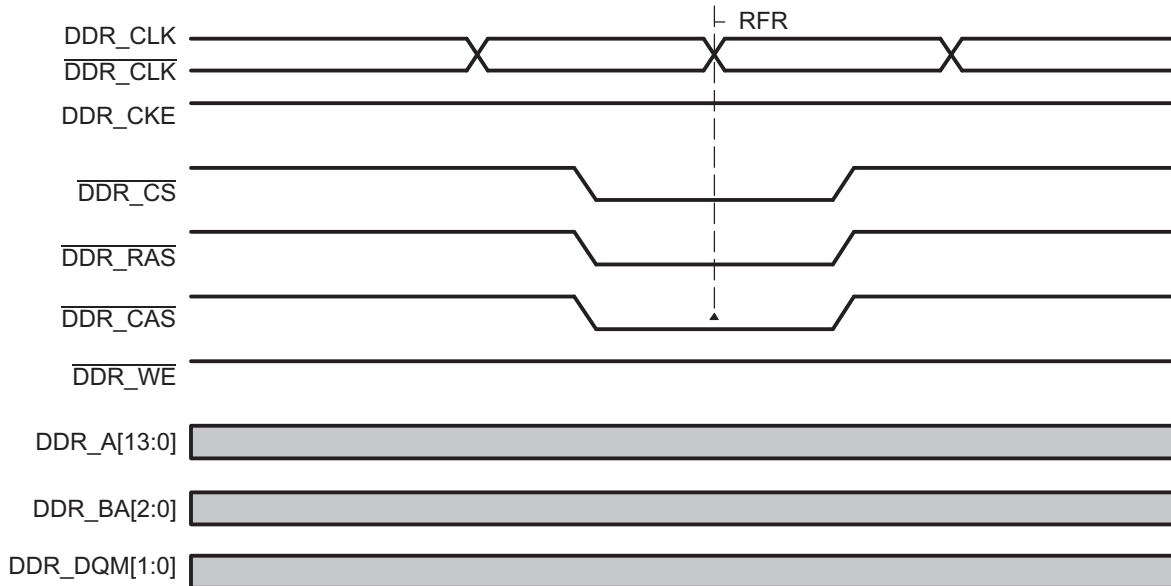
**Table 12-2. Truth Table for DDR2/mDDR SDRAM Commands**

DDR2/mDDR SDRAM:	CKE		$\overline{CS}$	RAS	$\overline{CAS}$	WE	BA[2:0]	A[13:11, 9:0]	A10
DDR2/mDDR memory controller:	DDR_CKE		$\overline{DDR\_CS}$	$\overline{DDR\_RAS}$	$\overline{DDR\_CAS}$	$\overline{DDR\_WE}$	DDR_BA[2:0]	DDR_A[13:11, 9:0]	DDR_A[10]
	Previous Cycles	Current Cycle							
ACTV	H	H	L	L	H	H	Bank	Row Address	
DCAB	H	H	L	L	H	L	X	X	H
DEAC	H	H	L	L	H	L	Bank	X	L
MRS	H	H	L	L	L	L	BA	OP Code	
EMRS	H	H	L	L	L	L	BA	OP Code	
READ	H	H	L	H	L	H	BA	Column Address	L
READ with precharge	H	H	L	H	L	H	BA	Column Address	H
WRT	H	H	L	H	L	L	BA	Column Address	L
WRT with precharge	H	H	L	H	L	L	BA	Column Address	H
REFR	H	H	L	L	L	H	X	X	X
SLFREFR entry	H	L	L	L	L	H	X	X	X
SLFREFR exit	L	H	H	X	X	X	X	X	X
			L	H	H	H	X	X	X
NOP	H	X	L	H	H	H	X	X	X
DESEL	H	X	H	X	X	X	X	X	X
Power Down entry	H	L	H	X	X	X	X	X	X
			L	H	H	H	X	X	X
Power Down exit	L	H	H	X	X	X	X	X	X
			L	H	H	H	X	X	X

### 12.2.3.1 Refresh Mode

The DDR2/mDDR memory controller issues refresh commands to the DDR2/mDDR SDRAM memory (Figure 12-4). REFR is automatically preceded by a DCAB command, ensuring the deactivation of all CE spaces and banks selected. Following the DCAB command, the DDR2/mDDR memory controller begins performing refreshes at a rate defined by the refresh rate (RR) bit in the SDRAM refresh control register (SDRCR). Page information is always invalid before and after a REFR command; thus, a refresh cycle always forces a page miss. This type of refresh cycle is often called autorefresh. Autorefresh commands may not be disabled within the DDR2/mDDR memory controller. See Section 12.2.7 for more details on REFR command scheduling.

Figure 12-4. Refresh Command

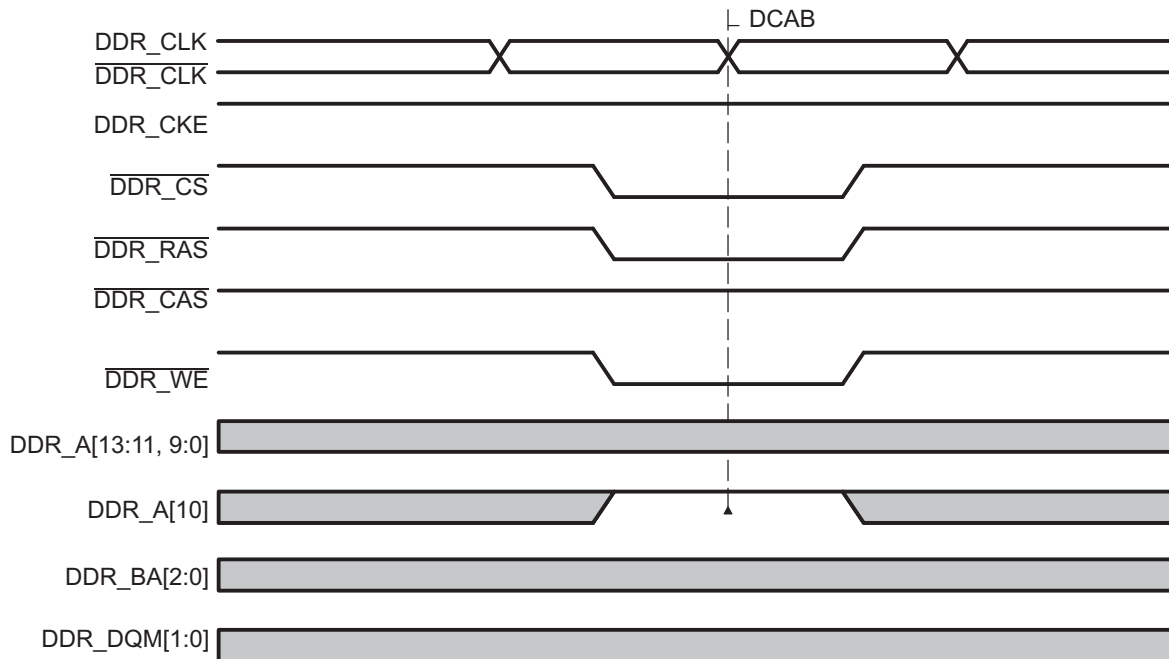




### 12.2.3.2 Deactivation (DCAB and DEAC)

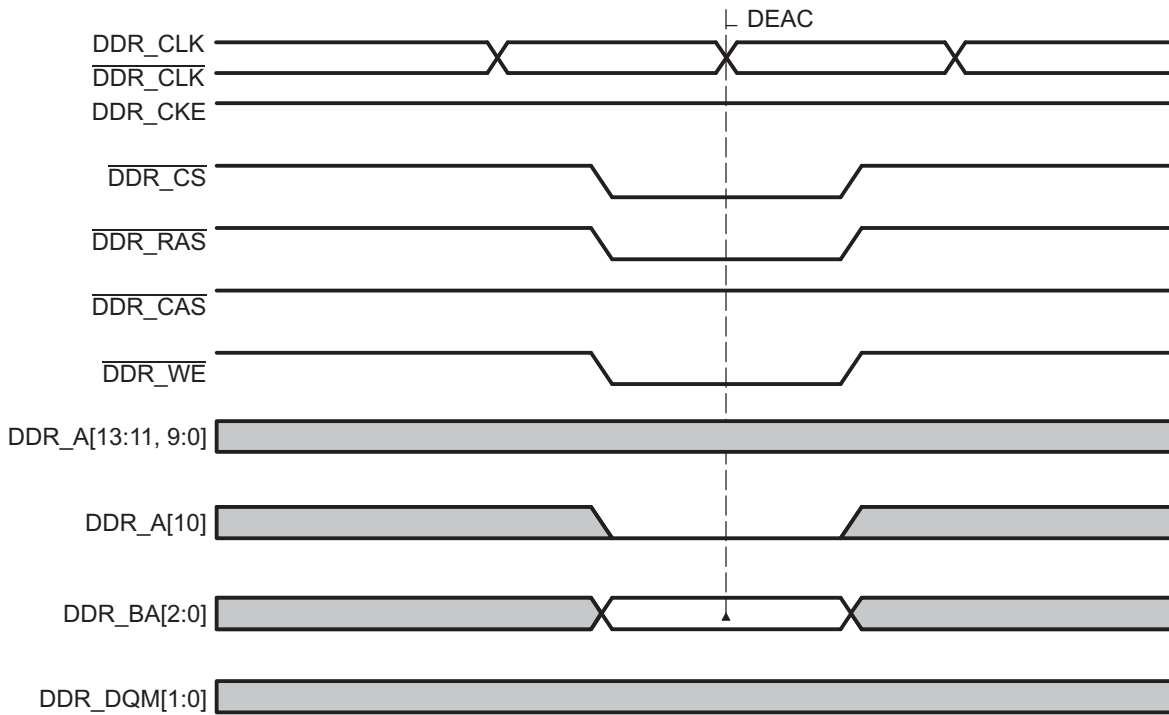
The precharge all banks command (DCAB) is performed after a reset to the DDR2/mDDR memory controller or following the initialization sequence. DDR2/mDDR SDRAMs also require this cycle prior to a refresh (REFR) and mode set register commands (MRS and EMRS). During a DCAB command, DDR\_A[10] is driven high to ensure the deactivation of all banks. Figure 12-5 shows the timing diagram for a DCAB command.

**Figure 12-5. DCAB Command**



The DEAC command closes a single bank of memory specified by the bank select signals. [Figure 12-6](#) shows the timings diagram for a DEAC command.

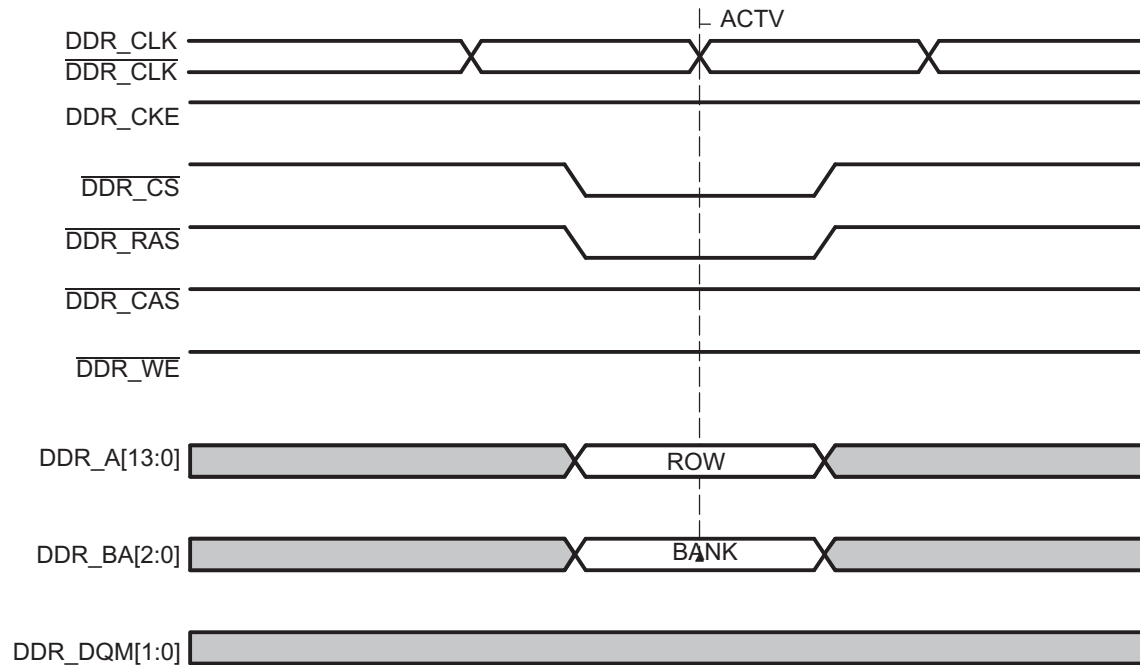
**Figure 12-6. DEAC Command**



### 12.2.3.3 Activation (ACTV)

The DDR2/mDDR memory controller automatically issues the activate (ACTV) command before a read or write to a closed row of memory. The ACTV command opens a row of memory, allowing future accesses (reads or writes) with minimum latency. The value of DDR\_BA[2:0] selects the bank and the value of DDR\_A[13:0] selects the row. When the DDR2/mDDR memory controller issues an ACTV command, a delay of  $t_{RCD}$  is incurred before a read or write command is issued. Figure 12-7 shows an example of an ACTV command. Reads or writes to the currently active row and bank of memory can achieve much higher throughput than reads or writes to random areas because every time a new row is accessed, the ACTV command must be issued and a delay of  $t_{RCD}$  incurred.

**Figure 12-7. ACTV Command**

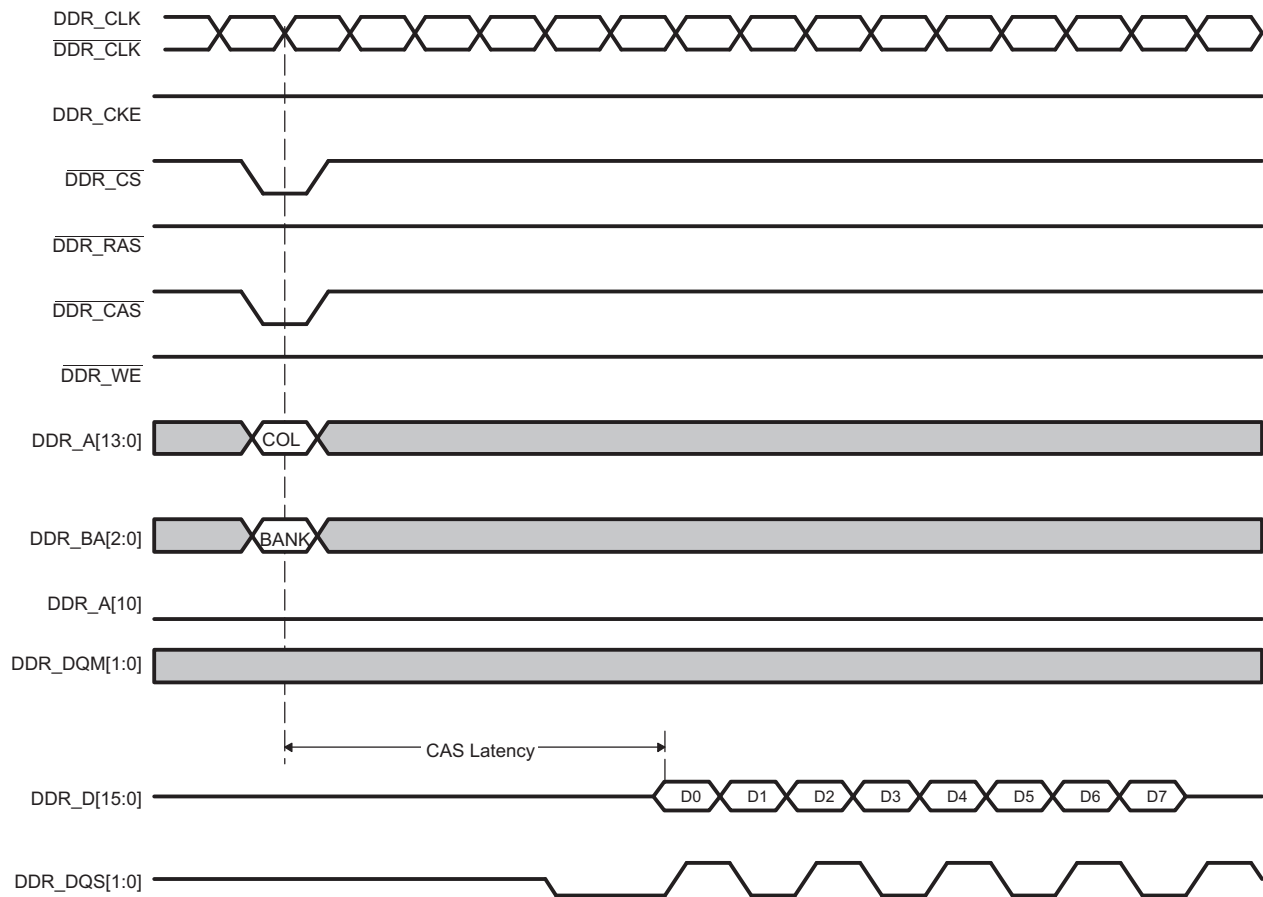


### 12.2.3.4 READ Command

Figure 12-8 shows the DDR2/mDDR memory controller performing a read burst from DDR2/mDDR SDRAM. The READ command initiates a burst read operation to an active row. During the READ command,  $\overline{\text{DDR\_CAS}}$  drives low,  $\overline{\text{DDR\_WE}}$  and  $\overline{\text{DDR\_RAS}}$  remain high, the column address is driven on  $\text{DDR\_A}[13:0]$ , and the bank address is driven on  $\text{DDR\_BA}[2:0]$ .

The DDR2/mDDR memory controller uses a burst length of 8, and has a programmable CAS latency of 2, 3, 4, or 5. The CAS latency is three cycles in Figure 12-8. Read latency is equal to CAS latency plus additive latency. The DDR2/mDDR memory controller always configures the memory to have an additive latency of 0, so read latency equals CAS latency. Since the default burst size is 8, the DDR2/mDDR memory controller returns 8 pieces of data for every read command. If additional accesses are not pending to the DDR2/mDDR memory controller, the read burst completes and the unneeded data is disregarded. If additional accesses are pending, depending on the scheduling result, the DDR2/mDDR memory controller can terminate the read burst and start a new read burst. Furthermore, the DDR2/mDDR memory controller does not issue a DAB/DEAC command until page information becomes invalid.

Figure 12-8. DDR2/mDDR READ Command



### 12.2.3.5 Write (WRT) Command

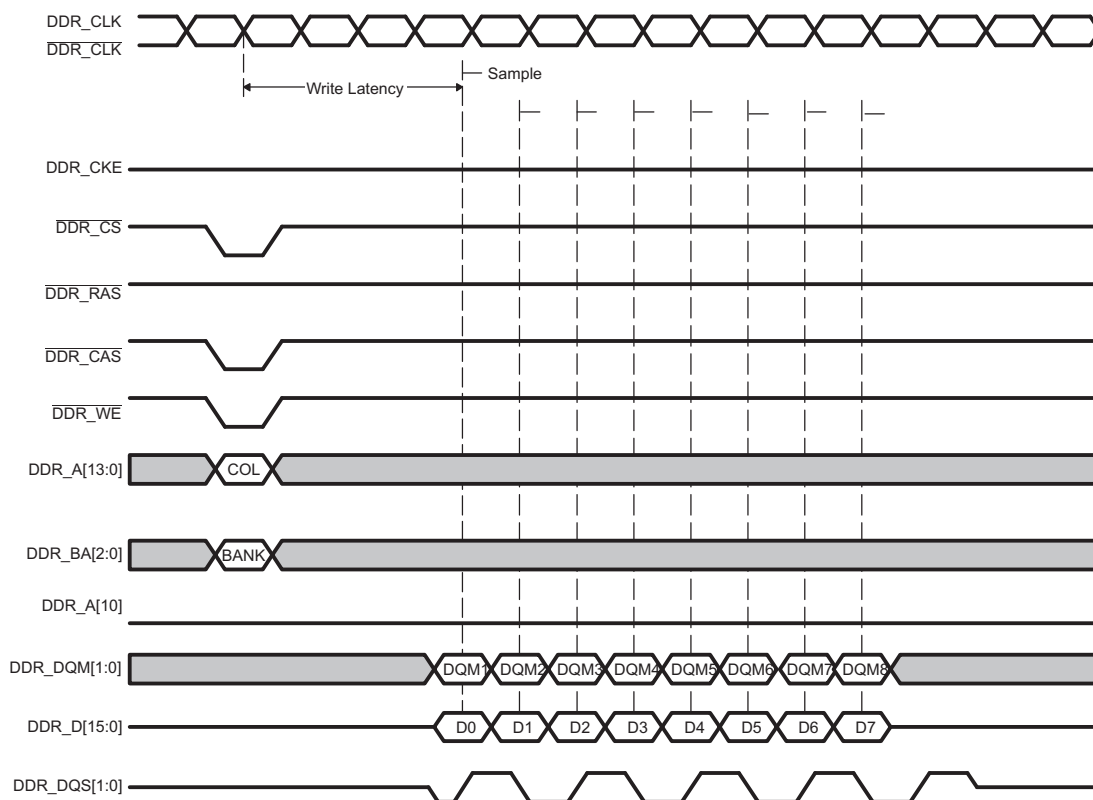
Prior to a WRT command, the desired bank and row are activated by the ACTV command. Following the WRT command, a write latency is incurred. For DDR2, write latency is equal to CAS latency minus 1 cycles. For mDDR, write latency is equal to 1 cycle, always. All writes have a burst length of 8. The use of the DDR\_DQM outputs allows byte and halfword writes to be executed. Figure 12-9 shows the timing for a DDR2 write on the DDR2/mDDR memory controller.

If the transfer request is for less than 8 words, depending on the scheduling result and the pending commands, the DDR2/mDDR memory controller can:

- Mask out the additional data using DDR\_DQM outputs
- Terminate the write burst and start a new write burst

The DDR2/mDDR memory controller does not perform the DEAC command until page information becomes invalid.

**Figure 12-9. DDR2/mDDR WRT Command**



NOTE: This diagrams shows write latency for DDR2. For mDDR, write latency is always equal to 1 cycle.

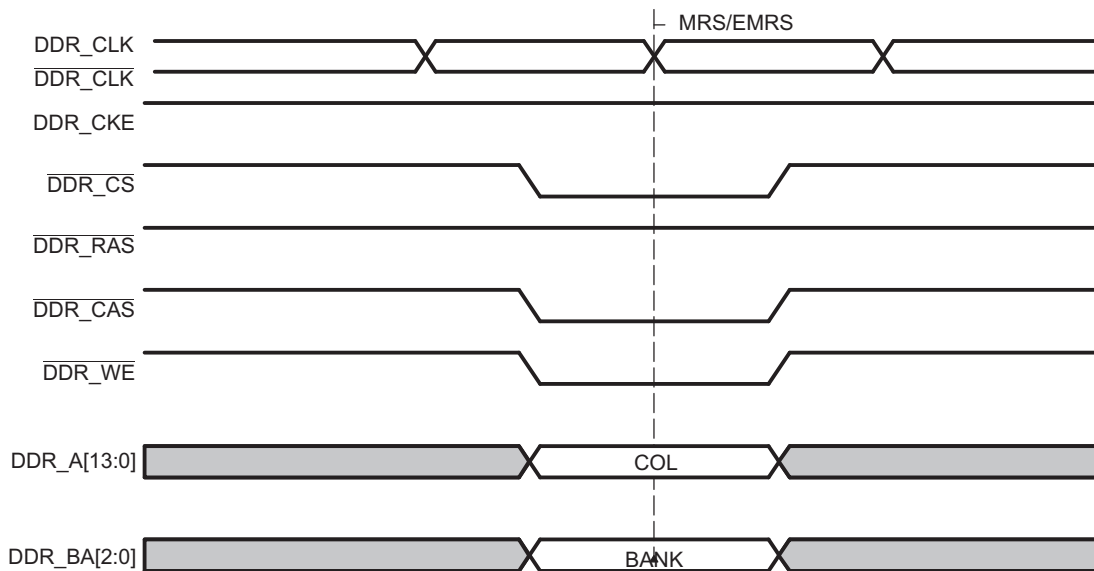
### 12.2.3.6 Mode Register Set (MRS and EMRS)

DDR2/mDDR SDRAM contains mode and extended mode registers that configure the DDR2/mDDR memory for operation. These registers control burst type, burst length, CAS latency, DLL enable/disable (on DDR2/mDDR device), single-ended strobe, differential strobe etc.

The DDR2/mDDR memory controller programs the mode and extended mode registers of the DDR2/mDDR memory by issuing MRS and EMRS commands. When the MRS or EMRS command is executed, the value on DDR\_BA[2:0] selects the mode register to be written and the data on DDR\_A[13:0] is loaded into the register. Figure 12-10 shows the timing for an MRS and EMRS command.

The DDR2/mDDR memory controller only issues MRS and EMRS commands during the DDR2/mDDR memory controller initialization sequence. See Section 12.2.13 for more information.

Figure 12-10. DDR2/mDDR MRS and EMRS Command



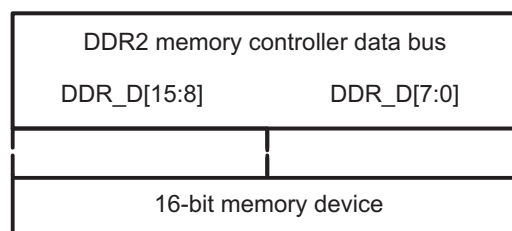
### 12.2.4 Memory Width and Byte Alignment

The DDR2/mDDR memory controller supports memory widths of 16 bits. Table 12-3 summarizes the addressable memory ranges on the DDR2/mDDR memory controller. Only little-endian format is supported. Figure 12-11 shows the byte lanes used on the DDR2/mDDR memory controller. The external memory is always right aligned on the data bus.

Table 12-3. Addressable Memory Ranges

Memory Width	Maximum addressable bytes per CS space	Description
x16	256 Mbytes	Halfword address

Figure 12-11. Byte Alignment



### 12.2.5 Address Mapping

The memory controller views the DDR2/mDDR SDRAM device as one continuous block of memory. The memory controller receives memory access requests with a 32-bit logical address, and it uses the logical address to generate a row, column, and bank address for accessing the DDR2/mDDR SDRAM device.

The memory controller supports two address mapping schemes: normal address mapping and special address mapping. Special address mapping is typically used only with mDDR devices using partial array self-refresh.

When the internal bank position (IBANKPOS) bit in the SDRAM configuration register (SDCR) is cleared, the memory controller operates with normal address mapping. In this case, the number of column and bank address bits is determined by the IBANK and PAGESIZE fields in SDCR. The number of row address bits is determined by the number of valid address pins for the device and does not need to be set in a register.

When IBANKPOS is set to 1, the memory controller operates with special address mapping. In this case, the number of column, row, and bank address bits is determined by the PAGESIZE, ROWSIZE, and IBANK fields. The ROWSIZE field is in the SDRAM configuration register 2 (SDCR2). See [Table 12-4](#) for a descriptions of these bit fields.

**Table 12-4. Configuration Register Fields for Address Mapping**

Bit Field	Bit Value	Bit Description
IBANK		Defines the number of internal banks in the external DDR2/mDDR memory.
	0	1 bank
	1h	2 banks
	2h	4 banks
	3h	8 banks
PAGESIZE		Defines the page size of each page in the external DDR2/mDDR memory.
	0	256 words (requires 8 column address bits)
	1h	512 words (requires 9 column address bits)
	2h	1024 words (requires 10 column address bits)
	3h	2048 words (requires 11 column address bits)
ROWSIZE		Defines the row size of each row in the external DDR2/mDDR memory
	0	512 (requires 9 row address bits)
	1h	1024 (requires 10 row address bits)
	2h	2048 (requires 11 row address bits)
	3h	4096 (requires 12 row address bits)
	4h	8192 (requires 13 row address bits)
	5h	16384 (requires 14 row address bits)

### 12.2.5.1 Normal Address Mapping (IBANKPOS = 0)

As stated in [Table 12-4](#), the IBANK and PAGESIZE fields of SDCR control the mapping of the logical, source address of the DDR2/mDDR memory controller to the DDR2/mDDR SDRAM row, column, and bank address bits. The DDR2/mDDR memory controller logical address always contains up to 14 row address bits, whereas the number of column and bank bits are determined by the IBANK and PAGESIZE fields. [Table 12-5](#) show how the logical address bits map to the DDR2/mDDR SDRAM row, column, and bank bits for combinations of IBANK and PAGESIZE values. The same DDR2/mDDR memory controller pins provide the row and column address to the DDR2/mDDR SDRAM, thus the DDR2/mDDR memory controller appropriately shifts the address during row and column address selection.

[Logical Address-to-DDR2/mDDR SDRAM Address Map](#) shows how this address-mapping scheme organizes the DDR2/mDDR SDRAM rows, columns, and banks into the device memory-map. Note that during a linear access, the DDR2/mDDR memory controller increments the column address as the logical address increments. When the DDR2/mDDR memory controller reaches a page/row boundary, it moves onto the same page/row in the next bank. This movement continues until the same page has been accessed in all banks. To the DDR2/mDDR SDRAM, this process looks as shown in [Figure 12-12](#).

By traversing across banks while remaining on the same row/page, the DDR2/mDDR memory controller maximizes the number of activated banks for a linear access. This results in the maximum number of open pages when performing a linear access being equal to the number of banks. Note that the DDR2/mDDR memory controller never opens more than one page per bank.

Ending the current access is not a condition that forces the active DDR2/mDDR SDRAM row to be closed. The DDR2/mDDR memory controller leaves the active row open until it becomes necessary to close it. This decreases the deactivate-reactivate overhead.

**Table 12-5. Logical Address-to-DDR2/mDDR SDRAM Address Map for 16-bit SDRAM**

SDCR Bit		Logical Address																			
IBANK	PAGESIZE	31	30	29	28	27	26	25	24	23	22	21:15	14	13	12	11	10	9	8:1	0	
0	0	-										nrb=14						ncb=8			
1	0	-										nrb=14						nbb=1		ncb=8	
2h	0	-										nrb=14						nbb=2		ncb=8	
3h	0	-										nrb=14						nbb=3		ncb=8	
0	1	-										nrb=14						ncb=9			
1	1	-										nrb=14						nbb=1		ncb=9	
2h	1	-										nrb=14						nbb=2		ncb=9	
3h	1	-										nrb=14						nbb=3		ncb=9	
0	2h	-										nrb=14						ncb=10			
1	2h	-										nrb=14						nbb=1		ncb=10	
2h	2h	-										nrb=14						nbb=2		ncb=10	
3h	2h	-										nrb=14						nbb=3		ncb=10	
0	3h	-										nrb=14						ncb=11			
1	3h	-										nrb=14						nbb=1		ncb=11	
2h	3h	-										nrb=14						nbb=2		ncb=11	
3h	3h	-										nrb=14						nbb=3		ncb=11	



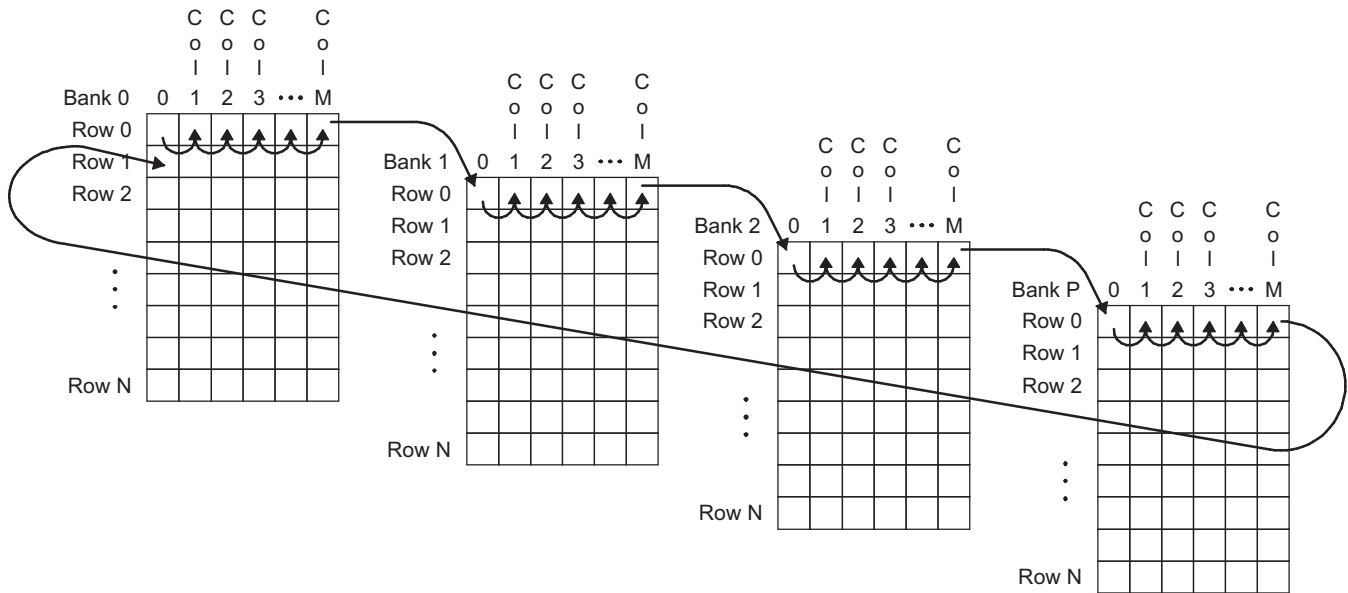
**Logical Address-to-DDR2/mDDR SDRAM Address Map**

Col. 0	Col. 1	Col. 2	Col. 3	Col. 4	...	Col. M-1	Col. M	
					...			Row 0, bank 0
					...			Row 0, bank 1
					...			Row 0, bank 2
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
					...			Row 0, bank P
					...			Row 1, bank 0
					...			Row 1, bank 1
					...			Row 1, bank 2
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
					...			Row 1, bank P
					...			•
					...			•
					...			•
					...			Row N, bank 0
					...			Row N, bank 1
					...			Row N, bank 2
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
					...			Row N, bank P

NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by both PAGESIZE and IBANK) minus 1.

Logical Address-to-DDR2/mDDR SDRAM Address Map (continued)

Figure 12-12. DDR2/mDDR SDRAM Column, Row, and Bank Access



NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by both PAGESIZE and IBANK) minus 1.

12.2.5.2 Special Address Mapping (IBANKPOS = 1)

When the internal bank position (IBANKPOS) bit is set to 1, the PAGESIZE, ROWSIZE, and IBANK fields control the mapping of the logical source address of the memory controller to the column, row, and bank address bits of the SDRAM device. Table 12-6 shows which source address bits map to the SDRAM column, row, and bank address bits for all combinations of PAGESIZE, ROWSIZE, and IBANK.

When IBANKPOS is set to 1, the effect of the address-mapping scheme is that as the source address increments across an SDRAM page boundary, the memory controller proceeds to the next page in the same bank. This movement along the same bank continues until all the pages have been accessed in the same bank. The memory controller then proceeds to the next bank in the device. This sequence is shown in Figure 12-13 and Figure 12-14.

Since, in this address mapping scheme, the memory controller can keep only one bank open, this scheme is lower in performance than the case when IBANKPOS is cleared to 0. Therefore, this case is only recommended to be used with Partial Array Self-refresh for mDDR SDRAM where performance may be traded-off for power savings.

Table 12-6. Address Mapping Diagram for 16-Bit SDRAM (IBANKPOS = 1)

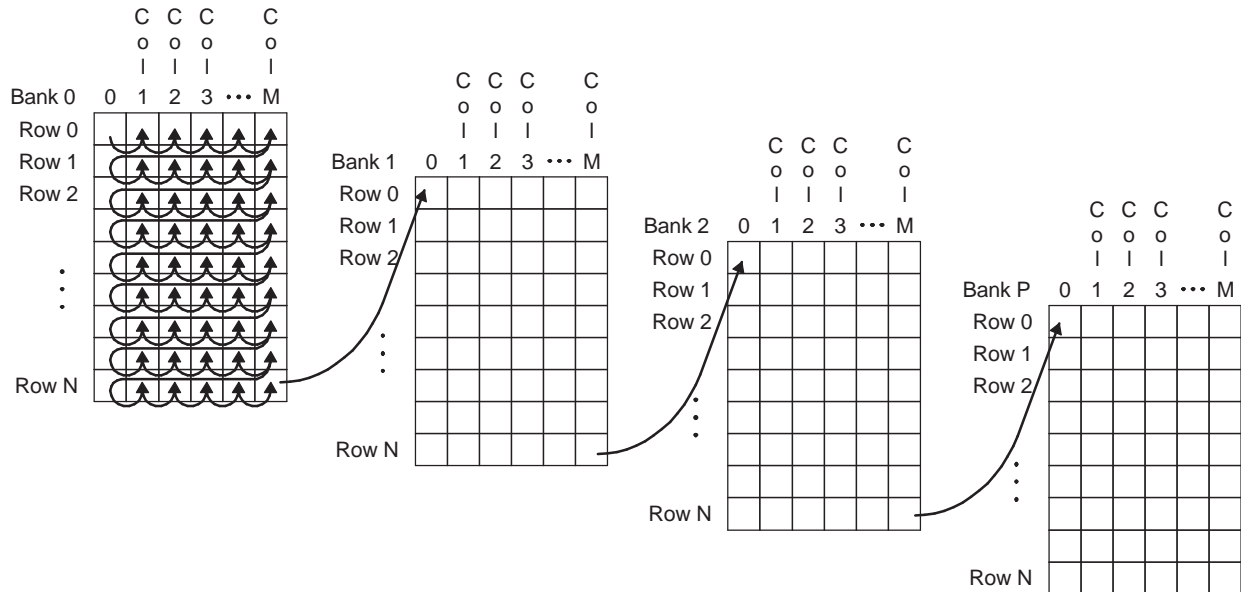
31	Source Address		1
<b>Bank Address</b>	<b>Row Address</b>	<b>Column Address</b>	
Number of bank bits is defined by IBANK nbb = 1, 2, or 3	Number of row bits is defined by ROWSIZE: nrb = 9, 10, 11, 12, 13, or 14	Number of column bits is defined by PAGESIZE: ncb = 8, 9, 10, or 11	

**Figure 12-13. Address Mapping Diagram (IBANKPOS = 1)**

Col. 0	Col. 1	Col. 2	Col. 3	Col. 4	...	Col. M-1	Col. M	
					...			Row 1, bank 0
					...			Row 2, bank 0
					...			Row 3, bank 0
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
					...			Row N, bank 0
					...			Row 1, bank 1
					...			Row 2, bank 1
					...			Row 3, bank 1
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
					...			Row N, bank 1
					...			•
					...			•
					...			•
					...			Row 1, bank P
					...			Row 2, bank P
					...			Row 3, bank P
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
•	•	•	•	•	...	•	•	•
					...			Row N, bank P

NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by ROWSIZE) minus 1.

**Figure 12-14. SDRAM Column, Row, Bank Access (IBANKPOS = 1)**



NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by ROWSIZE) minus 1.

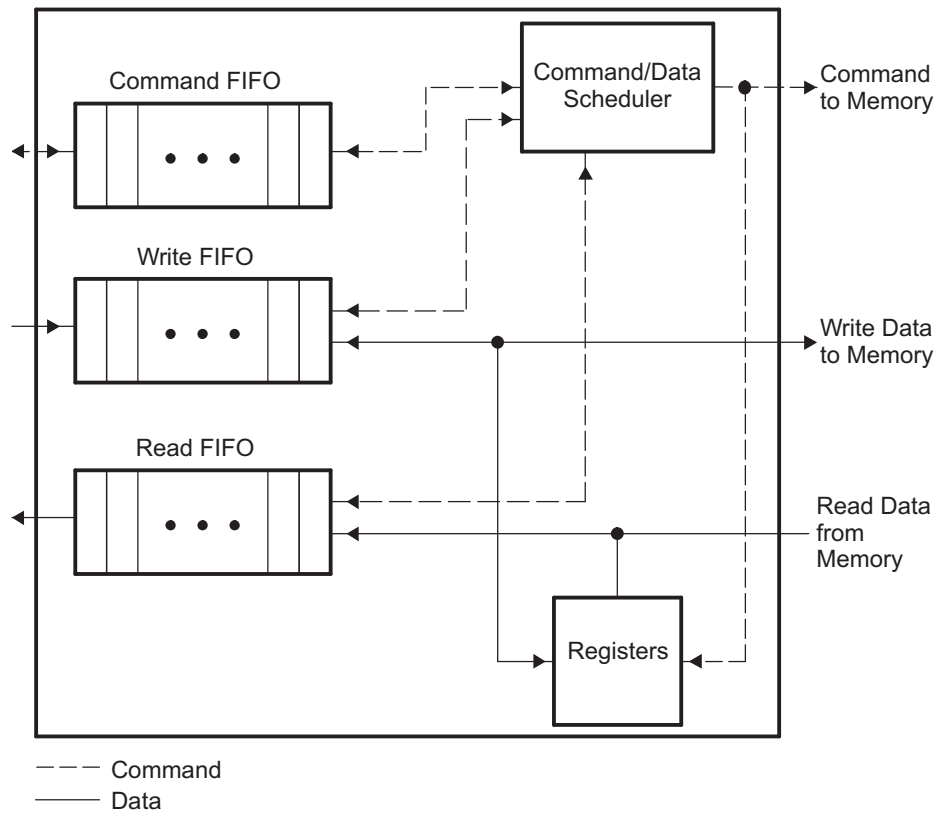
### 12.2.6 DDR2/mDDR Memory Controller Interface

To move data efficiently from on-chip resources to external DDR2/mDDR SDRAM memory, the DDR2/mDDR memory controller makes use of a command FIFO, a write FIFO, a read FIFO, and command and data schedulers. [Table 12-7](#) describes the purpose of each FIFO.

[Figure 12-15](#) shows the block diagram of the DDR2/mDDR memory controller FIFOs. Commands, write data, and read data arrive at the DDR2/mDDR memory controller parallel to each other. The same peripheral bus is used to write and read data from external memory as well as internal memory-mapped registers.

**Table 12-7. DDR2/mDDR Memory Controller FIFO Description**

FIFO	Description	Depth (64-bit doublewords)
Command	Stores all commands coming from on-chip requestors	7
Write	Stores write data coming from on-chip requestors to memory	11
Read	Stores read data coming from memory to on-chip requestors	17

**Figure 12-15. DDR2/mDDR Memory Controller FIFO Block Diagram**


### 12.2.6.1 Command Ordering and Scheduling, Advanced Concept

The DDR2/mDDR memory controller performs command re-ordering and scheduling in an attempt to achieve efficient transfers with maximum throughput. The goal is to maximize the utilization of the data, address, and command buses while hiding the overhead of opening and closing DDR2/mDDR SDRAM rows. Command re-ordering takes place within the command FIFO.

Typically, a given master issues commands on a single priority. EDMA transfer controller read and write ports are different masters. The DDR2/mDDR memory controller first reorders commands from each master based on the following rules:

- Selects the oldest command (first command in the queue)
- Selects a read before a write if:
  - The read is to a different block address (2048 bytes) than the write
  - The read has greater or equal priority

The second bullet above may be viewed as an exception to the first bullet. This means that for an individual master, all of its commands will complete from oldest to newest, with the exception that a read may be advanced ahead of an older, lower or equal priority write. Following this scheduling, each master may have one command ready for execution.

Next, the DDR2/mDDR memory controller examines each of the commands selected by the individual masters and performs the following reordering:

- Among all pending reads, selects reads to rows already open. Among all pending writes, selects writes to rows already open.
- Selects the highest priority command from pending reads and writes to open rows. If multiple commands have the highest priority, then the DDR2/mDDR memory controller selects the oldest command.

The DDR2/mDDR memory controller may now have a final read and write command. If the Read FIFO is not full, then the read command will be performed before the write command, otherwise the write command will be performed first.

Besides commands received from on-chip resources, the DDR2/mDDR memory controller also issues refresh commands. The DDR2/mDDR memory controller attempts to delay refresh commands as long as possible to maximize performance while meeting the SDRAM refresh requirements. As the DDR2/mDDR memory controller issues read, write, and refresh commands to DDR2/mDDR SDRAM memory, it adheres to the following rules:

1. Refresh request resulting from the Refresh Must level of urgency being reached
2. Read request without a higher priority write (selected from above reordering algorithm)
3. Refresh request resulting from the Refresh Need level of urgency being reached
4. Write request (selected from above reordering algorithm)
5. Refresh request resulting from Refresh May level of urgency being reached
6. Request to enter self-refresh mode

The following results from the above scheduling algorithm:

- All writes from a single master will complete in order
- All reads from a single master will complete in order
- From the same master, any read to the same location (or within 2048 bytes) as a previous write will complete in order

#### 12.2.6.2 Command Starvation

The reordering and scheduling rules listed above may lead to command starvation, which is the prevention of certain commands from being processed by the DDR2/mDDR memory controller. Command starvation results from the following conditions:

- A continuous stream of high-priority read commands can block a low-priority write command
- A continuous stream of DDR2/mDDR SDRAM commands to a row in an open bank can block commands to the closed row in the same bank.

To avoid these conditions, the DDR2/mDDR memory controller can momentarily raise the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR\_OLD\_COUNT bit in the peripheral bus burst priority register (PBBPR) sets the number of the transfers that must be made before the DDR2/mDDR memory controller will raise the priority of the oldest command.

#### 12.2.6.3 Possible Race Condition

A race condition may exist when certain masters write data to the DDR2/mDDR memory controller. For example, if master A passes a software message via a buffer in DDR2/mDDR memory and does not wait for indication that the write completes, when master B attempts to read the software message it may read stale data and therefore receive an incorrect message. In order to confirm that a write from master A has landed before a read from master B is performed, master A must wait for the write completion status from the DDR2/mDDR memory controller before indicating to master B that the data is ready to be read. If master A does not wait for indication that a write is complete, it must perform the following workaround:

1. Perform the required write.
2. Perform a dummy write to the DDR2/mDDR memory controller SDRAM status register.
3. Perform a dummy read to the DDR2/mDDR memory controller SDRAM status register.
4. Indicate to master B that the data is ready to be read after completion of the read in step 3. The completion of the read in step 3 ensures that the previous write was done.

The EDMA peripheral does not need to implement the above workaround. The above workaround is required for all other peripherals. See your device-specific data manual for more information.

### 12.2.7 Refresh Scheduling

The DDR2/mDDR memory controller issues autorefresh (REFR) commands to DDR2/mDDR SDRAM devices at a rate defined in the refresh rate (RR) bit field in the SDRAM refresh control register (SDRCR). A refresh interval counter is loaded with the value of the RR bit field and decrements by 1 each cycle until it reaches zero. Once the interval counter reaches zero, it reloads with the value of the RR bit. Each time the interval counter expires, a refresh backlog counter increments by 1. Conversely, each time the DDR2/mDDR memory controller performs a REFR command, the backlog counter decrements by 1. This means the refresh backlog counter records the number of REFR commands the DDR2/mDDR memory controller currently has outstanding.

The DDR2/mDDR memory controller issues REFR commands based on the level of urgency. The level of urgency is defined in [Table 12-8](#). Whenever the refresh must level of urgency is reached, the DDR2/mDDR memory controller issues a REFR command before servicing any new memory access requests. Following a REFR command, the DDR2/mDDR memory controller waits  $T_{RFC}$  cycles, defined in the SDRAM timing register 1 (SDTIMR1), before rechecking the refresh urgency level.

In addition to the refresh counter previously mentioned, a separate backlog counter ensures the interval between two REFR commands does not exceed  $8 \times$  the refresh rate. This backlog counter increments by 1 each time the interval counter expires and resets to zero when the DDR2/mDDR memory controller issues a REFR command. When this backlog counter is greater than 7, the DDR2/mDDR memory controller issues four REFR commands before servicing any new memory requests.

The refresh counters do not operate when the DDR2/mDDR memory is in self-refresh mode.

**Table 12-8. Refresh Urgency Levels**

Urgency Level	Description
Refresh May	Backlog count is greater than 0. Indicates there is a backlog of REFR commands, when the DDR2/mDDR memory controller is not busy it will issue the REFR command.
Refresh Release	Backlog count is greater than 3. Indicates the level at which enough REFR commands have been performed and the DDR2/mDDR memory controller may service new memory access requests.
Refresh Need	Backlog count is greater than 7. Indicates the DDR2/mDDR memory controller should raise the priority level of a REFR command above servicing a new memory access.
Refresh Must	Backlog count is greater than 11. Indicates the level at which the DDR2/mDDR memory controller should perform a REFR command before servicing new memory access requests.

### 12.2.8 Self-Refresh Mode

Clearing the self refresh/low power (SR\_PD) bit to 0 and then setting the low power mode enable (LPMODEN) bit to 1 in the SDRAM refresh control register (SDRCR), forces the DDR2/mDDR memory controller to place the external DDR2/mDDR SDRAM in a low-power mode (self refresh), in which the DDR2/mDDR SDRAM maintains valid data while consuming a minimal amount of power. When the LPMODEN bit is set to 1, the DDR2/mDDR memory controller continues normal operation until all outstanding memory access requests have been serviced and the refresh backlog has been cleared. At this point, all open pages of DDR2/mDDR SDRAM are closed and a self-refresh (SLFRFR) command (an autorefresh command with self refresh/low power) is issued.

The memory controller exits the self-refresh state when a memory access is received, when the LPMODEN bit in SDRCR is cleared to 0, or when the SR\_PD bit in SDRCR changed to 1. While in the self-refresh state, if a request for a memory access is received, the DDR2/mDDR memory controller services the memory access request, returning to the self-refresh state upon completion. The DDR2/mDDR memory controller will not wake up from the self-refresh state (whether from a memory access request, from clearing the LPMODEN bit, or from clearing the SR\_PD bit) until  $T_{CKE} + 1$  cycles have expired since the self-refresh command was issued. The value of  $T_{CKE}$  is defined in the SDRAM timing register 2 (SDTIMR2).

In the case of DDR2, after exiting from the self-refresh state, the memory controller will not immediately start executing commands. Instead, it will wait  $T_{SXNR} + 1$  clock cycles before issuing non-read/write commands and  $T_{SXRd} + 1$  clock cycles before issuing read or write commands. The SDRAM timing register 2 (SDTIMR2) programs the values of  $T_{SXNR}$  and  $T_{SXRd}$ .

In the case of mDDR, after exiting from the self-refresh state, the memory controller will not immediately start executing commands. Instead, it will wait  $T_{SXNR}+1$  clock cycles and then execute auto-refresh command before issuing any other commands. The SDRAM timing register 2 (SDTIMR2) programs the value of  $T_{SXNR}$ .

Once in self-refresh mode, the DDR2/mDDR memory controller input clocks (VCLK and 2X\_CLK) may be gated off or changed in frequency. Stable clocks must be present before exiting self-refresh mode. See [Section 12.2.16](#) for more information describing the proper procedure to follow when shutting down DDR2/mDDR memory controller input clocks.

See [Section 12.2.16.1](#) for a description of the self-refresh programming sequence.

### 12.2.9 Partial Array Self Refresh for Mobile DDR

For additional power savings during self-refresh, the partial array self-refresh (PASR) feature of the mDDR allows you to select the amount of memory that will be refreshed during self-refresh. Use the partial array self-refresh (PASR) bit field in the SDRAM configuration register 2 (SDCR2) to select the amount of memory to refresh during self-refresh. As shown in [Table 12-9](#) you may select either 4, 2, 1, 1/2, or 1/4 bank(s). The PASR bits are loaded into the extended mode register of the mDDR device, during autoinitialization (see [Section 12.2.13](#)).

The mDDR performs bank interleaving when the internal bank position (IBANKPOS) bit in SDRAM configuration register (SDCR) is cleared to 0. Since the SDRAM banks are only partially refreshed during partial array self-refresh, it is recommended that you set IBANKPOS to 1 to avoid bank interleaving. When IBANKPOS is cleared to 0, it is the responsibility of software to move critical data into the banks that are to be refreshed during partial array self-refresh. Refer to [Section 12.2.5.2](#) for more information on IBANKPOS and addressing mapping in general.

**Table 12-9. Configuration Bit Field for Partial Array Self-refresh**

Bit Field	Bit Value	Bit Description
PASR		Partial array self refresh.
	0	Refresh banks 0, 1, 2, and 3
	1h	Refresh banks 0 and 1
	2h	Refresh bank 0
	5h	Refresh 1/2 of bank 0
	6h	Refresh 1/4 of bank 0

### 12.2.10 Power-Down Mode

Setting the self-refresh/low power (SR\_PD) bit and the low-power mode enable (LPMODEN) bit in the SDRAM refresh control register (SDRCR) to 1, forces the DDR2/mDDR memory controller to place the external DDR2 SDRAM in the power-down mode. When the LPMODEN bit is asserted, the DDR2/mDDR memory controller continues normal operation until all outstanding memory access requests have been serviced and the refresh backlog has been cleared. At this point, all open pages of DDR2 SDRAM are closed and a Power Down command (same as NOP command but driving DDR\_CKE low on the same cycle) is issued.

The DDR2/mDDR memory controller exits the power-down state when a memory access is received, when a Refresh Must level is reached, when the LPMODEN bit in SDRCR is cleared to 0, or when the SR\_PD bit in SDRCR changed to 0. While in the power-down state, if a request for a memory access is received, the DDR2/mDDR memory controller services the memory access request, returning to the power-down state upon completion. The DDR2/mDDR memory controller will not wake-up from the power-down state (whether from a memory access request, from reaching a Refresh Must level, from clearing the LPMODEN bit, or from clearing the SR\_PD bit) until  $T_{CKE} + 1$  cycles have expired since the power-down command was issued. The value of  $T_{CKE}$  is defined in the SDRAM timing register 2 (SDTIMR2).



After exiting from the power-down state, the DDR2/mDDR memory controller will drive DDR\_CKE high and then not immediately start executing commands. Instead, it will wait  $T_{XP} + 1$  clock cycles before issuing commands. The SDRAM timing register 2 (SDTIMR2) programs the values of  $T_{XP}$ .

See [Section 12.2.16.1](#) for a description of the power-down mode programming sequence.

---

**NOTE:** Power-down mode is best suited as a power savings mode when SDRAM is being used intermittently and the system requires power savings as well as a short recovery time. You may use self-refresh mode if you desire additional power savings from disabling clocks.

---

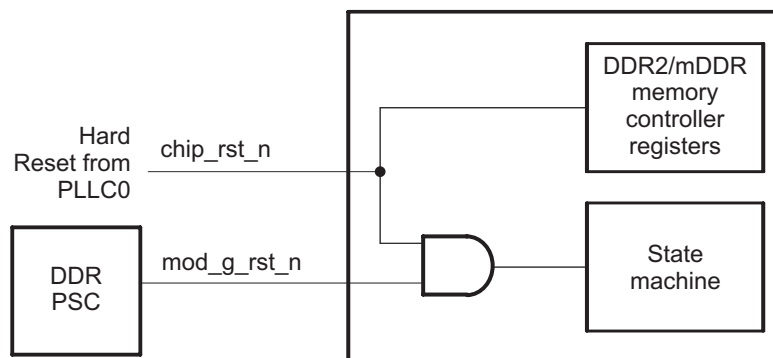
### 12.2.11 Reset Considerations

The DDR2/mDDR memory controller has two reset signals, `chip_rst_n` and `mod_g_rst_n`. The `chip_rst_n` is a module-level reset that resets both the state machine as well as the DDR2/mDDR memory controller memory-mapped registers. The `mod_g_rst_n` resets the state machine only; it does not reset the controller's registers, which allows soft reset (from PSC or WDT) to reset the module without resetting the configuration registers and reduces the programming overhead for setting up access to the DDR2/mDDR device. If the DDR2/mDDR memory controller is reset independently of other peripherals, the user's software should not perform memory, as well as register accesses, while `chip_rst_n` or `mod_g_rst_n` are asserted. If memory or register accesses are performed while the DDR2/mDDR memory controller is in the reset state, other masters may hang. Following the rising edge of `chip_rst_n` or `mod_g_rst_n`, the DDR2/mDDR memory controller immediately begins its initialization sequence. Command and data stored in the DDR2/mDDR memory controller FIFOs are lost. [Table 12-10](#) describes the different methods for asserting each reset signal. The Power and Sleep Controller (PSC) acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter. [Figure 12-16](#) shows the DDR2/mDDR memory controller reset diagram.

**Table 12-10. Reset Sources**

Reset Signal	Reset Source
<code>chip_rst_n</code>	Hardware/device reset
<code>mod_g_rst_n</code>	Power and sleep controller

**Figure 12-16. DDR2/mDDR Memory Controller Reset Block Diagram**



### 12.2.12 VTP IO Buffer Calibration

The DDR2/mDDR memory controller is able to control the impedance of the output IO. This feature allows the DDR2/mDDR memory controller to tune the output impedance of the IO to match that of the PCB board. Control of the output impedance of the IO is an important feature because impedance matching reduces reflections, creating a cleaner board design. Calibrating the output impedance of the IO will also reduce the power consumption of the DDR2/mDDR memory controller. The calibration is performed with respect to voltage, temperature, and process (VTP). The VTP information obtained from the calibration is used to control the output impedance of the IO.

The impedance of the output IO is selected by the value of a reference resistor connected to pin DDR\_ZP. The DDR2/mDDR reference design requires the reference resistor to be a 50 ohm, 5.0% tolerance, 1/16th watt resistor (49.9 ohm, 0.5% tolerance is acceptable).

The VTP IO control register (VTPIO\_CTL) is written to begin the calibration process. The VTP calibration process is described in the DDR2/mDDR initialization sequence in [Section 12.2.13.1](#).

---

**NOTE:** VTP IO calibration must be performed following device power up and device reset. If the DDR2/mDDR memory controller is reset via the Power and Sleep Controller (PSC) and the VTP input clock is disabled, accesses to the DDR2/mDDR memory controller will not complete. To re-enable accesses to the DDR2/mDDR memory controller, enable the VTP input clock and then perform the VTP calibration sequence again.

---

### 12.2.13 Auto-Initialization Sequence

The DDR2/mDDR SDRAM contains mode and extended mode registers that configure the DDR2/mDDR memory for operation. These registers control burst type, burst length, CAS latency, DLL enable/disable (on the DDR2/mDDR device), single-ended strobe, differential strobe, etc. The DDR2/mDDR memory controller programs the mode and extended mode registers of the DDR2/mDDR memory by issuing MRS and EMRS commands during the initialization sequence. The SDRAMEN, MSDRAMEN, DDREN, and DDR2EN bits in the SDRAM configuration register (SDCR) determine if the DDR2/mDDR memory controller will perform a DDR2 or mobile DDR initialization sequence. Set these bits as follows for DDR2: SDRAMEN = 1, MSDRAMEN = 0, DDREN = 1, DDR2EN = 1. Set these bits as follow for mDDR: SDRAMEN = 1, MSDRAMEN = 1, DDREN = 1, DDR2EN = 0. The DDR2 initialization sequence performed by the DDR2/mDDR memory controller is compliant with the JESD79D-2 specification and the mDDR initialization sequence is compliant with the JESD209 specification. The DDR2/mDDR memory controller performs an initialization sequence under the following conditions:

- Following reset (rising edge of chip\_rst\_n or mod\_g\_rst\_n)
- Following a write to the DDRDRIVE, CL, IBANK, or PAGESIZE bit fields in the SDRAM configuration register (SDCR)

During the initialization sequence, the memory controller issues MRS and EMRS commands that configure the DDR2/mobile DDR SDRAM mode register and extended mode register 1. The register values for DDR2 are described in [Table 12-11](#) and [Table 12-12](#), and the register values for mDDR are described in [Table 12-13](#) and [Table 12-14](#). The extended mode registers 2 and 3 are configured with a value of 0h. At the end of the initialization sequence, the memory controller performs an autorefresh cycle, leaving the memory controller in an idle state with all banks deactivated.

When a reset occurs, the DDR2/mDDR memory controller immediately begins the initialization sequence. Under this condition, commands and data stored in the DDR2/mDDR memory controller FIFOs will be lost. However, when the initialization sequence is initiated by a write to the two least-significant bytes in SDCR, data and commands stored in the DDR2/mDDR memory controller FIFOs will not be lost and the DDR2/mDDR memory controller will ensure read and write commands are completed before starting the initialization sequence.

**Table 12-11. DDR2 SDRAM Configuration by MRS Command**

Memory Controller Address Bus	Value	DDR2/mDDR SDRAM Register Bit	DDR2/mDDR SDRAM Field	Function Selection
DDR_A[12]	0	12	Power Down Exit	Fast exit
DDR_A[11:9]	t_WR	11:9	Write Recovery	Write recovery from autoprerecharge. Value of 2, 3, 4, 5, or 6 is programmed based on value of the T_WR bit in the SDRAM timing register 1 (SDTIMR1).
DDR_A[8]	0	8	DLL Reset	Out of reset
DDR_A[7]	0	7	Mode: Test or Normal	Normal mode
DDR_A[6:4]	CL bit	6:4	CAS Latency	Value of 2, 3, 4, or 5 is programmed based on value of the CL bit in the SDRAM configuration register (SDCR).
DDR_A[3]	0	3	Burst Type	Sequential
DDR_A[2:0]	3h	2:0	Burst Length	Value of 8

**Table 12-12. DDR2 SDRAM Configuration by EMRS(1) Command**

Memory Controller Address Bus	Value	DDR2/mDDR SDRAM Register Bit	DDR2/mDDR SDRAM Field	Function Selection
DDR_A[12]	0	12	Output Buffer Enable	Output buffer enable
DDR_A[11]	0	11	RDQS Enable	RDQS disable
DDR_A[10]	1	10	$\overline{\text{DQS enable}}$	Disables differential DQS signaling.
DDR_A[9:7]	0	9:7	OCD Calibration Program	Exit OCD calibration
DDR_A[6]	0	6	ODT Value (Rtt)	Cleared to 0 to select 75 ohms. This feature is not supported because the DDR_ODT signal is not pinned out.
DDR_A[5:3]	0	5:3	Additive Latency	0 cycles of additive latency
DDR_A[2]	1	2	ODT Value (Rtt)	Set to 1 to select 75 ohms. This feature is not supported because the DDR_ODT signal is not pinned out.
DDR_A[1]	DDRDRIVE[0]	1	Output Driver Impedance	Value of 0 or 1 is programmed based on value of DDRDRIVE0 bit in SDRAM configuration register (SDCR).
DDR_A[0]	0	0	DLL enable	DLL enable

**Table 12-13. Mobile DDR SDRAM Configuration by MRS Command**

Memory Controller Address Bus	Value	mDDR SDRAM Register Bit	mDDR SDRAM Field	Function Selection
DDR_A[11:7]	0	11:7	Operating mode	Normal operating mode
DDR_A[6:4]	CL bit	6:4	CAS Latency	Value of 2 or 3 is programmed based on value of CL bit in SDRAM configuration register (SDCR).
DDR_A[3]	0	3	Burst Type	Sequential
DDR_A[2:0]	3h	2:0	Burst Length	Value of 8

**Table 12-14. Mobile DDR SDRAM Configuration by EMRS(1) Command**

Memory Controller Address Bus	Value	mDDR SDRAM Register Bit	mDDR SDRAM Field	Function Selection
DDR_A[11:7]	0	11:7	Operating Mode	Normal operating mode
DDR_A[6:5]	DDRDRIVE[1:0]	6:5	Output Driver Impedance	Value of 0, 1, 2, or 3 is programmed based on value of DDRDRIVE[1:0] bits in SDRAM configuration register (SDCR).
DDR_A[4:3]	0	4:3	Temperature Compensated Self Refresh	Value of 0
DDR_A[2:0]	PASR bits	2:0	Partial Array Self Refresh	Value of 0, 1, 2, 5, or 6 is programmed based on value of PASR bits in SDRAM configuration register 2 (SDCR2).

### 12.2.13.1 Initializing Following Device Power Up or Reset

Following device power up or reset, the DDR2/mDDR memory controller is held in reset with the internal clocks to the module gated off. Before releasing the DDR2/mDDR memory controller from reset, the clocks to the module must be turned on. Perform the following steps when turning the clocks on and initializing the module:

1. Program PLLC1 registers to start the PLL1\_SYSCLK1 (that drives 2X\_CLK). For information on programming PLLC1, see the *Phase-Locked Loop Controller (PLLC)* chapter.
2. Program Power and Sleep Controller (PSC) to enable the DDR2/mDDR memory controller clock.
3. Perform VTP IO calibration:
  - (a) Clear POWERDN bit in the VTP IO control register (VTPIO\_CTL).
  - (b) Clear LOCK bit in VTPIO\_CTL.
  - (c) Pulse CLKRZ bit in VTPIO\_CTL:
    - (i) Set CLKRZ bit and wait at least 1 VTP clock cycle (clock cycle wait can be achieved by performing a read-modify-write of VTPIO\_CTL in the next step).
    - (ii) Clear CLKRZ bit and wait at least 1 VTP clock cycle (clock cycle wait can be achieved by performing a read-modify-write of VTPIO\_CTL in the next step).
    - (iii) Set CLKRZ bit.
  - (d) Poll READY bit in VTPIO\_CTL until it changes to 1.
  - (e) Set LOCK bit in VTPIO\_CTL. VTP is locked and dynamic calibration is disabled.
  - (f) Set POWERDN bit in VTPIO\_CTL to save power.
4. Set IOPWRDN bit in VTPIO\_CTL to allow the input receivers to save power when the PWRDNEN bit in the DDR PHY control register 1 (DRPYC1R) is set.
5. Configure DRPYC1R. All of the following steps may be done with a single register write to DRPYC1R:
  - (a) Set EXT\_STRBEN bit to select external DQS strobe gating.
  - (b) Set PWRDNEN bit to allow the input receivers to power down when they are idle.
  - (c) Program RL bit value to meet the memory data sheet specification.
6. Configure the DDR slew register (DDR\_SLEW):
  - (a) For DDR2, clear DDR\_PDENA and CMOSEN bits.
  - (b) For mDDR, set the DDR\_PDENA and CMOSEN bits.
7. Set the BOOTUNLOCK bit (unlocked) in the SDRAM configuration register (SDCR).
8. Program SDCR to the desired value with BOOTUNLOCK bit cleared to 0 and TIMUNLOCK bit set to 1 (unlocked).
9. For mDDR only, program the SDRAM configuration register 2 (SDCR2) to the desired value.
10. Program the SDRAM timing register 1 (SDTIMR1) and SDRAM timing register 2 (SDTIMR2) to the desired values to meet the memory data sheet specification.
11. Clear TIMUNLOCK bit (locked) in SDCR.

12. Program the SDRAM refresh control register (SDRCR). All of the following steps may be done with a single register write to SDRCR:
  - (a) Set LPMODEN bit to enable self-refresh. This is necessary for the next two steps.
  - (b) Set MCLKSTOPEN bit to enable MCLK stopping. This is necessary for the next two steps.
  - (c) Clear SR\_PD bit to select self-refresh. This is necessary for the next two steps.
  - (d) Program RR refresh rate value to meet the memory data sheet specification.
13. Program the Power and Sleep Controller (PSC) to reset (SyncReset) the DDR2/mDDR memory controller.
14. Program the Power and Sleep Controller (PSC) to re-enable the DDR2/mDDR memory controller.
15. Clear LPMODEN and MCLKSTOPEN bits in SDRCR to disable self-refresh.
16. Configure the peripheral bus burst priority register (PBBPR) to a value lower than the default value of FFh. A lower value reduces the likelihood of prolonged command starvation for accesses made from different master/peripherals to mDDR/DDR2 memory. The optimal value should be determined based on system considerations; however, a value of 20h or 30h is sufficient for typical applications.

---

**NOTE:** Some memory data sheet timing values such as those programmed into the SDRAM timing register 1 (SDTIMR1) and SDRAM timing register 2 (SDTIMR2) may need to be relaxed in order to compensate for signal delays introduced by board layout.

---

#### 12.2.14 Interrupt Support

The DDR2/mDDR memory controller supports two addressing modes, linear incrementing and cache line wrap. Upon receipt of an access request for an unsupported addressing mode, the DDR2/mDDR memory controller generates an interrupt by setting the LT bit in the interrupt raw register (IRR). The DDR2/mDDR memory controller will then treat the request as a linear incrementing request.

This interrupt is called the line trap interrupt and is the only interrupt the DDR2/mDDR memory controller supports. It is an active-high interrupt and is enabled by the LTMSET bit in the interrupt mask set register (IMSR). This interrupt is mapped to the CPU and is multiplexed with RTCINT.

#### 12.2.15 DMA Event Support

The DDR2/mDDR memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests may be made directly by masters and by the DMA.

### 12.2.16 Power Management

Power dissipation from the DDR2/mDDR memory controller may be managed by the following methods:

- Self-refresh mode (see [Section 12.2.8](#))
- Power-down mode (see [Section 12.2.10](#))
- Disabling the DDR PHY to reduce power

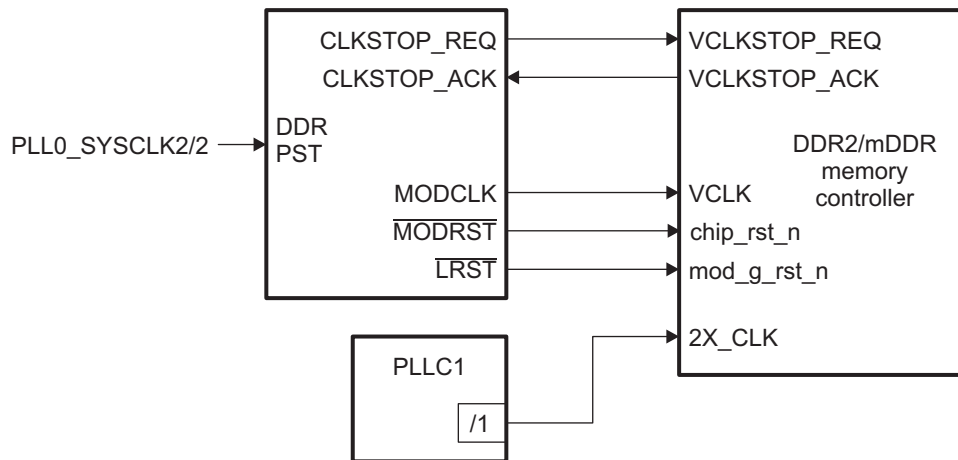
The DDR2/mDDR memory controller supports low-power modes where the DLL internal to the PHY and the receivers at the I/O pins can be disabled. These functions are controlled through the DDR2/mDDR memory controller. Even if the PHY is active, the receivers can be configured to disable whenever writes are in progress and the receivers are not needed.

- Gating input clocks to the module off

Gating input clocks off to the DDR2/mDDR memory controller achieves higher power savings when compared to the power savings of self-refresh mode and power-down mode. The input clocks are turned off outside of the DDR2/mDDR memory controller through the use of the Power and Sleep Controller (PSC) and the PLL controller 1 (PLL1). [Figure 12-17](#) shows the connections between the DDR2/mDDR memory controller, PSC, and PLL1. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

Before gating clocks off, the DDR2/mDDR memory controller must place the DDR2/mDDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the DDR2/mDDR memory controller clock provided by PLL1 must not be turned off because this may result in data corruption. See the following subsections for the proper procedures to follow when stopping the DDR2/mDDR memory controller clocks. Once the clocks are stopped, to re-enable the clocks follow the clock stop procedure in each respective subsection in reverse order.

**Figure 12-17. DDR2/mDDR Memory Controller Power Sleep Controller Diagram**



### 12.2.16.1 DDR2/mDDR Memory Controller Clock Stop Procedure

**NOTE:** If a data access occurs to the DDR2/mDDR memory after completing steps 1-4, the DLL will wake up and lock, then the MCLK will turn on and the access will be performed. Following steps 5 and 6, in which the clocks are disabled, all DDR2/mDDR memory accesses are not possible until the clocks are reenabled.

In power-down mode, the DDR2/mDDR memory controller input clocks (VCLK and 2X\_CLK) may not be gated off. This is a limitation of the DDR2/mDDR controller. For this reason, power-down mode is best suited as a power savings mode when SDRAM is being used intermittently and the system requires power savings as well as a short recovery time. You may use self-refresh mode if you desire additional power savings from disabling clocks.

To achieve maximum power savings VCLK, MCLK, 2X\_CLK, DDR\_CLK, and  $\overline{\text{DDR\_CLK}}$  should be gated off. The procedure for clock gating is described in the following steps.

1. Allow software to complete the desired DDR transfers.
2. Change the SR\_PD bit to 0 and set the LPMODEN bit to 1 in the DDR2 SDRAM refresh control register (SDRCR) to enable self-refresh mode. The DDR2/mDDR memory controller will complete any outstanding accesses and backlogged refresh cycles and then place the external DDR2/mDDR memory in self-refresh mode.
3. Set the MCLKSTOPEN bit in SDRCR to 1. This enables the DDR2/mDDR memory controller to shut off the MCLK.
4. Wait 150 CPU clock cycles to allow the MCLK to stop.
5. Program the PSC to disable the DDR2/mDDR memory controller VCLK. You must not disable VCLK in power-down mode; use only for self-refresh mode (see notes in this section).
6. For maximum power savings, the PLL/PLL1 should be placed in bypass and powered-down mode to disable 2X\_CLK. You must not disable 2X\_CLK in power-down mode; use only for self-refresh mode (see notes in this section). For information on programming PLL1, see the *Phase-Locked Loop Controller (PLL1)* chapter.

To turn clocks back on:

1. Place the PLL/PLL1 in PLL mode to start 2X\_CLK to the DDR2/mDDR memory controller.
2. Once 2X\_CLK is stable, program the PSC to enable VCLK.
3. Set the RESET\_PHY bit in the DDR PHY reset control register (DRPYRCR) to 1. This resets the DDR2/mDDR memory controller PHY. This bit will self-clear to 0 when reset is complete.
4. Clear the MCLKSTOPEN bit in SDRCR to 0.
5. Clear the LPMODEN bit in the DDR2 SDRAM refresh control register (SDRCR) to 0.

### 12.2.17 Emulation Considerations

The DDR2/mDDR memory controller will remain fully functional during emulation halts to allow emulation access to external memory.

**NOTE:** VTP IO calibration must be performed before emulation tools attempt to access the register or data space of the DDR2/mDDR memory controller. A bus lock-up condition will occur if the emulation tool attempts to access the register or data space of the DDR2/mDDR memory controller before completing VTP IO calibration. See [Section 12.2.12](#) for information on VTP IO calibration.



### 12.3 Supported Use Cases

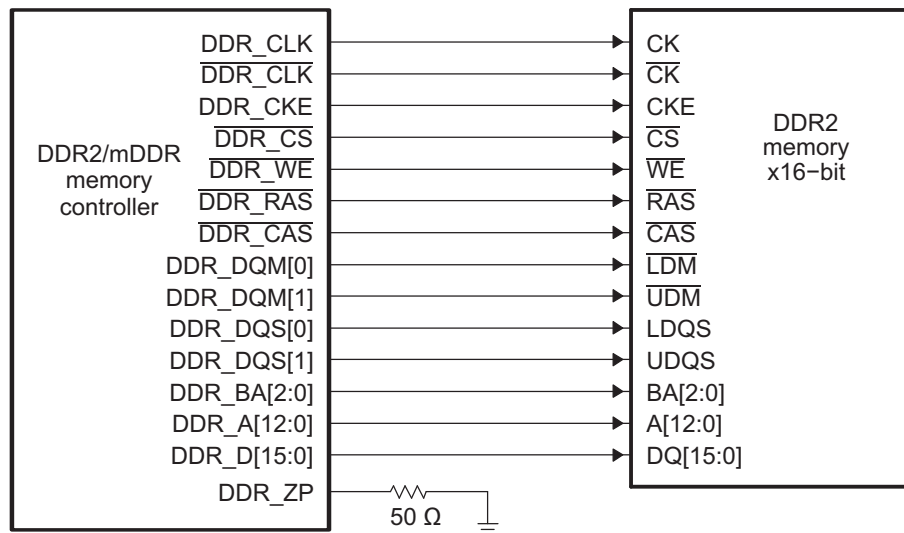
The DDR2/mDDR memory controller allows a high degree of programmability for shaping DDR2/mDDR accesses. The programmability inherent to the DDR2/mDDR memory controller provides the DDR2/mDDR memory controller with the flexibility to interface with a variety of DDR2/mDDR devices. By programming the SDRAM configuration register (SDCR), SDRAM refresh control register (SDRCR), SDRAM timing register 1 (SDTIMR1), and SDRAM timing register 2 (SDTIMR2), the DDR2/mDDR memory controller can be configured to meet the data sheet specification for DDR2 SDRAM as well as mDDR memory devices.

This section presents an example describing how to interface the DDR2 memory controller to a DDR2/mDDR-400 device. The DDR2/mDDR memory controller is assumed to be operating at 150 MHz. A similar procedure can be followed when interfacing to a mDDR memory device.

#### Connecting the DDR2/mDDR Memory Controller to DDR2/mDDR Memory

Figure 12-18 shows how to connect the DDR2/mDDR memory controller to a DDR2 device. Figure 12-18 displays a 16-bit interface; you can see that all signals are point-to-point connection.

Figure 12-18. Connecting DDR2/mDDR Memory Controller to a 16-Bit DDR2 Memory





## Configuring Memory-Mapped Registers to Meet DDR2 Specification

As previously stated, four memory-mapped registers must be programmed to configure the DDR2/mDDR memory controller to meet the data sheet specification of the attached DDR2/mDDR device. The registers are:

- SDRAM configuration register (SDCR)
- SDRAM refresh control register (SDRCR)
- SDRAM timing register 1 (SDTIMR1)
- SDRAM timing register 2 (SDTIMR2)

In addition to these registers, the DDR PHY control register (DRPYC1R) must also be programmed. The configuration of DRPYC1R is not dependent on the DDR2 device specification but rather on the board layout.

The following sections describe how to configure each of these registers. See [Section 12.4](#) for more information on the DDR2/mDDR memory controller registers.

---

**NOTE:** When interfacing the DDR2/mDDR memory controller to a mDDR device, the SDRAM configuration register 2 (SDCR2) must be programmed in addition to the registers mentioned above.

---

### Configuring SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) contains register fields that configure the DDR2/mDDR memory controller to match the data bus width, CAS latency, number of banks, and page size of the attached memory. In this example, we assume the following DDR2 configuration:

- Data bus width = 16 bits
- CAS latency = 3
- Number of banks = 8
- Page size = 1024 words

[Table 12-15](#) shows the resulting SDCR configuration. Note that the value of the TIMING\_UNLOCK field is dependent on whether or not it is desirable to unlock SDTIMR1 and SDTIMR2. The TIMING\_UNLOCK bit should only be set to 1 when the SDTIMR1 and SDTIMR2 needs to be updated.

**Table 12-15. SDCR Configuration**

Field	Value	Function Selection
TIMING_UNLOCK	x	Set to 1 to unlock the SDRAM timing register 1 and SDRAM timing register 2. Cleared to 0 to lock the SDRAM timing register 1 and SDRAM timing register 2.
NM	1h	To configure the DDR2/mDDR memory controller for a 16-bit data bus width.
CL	3h	To select a CAS latency of 3.
IBANK	3h	To select 8 internal DDR2 banks.
PAGESIZE	2h	To select 1024-word page size.

## Configuring SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) configures the DDR2/mDDR memory controller to meet the refresh requirements of the attached memory device. SDRCR also allows the DDR2/mDDR memory controller to enter and exit self refresh and enable and disable the MCLK stopping. In this example, we assume that the DDR2/mDDR memory controller is not in self-refresh mode or power-down mode and that MCLK stopping is disabled.

The RR field in SDRCR is defined as the rate at which the attached memory device is refreshed in DDR2/mDDR cycles. The value of this field may be calculated using the following equation:

$$RR = \text{DDR2/mDDR clock frequency} \times \text{DDR2/mDDR memory refresh period}$$

Table 12-16 displays the DDR2-400 refresh rate specification.

**Table 12-16. DDR2 Memory Refresh Specification**

Symbol	Description	Value
$t_{REF}$	Average Periodic Refresh Interval	7.8 $\mu$ s

Therefore, the following results assuming 150 MHz DDR2/mDDR clock frequency.

$$RR = 150 \text{ MHz} \times 7.8 \text{ } \mu\text{s} = 1170$$

Therefore,  $RR = 1170 = 492h$ .

Table 12-17 shows the resulting SDRCR configuration.

**Table 12-17. SDRCR Configuration**

Field	Value	Function Selection
LPMODEN	0	DDR2/mDDR memory controller is not in power-down mode.
MCLKSTOP_EN	0	MCLK stopping is disabled.
SR_PD	0	Leave a default value.
RR	492h	Set to 492h DDR2 clock cycles to meet the DDR2/mDDR memory refresh rate requirement.

## Configuring SDRAM Timing Registers (SDTMR1 and SDTMR2)

The SDRAM timing register 1 (SDTMR1) and SDRAM timing register 2 (SDTMR2) configure the DDR2/mDDR memory controller to meet the data sheet timing parameters of the attached memory device. Each field in SDTMR1 and SDTMR2 corresponds to a timing parameter in the DDR2/mDDR data sheet specification. [Table 12-18](#) and [Table 12-19](#) display the register field name and corresponding DDR2 data sheet parameter name along with the data sheet value. These tables also provide a formula to calculate the register field value and displays the resulting calculation. Each of the equations include a minus 1 because the register fields are defined in terms of DDR2/mDDR clock cycles minus 1. See [Section 12.4.4](#) and [Section 12.4.5](#) for more information.

**Table 12-18. SDTMR1 Configuration**

Register Field Name	DDR2 Data Manual Parameter Name	Description	Data Manual Value (nS)	Formula (Register field must be $\geq$ )	Register Value
T_RFC	$t_{RFC}$	Refresh cycle time	127.5	$(t_{RFC} \times f_{DDR2/mDDR\_CLK}) - 1$	19
T_RP	$t_{RP}$	Precharge command to refresh or activate command	15	$(t_{RP} \times f_{DDR2/mDDR\_CLK}) - 1$	2
T_RCD	$t_{RCD}$	Activate command to read/write command	15	$(t_{RCD} \times f_{DDR2/mDDR\_CLK}) - 1$	2
T_WR	$t_{WR}$	Write recovery time	15	$(t_{WR} \times f_{DDR2/mDDR\_CLK}) - 1$	2
T_RAS	$t_{RAS}$	Active to precharge command	40	$(t_{RAS} \times f_{DDR2/mDDR\_CLK}) - 1$	5
T_RC	$t_{RC}$	Activate to Activate command in the same bank	55	$(t_{RC} \times f_{DDR2/mDDR\_CLK}) - 1$	8
T_RRD <sup>(1)</sup>	$t_{RRD}$	Activate to Activate command in a different bank	10	$((4 \times t_{RRD}) + (2 \times t_{CK})) / (4 \times t_{CK}) - 1$	1
T_WTR	$t_{WTR}$	Write to read command delay	10	$(t_{WTR} \times f_{DDR2/mDDR\_CLK}) - 1$	1

<sup>(1)</sup> The formula for the T\_RRD field applies only for 8 bank DDR2/mDDR memories; when interfacing to DDR2/mDDR memories with less than 8 banks, the T\_RRD field should be calculated using the following formula:  $(t_{RRD} \times f_{DDR2/mDDR\_CLK}) - 1$ .

**Table 12-19. SDTMR2 Configuration**

Register Field Name	DDR2 Data Manual Parameter Name	Description	Data Manual Value	Formula (Register field must be $\geq$ )	Register Value
T_RASMAX	$t_{RAS(MAX)}$	Active to precharge command	70 $\mu$ S	$t_{RAS(MAX)} / \text{DDR refresh rate} - 1$	8
T_XP	$t_{XP}$	Exit power down to a non-read command	2( $t_{CK}$ cycles)	If $t_{XP} > t_{CKE}$ , then $T\_XP = t_{XP} - 1$ , else $T\_XP = t_{CKE} - 1$	2
T_XSNR	$t_{XSNR}$	Exit self refresh to a non-read command	137.5 nS	$(t_{XSNR} \times f_{DDR2/mDDR\_CLK}) - 1$	18
T_XSRD	$t_{XSRD}$	Exit self refresh to a read command	200 ( $t_{CK}$ cycles)	$t_{XSRD} - 1$	199
T_RTP	$t_{RTP}$	Read to precharge command delay	15 nS	$(t_{RTP} \times f_{DDR2/mDDR\_CLK}) - 1$	1
T_CKE	$t_{CKE}$	CKE minimum pulse width	3 ( $t_{CK}$ cycles)	$t_{CKE} - 1$	2

### Configuring DDR PHY Control Register (DRPYC1R)

The DDR PHY control register (DRPYC1R) contains a read latency (RL) field that helps the DDR2/mDDR memory controller determine when to sample read data. The RL field should be programmed to a value equal to the CAS latency plus the round trip board delay minus 1. The minimum RL value is CAS latency plus 1 and the maximum RL value is CAS latency plus 2 (again, the RL field would be programmed to these values minus 1). [Table 12-20](#) shows the resulting DRPYC1R configuration.

When calculating round trip board delay the signals of primary concern are the differential clock signals (DDR\_CLK and  $\overline{\text{DDR\_CLK}}$ ) and data strobe signals (DDR\_DQS). For these signals, calculate the round trip board delay from the DDR memory controller to the memory and then choose the maximum delay to determine the RL value. In this example, we will assume the round trip board delay is one DDR\_CLK cycle; therefore, RL can be calculated as:

$$\text{RL} = \text{CAS latency} + \text{round trip board delay} - 1 = 4 + 1 - 1 = 4$$

**Table 12-20. DRPYC1R Configuration**

Field	Value	Function Selection
EXT_STRBEN	1h	Programs to select external strobe gating
RL	4h	Read latency is equal to CAS latency plus round trip board delay for data minus 1
PWRDNEN	0	Programmed to power up the DDR2/mDDR memory controller receivers

## 12.4 Registers

Table 12-21 lists the memory-mapped registers for the DDR2/mDDR memory controller. Note that the VTP IO control register (VTPIO\_CTL) resides in the System Configuration Module.

**Table 12-21. DDR2/mDDR Memory Controller Registers**

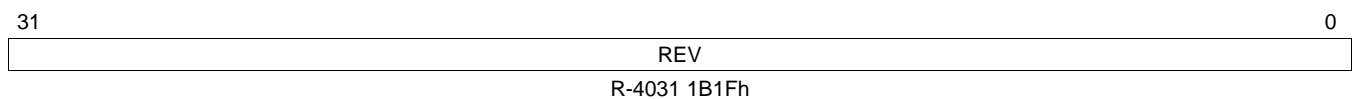
Address Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	Revision ID Register (REVID)
4h	SDRSTAT	SDRAM Status Register	Section 12.4.1
8h	SDCR	SDRAM Configuration Register	Section 12.4.2
Ch	SDRCR	SDRAM Refresh Control Register	Section 12.4.3
10h	SDTIMR1	SDRAM Timing Register 1	Section 12.4.4
14h	SDTIMR2	SDRAM Timing Register 2	Section 12.4.5
1Ch	SDCR2	SDRAM Configuration Register 2	Section 12.4.6
20h	PBBPR	Peripheral Bus Burst Priority Register	Section 12.4.7
40h	PC1	Performance Counter 1 Register	Section 12.4.8
44h	PC2	Performance Counter 2 Register	Section 12.4.9
48h	PCC	Performance Counter Configuration Register	Section 12.4.10
4Ch	PCMRS	Performance Counter Master Region Select Register	Section 12.4.11
50h	PCT	Performance Counter Time Register	Performance Counter Time Register (PCT)
60h	DRPYRCR	DDR PHY Reset Control Register	Section 12.4.12
C0h	IRR	Interrupt Raw Register	Section 12.4.13
C4h	IMR	Interrupt Masked Register	Section 12.4.14
C8h	IMSR	Interrupt Mask Set Register	Section 12.4.15
CCh	IMCR	Interrupt Mask Clear Register	Section 12.4.16
E4h	DRPYC1R	DDR PHY Control Register 1	Section 12.4.17
01E2 C000h <sup>(1)</sup>	VTPIO_CTL	VTP IO Control Register	Section 10.5.18
01E2 C004h <sup>(1)</sup>	DDR_SLEW	DDR Slew Register	Section 10.5.19

<sup>(1)</sup> This register resides in the register space of the System Configuration (SYSCFG) Module. It is listed in the register space of the DDR2/mDDR controller because it is applicable to the DDR2/mDDR controller.

### Revision ID Register (REVID)

The revision ID register (REVID) contains the current revision ID for the DDR2/mDDR memory controller. The REVID is shown in Figure 12-19 and described in Table 12-22.

**Figure 12-19. Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

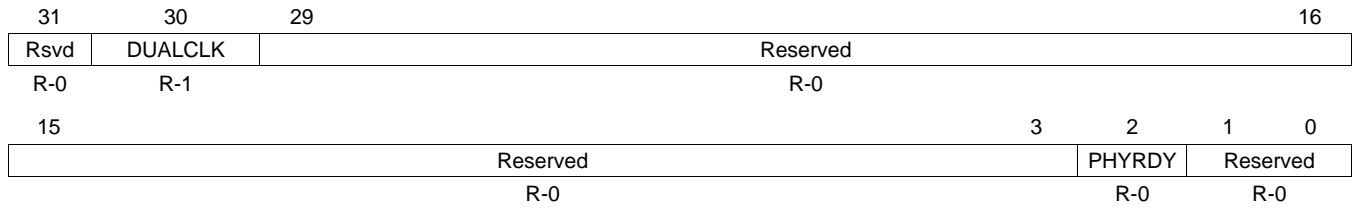
**Table 12-22. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4031 1B1Fh	Revision ID value of the DDR2/mDDR memory controller.

### 12.4.1 SDRAM Status Register (SDRSTAT)

The SDRAM status register (SDRSTAT) is shown in [Figure 12-20](#) and described in [Table 12-23](#).

**Figure 12-20. SDRAM Status Register (SDRSTAT)**



LEGEND: R = Read only; -n = value after reset

**Table 12-23. SDRAM Status Register (SDRSTAT) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	DUALCLK	0	Dual clock. Specifies whether the VCLK and MCLK inputs are asynchronous. This bit should always be read as 1. VCLK and MCLK are not asynchronous.
		1	VCLK and MCLK are asynchronous.
29-3	Reserved	0	Reserved
2	PHYRDY	0	DDR2/mDDR memory controller DLL ready. Specifies whether the DDR2/mDDR memory controller DLL is powered up and locked. DLL is not ready, either powered down, in reset, or not locked.
		1	DLL is powered up, locked, and ready for operation.
1-0	Reserved	0	Reserved

## 12.4.2 SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) contains fields that program the DDR2/mDDR memory controller to meet the specification of the attached DDR2/mDDR memory. These fields configure the DDR2/mDDR memory controller to match the data bus width, CAS latency, number of internal banks, and page size of the attached DDR2/mDDR memory. Writing to the DDRDRIVE[1:0], CL, IBANK, and PAGESIZE bit fields causes the DDR2/mDDR memory controller to start the DDR2/mDDR SDRAM initialization sequence. The SDCR is shown in [Figure 12-21](#) and described in [Table 12-24](#).

**Figure 12-21. SDRAM Configuration Register (SDCR)**

31		28		27		26		25		24	
Reserved				DDR2TERM1		IBANK_POS		MSDRAMEN		DDRDRIVE1	
R-0				R/W-1		R/W-0		R/W-0		R/W-0	
23		22		21		20		19		18	
BOOTUNLOCK		DDR2DDQS		DDR2TERM0		DDR2EN		DDRDLL_DIS		DDRDRIVE0	
R/W-0		R/W-0		R/W-0		R/W-1		R/W-0		R/W-1	
15		14		13		12		11		9	
TIMUNLOCK		NM		Reserved				CL		Reserved	
R/W-0		R/W-1		R-0				R/W-5h		R-0	
7		6		4		3		2		0	
Reserved		IBANK				Reserved		PAGESIZE			
R-0		R/W-2h				R-0		R/W-0			

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-24. SDRAM Configuration Register (SDCR) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	DDR2TERM1	0-3h	DDR2 termination resistor value. This bit is used in conjunction with the DDR2TERM0 bit to make a 2-bit field. This bit is writeable only when the BOOTUNLOCK bit is unlocked. See the DDR2TERM0 bit. Note that the reset value of DDR2TERM[1:0] = 10, these bits must be cleared and forced to 00 to disable the termination because the ODT feature is not supported.
26	IBANK_POS	0 1	Internal Bank position. 0 Normal addressing 1 Special addressing. Typically used with mobile DDR partial array self-refresh.
25	MSDRAMEN	0 1	Mobile SDRAM enable. Use this bit in conjunction with DDR2EN, DDREN, and SDRAMEN to enable/disable mobile SDRAM. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the MSDRAMEN bit. 0 Disable mobile SDRAM 1 Enable mobile SDRAM
24	DDRDRIVE1	0-3h	SDRAM drive strength. This bit is used in conjunction with the DDRDRIVE0 bit to make a 2-bit field. This bit is writeable only when the BOOTUNLOCK bit is unlocked. See the DDRDRIVE0 bit.
23	BOOTUNLOCK	0 1	Boot Unlock. Controls the write permission settings for the DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bit fields. To change these bits, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bits. 0 DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bit fields may not be changed. 1 DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bit fields may be changed.

**Table 12-24. SDRAM Configuration Register (SDCR) Field Descriptions (continued)**

Bit	Field	Value	Description
22	DDR2DDQS	0 1	DDR2 SDRAM differential DQS enable. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2DDQS bit.</li> </ol> Single-ended DQS Reserved
21	DDR2TERM0	0-3h 0 1h-3h	DDR2 termination resistor value. This bit is used in conjunction with the DDR2TERM1 bit to make a 2-bit field. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2TERM[1:0] bits.</li> </ol> Note that the reset value of DDR2TERM[1:0] = 10, these bits must be cleared and forced to 00 to disable the termination because the ODT feature is not supported. Disable termination Reserved
20	DDR2EN	0 1	DDR2 enable. This bit is used in conjunction with the DDREN and SDRAMEN bits to enable/disable DDR2. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2EN bit.</li> </ol> Disable DDR2 Enable DDR2
19	DDRDLL_DIS	0 1	DLL disable for DDR SDRAM. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDRDLL_DIS bit.</li> </ol> Enable DLL Disable DLL inside DDR SDRAM
18	DDRDRIVE0	0-3h 0 1h 2h 3h	SDRAM drive strength. This bit is used in conjunction with the DDRDRIVE1 bit to make a 2-bit field. The DDRDRIVE[1:0] bits configure the output driver impedance control value of the SDRAM memory. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDRDRIVE[1:0] bits.</li> </ol> For DDR2, normal drive strength. For mobile DDR, full drive strength. For DDR2, weak drive strength. For mobile DDR, 1/2 drive strength. For DDR2, reserved. For mobile DDR, 1/4 drive strength. For DDR2, reserved. For mobile DDR, 1/8 drive strength.
17	DDREN	0 1	DDR enable. This bit is used in conjunction with the DDR2EN and SDRAMEN bits to enable/disable DDR. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDREN bit.</li> </ol> Disable DDR Enable DDR
16	SDRAMEN	0 1	SDRAM enable. This bit is used in conjunction with the DDR2EN and DDREN bits to enable/disable SDRAM. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the BOOTUNLOCK bit.</li> <li>Write a 0 to the BOOTUNLOCK bit along with the desired value of the SDRAMEN bit.</li> </ol> Disable SDRAM Enable SDRAM



**Table 12-24. SDRAM Configuration Register (SDCR) Field Descriptions (continued)**

Bit	Field	Value	Description
15	TIMUNLOCK	0 1	Timing unlock. Controls the write permission settings for the CL bit field, and the SDRAM timing register 1 (SDTIMR1) and the SDRAM timing register 2 (SDTIMR2) bit fields. To change these bits, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the TIMUNLOCK bit.</li> <li>Write a 0 to the TIMUNLOCK bit along with the desired value of the CL bit and SDTIMR1 and SDTIMR2 bit fields.</li> </ol> CL bit, and SDTIMR1 and SDTIMR2 bit fields may not be changed. CL bit, and SDTIMR1 and SDTIMR2 bit fields may be changed.
14	NM	0 1	SDRAM data bus width. 0 Reserved 1 16-bit bus width.
13-12	Reserved	0	Reserved
11-9	CL	0-7h 0-1h 2h 3h 4h 5h 6h-7h	SDRAM CAS latency. This bit is writeable only when the TIMUNLOCK bit is unlocked. To change this bit value, use the following sequence: <ol style="list-style-type: none"> <li>Write a 1 to the TIMUNLOCK bit.</li> <li>Write a 0 to the TIMUNLOCK bit along with the desired value of the CL bit.</li> </ol> Reserved CAS Latency = 2 CAS Latency = 3 CAS Latency = 4 CAS Latency = 5 Reserved
8-7	Reserved	0	Reserved
6-4	IBANK	0-7h 0 1h 2h 3h 4h-7h	Internal SDRAM bank setup. Defines the number of internal banks on the external SDRAM device. 1 bank 2 banks 4 banks 8 banks Reserved
3	Reserved	0	Reserved
2-0	PAGESIZE	0-7h 0 1h 2h 3h 4h-7h	Page Size. Defines the page size of the SDRAM device. 256-word page requiring 8 column address bits. 512-word page requiring 9 column address bits. 1024-word page requiring 10 column address bits. 2048-word page requiring 11 column address bits. Reserved

### 12.4.3 SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) is used to configure the DDR2/mDDR memory controller to:

- Enter and Exit the self-refresh and power-down states.
- Enable and disable MCLK, stopping when in the self-refresh state.
- Meet the refresh requirement of the attached DDR2/mDDR device by programming the rate at which the DDR2/mDDR memory controller issues autorefresh commands.

The SDRCR is shown in [Table 12-25](#) and described in [Figure 12-22](#).

**Figure 12-22. SDRAM Refresh Control Register (SDRCR)**

31	30	29	24	23	22	16
LPMODEN	MCLKSTOPEN	Reserved	SR_PD	Reserved		
R/W-0	R/W-0	R-0	R/W-0	R-0		
15						0
RR						
R/W-884h						

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-25. SDRAM Refresh Control Register (SDRCR) Field Descriptions**

Bit	Field	Value	Description
31	LPMODEN	0	Low-power mode enable. Disable low-power mode.
		1	Enable low-power mode. The state of bit SR_PD selects either self-refresh or power-down mode.
30	MCLKSTOPEN	0	MCLK stop enable. Disables MCLK stopping, MCLK may not be stopped.
		1	Enables MCLK stopping, MCLK may be stopped. The LPMODEN bit must be set to 1 before setting the MCLKSTOPEN bit to 1.
29-24	Reserved	0	Reserved
23	SR_PD	0	Self-refresh or Power-down select. This bit is only in effect when the LPMODEN bit is set to 1; this bit is ignored when the LPMODEN bit is cleared to 0.
		1	Self-refresh mode. Power-down mode.
22-16	Reserved	0	Reserved
15-0	RR	0-FFFFh	Refresh rate. Defines the rate at which the attached SDRAM devices will be refreshed. The value of this field may be calculated with the following equation: $RR = SDRAM\ frequency / SDRAM\ refresh\ rate$ where <i>SDRAM refresh rate</i> is derived from the SDRAM data sheet.

### 12.4.4 SDRAM Timing Register 1 (SDTIMR1)

The SDRAM timing register 1 (SDTIMR1) configures the DDR2/mDDR memory controller to meet many of the AC timing specification of the DDR2/mDDR memory. The SDTIMR1 is programmable only when the TIMUNLOCK bit is set to 1 in the SDRAM configuration register (SDCR). Note that DDR\_CLK is equal to the period of the DDR\_CLK signal. See the DDR2/mDDR memory data sheet for information on the appropriate values to program each field. The SDTIMR1 is shown in Figure 12-23 and described in Table 12-26.

**Figure 12-23. SDRAM Timing Register 1 (SDTIMR1)**

31	25	24	22	21	19	18	16
T_RFC		T_RP		T_RCD		T_WR	
R/W-Fh		R/W-2h		R/W-2h		R/W-2h	
15	11	10	6	5	3	2	1 0
T_RAS		T_RC		T_RRD		Rsvd	T_WTR
R/W-6h		R/W-9h		R/W-1		R-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-26. SDRAM Timing Register 1 (SDTIMR1) Field Descriptions**

Bit	Field	Value	Description
31-25	T_RFC	0-7Fh	Specifies the minimum number of DDR_CLK cycles from a refresh or load mode command to a refresh or activate command, minus 1. Corresponds to the $t_{rfc}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RFC = (t_{rfc}/DDR\_CLK) - 1$
24-22	T_RP	0-7h	Specifies the minimum number of DDR_CLK cycles from a precharge command to a refresh or activate command, minus 1. Corresponds to the $t_{rp}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RP = (t_{rp}/DDR\_CLK) - 1$
21-19	T_RCD	0-7h	Specifies the minimum number of DDR_CLK cycles from an activate command to a read or write command, minus 1. Corresponds to the $t_{rcd}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RCD = (t_{rcd}/DDR\_CLK) - 1$
18-16	T_WR	0-7h	Specifies the minimum number of DDR_CLK cycles from the last write transfer to a precharge command, minus 1. Corresponds to the $t_{wr}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_WR = (t_{wr}/DDR\_CLK) - 1$ When the value of this field is changed from its previous value, the initialization sequence will begin.
15-11	T_RAS	0-1Fh	Specifies the minimum number of DDR_CLK cycles from an activate command to a precharge command, minus 1. Corresponds to the $t_{ras}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RAS = (t_{ras}/DDR\_CLK) - 1$ $T\_RAS$ must be greater than or equal to $T\_RCD$ .
10-6	T_RC	0-1Fh	Specifies the minimum number of DDR_CLK cycles from an activate command to an activate command, minus 1. Corresponds to the $t_{rc}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RC = (t_{rc}/DDR\_CLK) - 1$
5-3	T_RRD	0-7h	Specifies the minimum number of DDR_CLK cycles from an activate command to an activate command in a different bank, minus 1. Corresponds to the $t_{rrd}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RRD = (t_{rrd}/DDR\_CLK) - 1$ For an 8 bank DDR2/mDDR device, this field must be equal to $((4 \times t_{RRD}) + (2 \times t_{CK})) / (4 \times t_{CK}) - 1$ .
2	Reserved	0	Reserved
1-0	T_WTR	0-3h	Specifies the minimum number of DDR_CLK cycles from the last write to a read command, minus 1. Corresponds to the $t_{wtr}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_WTR = (t_{wtr}/DDR\_CLK) - 1$

### 12.4.5 SDRAM Timing Register 2 (SDTMR2)

Like the SDRAM timing register 1 (SDTMR1), the SDRAM timing register 2 (SDTMR2) also configures the DDR2/mDDR memory controller to meet the AC timing specification of the DDR2/mDDR memory. The SDTMR2 is programmable only when the TIMUNLOCK bit is set to 1 in the SDRAM configuration register (SDCR). Note that DDR\_CLK is equal to the period of the DDR\_CLK signal. See the DDR2/mDDR data sheet for information on the appropriate values to program each field. SDTMR2 is shown in Figure 12-24 and described in Table 12-27.

**Figure 12-24. SDRAM Timing Register 2 (SDTMR2)**

31	30	27	26	25	24	23	22	16
Rsvd	T_RASMAX		T_XP		T_ODT		T_XSNR	
R-0	R/W-8h		R/W-2h		R/W-2h		R/W-32h	
15	T_XSRD				T_RTP		T_CKE	
	R/W-A7h				R/W-1		R/W-2h	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-27. SDRAM Timing Register 2 (SDTMR2) Field Descriptions**

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-27	T_RASMAX	0-Fh	Specifies the maximum number of refresh rate intervals from Activate to Precharge command. Corresponds to the $t_{ras}$ AC timing parameter and the refresh rate in the DDR2/mDDR data sheet. Calculate by: $T\_RASMAX = (t_{ras\_max}/refresh\_rate) - 1$ Round down to the nearest cycle.
26-25	T_XP	0-3h	Specifies the minimum number of DDR_CLK cycles from Power Down exit to any other command except a read command, minus 1. Corresponds to the $t_{xp}$ or $t_{cke}$ AC timing parameter in the DDR2/mDDR data sheet. This field must satisfy the greater of $t_{xp}$ or $t_{cke}$ . If $t_{xp} > t_{cke}$ , then calculate by $T\_XP = t_{xp} - 1$ If $t_{xp} < t_{cke}$ , then calculate by $T\_XP = t_{cke} - 1$
24-23	T_ODT	0-3h	Specifies the minimum number of DDR_CLK cycles from ODT enable to write data driven for DDR2 SDRAM. T_ODT must be equal to (CAS latency - tAOND - 1). T_ODT must be less than CAS latency minus 1. This feature is not supported because the DDR_ODT signal is not pinned out.
22-16	T_XSNR	0-7Fh	Specifies the minimum number of DDR_CLK cycles from a self_refresh exit to any other command except a read command, minus 1. Corresponds to the $t_{xsnr}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_XSNR = (t_{xsnr}/DDR\_CLK) - 1$
15-8	T_XSRD	0-FFh	Specifies the minimum number of DDR_CLK cycles from a self_refresh exit to a read command, minus 1. Corresponds to the $t_{xsrld}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_XSRD = t_{xsrld} - 1$
7-5	T_RTP	0-7h	Specifies the minimum number of DDR_CLK cycles from a last read command to a precharge command, minus 1. Corresponds to the $t_{rtp}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_RTP = (t_{rtp}/DDR\_CLK) - 1$
4-0	T_CKE	0-1Fh	Specifies the minimum number of DDR_CLK cycles between transitions on the DDR_CKE pin, minus 1. Corresponds to the $t_{cke}$ AC timing parameter in the DDR2/mDDR data sheet. Calculate by: $T\_CKE = t_{cke} - 1$

### 12.4.6 SDRAM Configuration Register 2 (SDCR2)

The SDRAM configuration register 2 (SDCR2) contains fields to configure partial array self-refresh and row size of the mDDR. This register is applicable only when the IBANK\_POS bit in the SDRAM configuration register (SDCR) is set to 1 for special addressing. Writing to the PASR and ROWSIZE bit fields will cause the DDR2/mDDR memory controller to start the DDR2/mDDR SDRAM initialization sequence. SDCR2 is shown in Figure 12-25 and described in Table 12-28.

**Figure 12-25. SDRAM Configuration Register 2 (SDCR2)**

31	Reserved	19	18	16
	R-0			PASR R/W-0
15	Reserved	3	2	0
	R-0			ROWSIZE R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-28. SDRAM Configuration Register 2 (SDCR2) Field Descriptions**

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18-16	PASR	0-7h	Partial array self-refresh.
		0	4 banks will be refreshed.
		1h	2 banks will be refreshed.
		2h	1 bank will be refreshed.
		3h-4h	Reserved
		5h	1/2 bank will be refreshed.
		6h	1/4 bank will be refreshed.
		7h	Reserved
15-3	Reserved	0	Reserved
2-0	ROWSIZE	0-7h	Row size. Defines the number of row address bit for DDR device.
		0	9 row address bits
		1h	10 row address bits
		2h	11 row address bits
		3h	12 row address bits
		4h	13 row address bits
		5h	14 row address bits
		6h	15 row address bits
		7h	16 row address bits

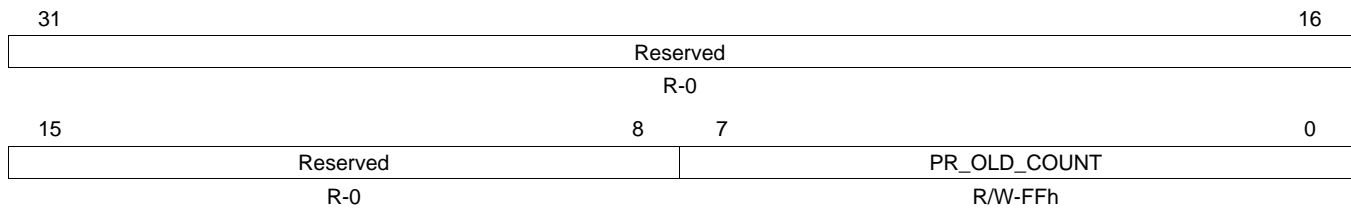
### 12.4.7 Peripheral Bus Burst Priority Register (PBBPR)

The peripheral bus burst priority register (PBBPR) helps prevent command starvation within the DDR2/mDDR memory controller. To avoid command starvation, the DDR2/mDDR memory controller momentarily raises the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR\_OLD\_COUNT bit sets the number of transfers that must be made before the DDR2/mDDR memory controller raises the priority of the oldest command. See [Section 12.2.6.2](#) for more details on command starvation.

Proper configuration of the PBBPR is critical to correct system operation. The DDR2/mDDR memory controller always prioritizes accesses to open rows as highest, if there is any bank conflict regardless of master priority. This is done to allow most efficient utilization of the DDR2/mDDR. However, it could lead to excessive blocking of high priority masters. If the PR\_OLD\_COUNT bits are cleared to 00h, then the DDR2/mDDR memory controller always honors the master priority, regardless of open row/bank status. For most systems, the PBBPR should be set to a moderately low value to provide an acceptable balance of DDR2/mDDR efficiency and latency for high priority masters (for example, 10h or 20h).

The PBBPR is shown in [Figure 12-26](#) and described in [Table 12-29](#).

**Figure 12-26. Peripheral Bus Burst Priority Register (PBBPR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-29. Peripheral Bus Burst Priority Register (PBBPR) Field Descriptions**

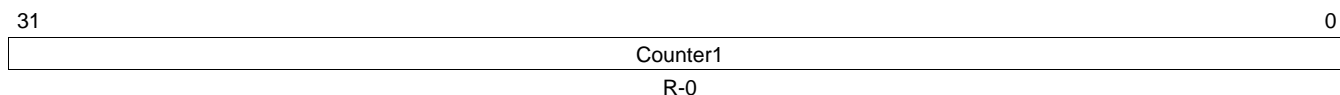
Bit	Field	Value	Description
31-8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7-0	PR_OLD_COUNT	0-FFh	Priority raise old counter. Specifies the number of memory transfers after which the DDR2/mDDR memory controller will elevate the priority of the oldest command in the command FIFO. Clearing to 00h will ensure master priority is strictly honored (at the cost of decreased DDR2/mDDR memory controller efficiency, as open row will always be closed immediately if any bank conflict occurs). Recommended setting for typical system operation is between 10h and 20h.
		0	1 memory transfer
		1h	2 memory transfers
		2h	3 memory transfers
		3h-FFh	4 to 256 memory transfers

### 12.4.8 Performance Counter 1 Register (PC1)

For debug or gathering performance statistics, the PC1 and PC2 counters and associated configuration registers are provided. These are intended for debug and analysis only. By configuring the performance counter configuration register (PCC) to define the type of statistics to gather and configuring the performance counter master region select register (PCMRS) to filter accesses only to specific chip select regions, performing system applications and then reading these counters, different statistics can be gathered. To reset the counters, you must reset (mod\_g\_rst\_n) the DDR2/mDDR memory controller through the PSC. For details on the PSC, see the *Power and Sleep Controller (PSC)* chapter.

The performance counter 1 register (PC1) is shown in [Figure 12-27](#) and described in [Table 12-30](#).

**Figure 12-27. Performance Counter 1 Register (PC1)**



LEGEND: R = Read only; -n = value after reset

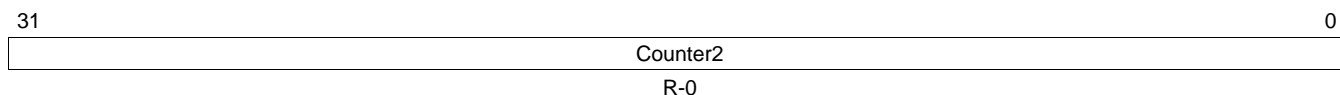
**Table 12-30. Performance Counter 1 Register (PC1) Field Descriptions**

Bit	Field	Value	Description
31-0	Counter1	0-FFFF FFFFh	32-bit counter that can be configured as specified in the performance counter configuration register (PCC) and the performance counter master region select register (PCMRS).

### 12.4.9 Performance Counter 2 Register (PC2)

The performance counter 2 register (PC2) is shown in [Figure 12-28](#) and described in [Table 12-31](#).

**Figure 12-28. Performance Counter 2 Register (PC2)**



LEGEND: R = Read only; -n = value after reset

**Table 12-31. Performance Counter 2 Register (PC2) Field Descriptions**

Bit	Field	Value	Description
31-0	Counter2	0-FFFF FFFFh	32-bit counter that can be configured as specified in the performance counter configuration register (PCC) and the performance counter master region select register (PCMRS).

### 12.4.10 Performance Counter Configuration Register (PCC)

The performance counter configuration register (PCC) is shown in [Figure 12-29](#) and described in [Table 12-32](#).

[Table 12-33](#) shows the possible filter configurations for the two performance counters. These filter configurations can be used in conjunction with a Master ID and/or an external chip select to obtain performance statistics for a particular master and/or an external chip select.

**Figure 12-29. Performance Counter Configuration Register (PCC)**

31	30	29	20	19	16
CNTR2_MSTID_EN	CNTR2_REGION_EN	Reserved		CNTR2_CFG	
R/W-0	R/W-0	R-0		R/W-1	
15	14	13	4	3	0
CNTR1_MSTID_EN	CNTR1_REGION_EN	Reserved		CNTR1_CFG	
R/W-0	R/W-0	R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-32. Performance Counter Configuration Register (PCC) Field Descriptions**

Bit	Field	Value	Description
31	CNTR2_MSTID_EN	0 1	Master ID filter enable for performance counter 2 register (PC2). Refer to <a href="#">Table 12-33</a> for details. 0 Master ID filter is disabled. PC2 counts accesses from all masters to DDR2/mDDR SDRAM. 1 Master ID filter is enabled. PC2 counts accesses from the master, corresponding to the Master ID value in the MST_ID2 bit field of the performance counter master region select register (PCMRS).
30	CNTR2_REGION_EN	0 1	Chip select filter enable for performance counter 2 register (PC2). Refer to <a href="#">Table 12-33</a> for details. 0 Chip select filter is disabled. PC2 counts total number of accesses (DDR2/mDDR SDRAM + DDR2/mDDR memory controller memory-mapped register accesses). The REGION_SEL2 bit field value in the performance counter master region select register (PCMRS) is a don't care. 1 Chip select filter is enabled. If the REGION_SEL2 bit field value in the performance counter master region select register (PCMRS) is: <b>REGION_SEL2 = 0:</b> PC2 counts accesses to DDR2/mDDR SDRAM memory. <b>REGION_SEL2 = 7h:</b> PC2 counts accesses to DDR2/mDDR memory controller memory-mapped registers.
29-20	Reserved	0	Any writes to these bit(s) must always have a value of 0.
19-16	CNTR2_CFG	0-Fh	Filter configuration for performance counter 2 register (PC2). Refer to <a href="#">Table 12-33</a> for details.
15	CNTR1_MSTID_EN	0 1	Master ID filter enable for performance counter 1 register (PC1). Refer to <a href="#">Table 12-33</a> for details. 0 Master ID filter is disabled. PC1 counts accesses from all masters to DDR2/mDDR SDRAM. 1 Master ID filter is enabled. PC1 counts accesses from the master, corresponding to the Master ID value in the MST_ID1 bit field of the performance counter master region select register (PCMRS).
14	CNTR1_REGION_EN	0 1	Chip select filter enable for performance counter 1 register (PC1). Refer to <a href="#">Table 12-33</a> for details. 0 Chip select filter is disabled. PC1 counts total number of accesses (DDR2/mDDR SDRAM + DDR2/mDDR memory controller memory-mapped register accesses). The REGION_SEL1 bit field value in the performance counter master region select register (PCMRS) is a don't care. 1 Chip select filter is enabled. If the REGION_SEL1 bit field value in the performance counter master region select register (PCMRS) is: <b>REGION_SEL1 = 0:</b> PC1 counts accesses to DDR2/mDDR SDRAM memory. <b>REGION_SEL1 = 7h:</b> PC1 counts accesses to DDR2/mDDR memory controller memory-mapped registers.
13-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.



**Table 12-32. Performance Counter Configuration Register (PCC) Field Descriptions (continued)**

Bit	Field	Value	Description
3-0	CNTR1_CFG	0-Fh	Filter configuration for performance counter 1 register (PC1). Refer to <a href="#">Table 12-33</a> for details.

**Table 12-33. Performance Counter Filter Configuration**

Performance Counter Configuration Register (PCC) Bit			
CNTR $n$ _CFG	CNTR $n$ _REGION_EN	CNTR $n$ _MSTID_EN	Description
0	0	0 or 1	<p><b>Counts the total number of READ/WRITE commands the external memory controller receives.</b></p> <p>The size of counter increments are determined by the size of the transfer and the default burst size (DBS). The counter breaks up transfers into sizes according to DBS. Therefore, counter increments for transfers aligned to DBS are equal to the transfer size divided by the DBS.</p>
1h	0	0	<p><b>Counts the total number of ACTIVATE commands the external memory controller issues to DDR2/mDDR memory.</b></p> <p>The counter increments by a value of 1 for every request to read/write data to a closed bank in DDR2/mDDR memory by the external memory controller.</p>
2h	0 or 1	0 or 1	<p><b>Counts the total number of READ commands (read accesses) the DDR2/mDDR memory controller receives.</b></p> <p>Counter increments for transfers aligned to the default burst size (DBS) are equal to the transfer size divided by the DBS.</p>
3h	0 or 1	0 or 1	<p><b>Counts the total number of WRITE commands the DDR2/mDDR memory controller receives.</b></p> <p>Counter increments for transfers aligned to the default burst size (DBS) are equal to the transfer size of data written to the DDR2/mDDR memory controller divided by the DBS.</p>
4h	0	0	<p><b>Counts the number of external memory controller cycles (DDR_CLK cycles) that the command FIFO is full.</b></p> <p>Use the following to calculate the counter value as a percentage:  <math>\% = \text{counter value} / \text{total DDR\_CLK cycles in a sample period}</math></p> <p>As the value of this counter approaches 100%, the DDR2/mDDR memory controller is approaching a congestion point where the command FIFO is full 100% of the time and a command will have to wait at the SCR to be accepted in the command FIFO.</p>
5h-7h	0	0	Reserved
8h	0 or 1	0 or 1	<p><b>Counts the number of commands (requests) in the command FIFO that require a priority elevation.</b></p> <p>To avoid command starvation, the DDR2/mDDR memory controller can momentarily raise the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR_OLD_COUNT bit field in the peripheral bus burst priority register (PBBPR) sets the number of the transfers that must be made before the DDR2/mDDR memory controller will raise the priority of the oldest command.</p>
9h	0	0	<p><b>Counts the number of DDR2/mDDR memory controller cycles (DDR_CLK cycles) that a command is pending in the command FIFO. This counter increments every cycle the command FIFO is not empty.</b></p> <p>Use the following to calculate the counter value as a percentage:  <math>\% = \text{counter value} / \text{total DDR\_CLK cycles in sample period}</math></p> <p>As the value of this counter approaches 100%, the number of cycles the DDR2/mDDR memory controller has a command in the command FIFO to service approaches 100%.</p>
Ah-Fh	0	0	Reserved

### 12.4.11 Performance Counter Master Region Select Register (PCMRS)

The performance counter master region select register (PCMRS) is shown in [Figure 12-30](#) and described in [Table 12-34](#).

**Figure 12-30. Performance Counter Master Region Select Register (PCMRS)**

31	24	23	20	19	16
MST_ID2		Reserved		REGION_SEL2	
R/W-0		R-0		R/W-0	
15	8	7	4	3	0
MST_ID1		Reserved		REGION_SEL1	
R/W-0		R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

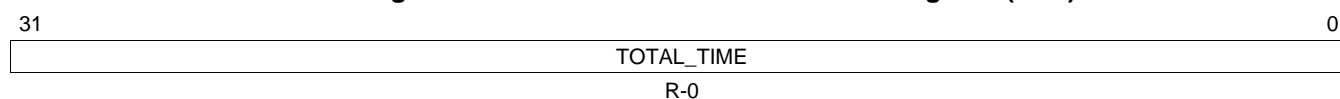
**Table 12-34. Performance Counter Master Region Select Register (PCMRS) Field Descriptions**

Bit	Field	Value	Description
31-24	MST_ID2	0-FFh	Master ID for performance counter 2 register (PC2). For the Master ID value for master peripherals in the device, see the <i>System Configuration (SYSCFG) Module</i> chapter.
23-20	Reserved	0	Any writes to these bit(s) must always have a value of 0.
19-16	REGION_SEL2	0-Fh	Region select for performance counter 2 register (PC2).
		0	PC2 counts total DDR2/mDDR accesses.
		1h-6h	Reserved
		7h	PC2 counts total DDR2/mDDR memory controller memory-mapped register accesses.
		8h-Fh	Reserved
15-8	MST_ID1	0-FFh	Master ID for performance counter 1 register (PC1). For the Master ID value for master peripherals in the device, see the <i>System Configuration (SYSCFG) Module</i> chapter.
7-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.
3-0	REGION_SEL1	0-Fh	Region select for performance counter 1 register (PC1).
		0	PC1 counts total DDR2/mDDR accesses.
		1h-6h	Reserved
		7h	PC1 counts total DDR2/mDDR memory controller memory-mapped register accesses.
		8h-Fh	Reserved

## Performance Counter Time Register (PCT)

The performance counter time register (PCT) is shown in [Figure 12-31](#) and described in [Table 12-35](#).

**Figure 12-31. Performance Counter Time Register (PCT)**



LEGEND: R = Read only; -n = value after reset

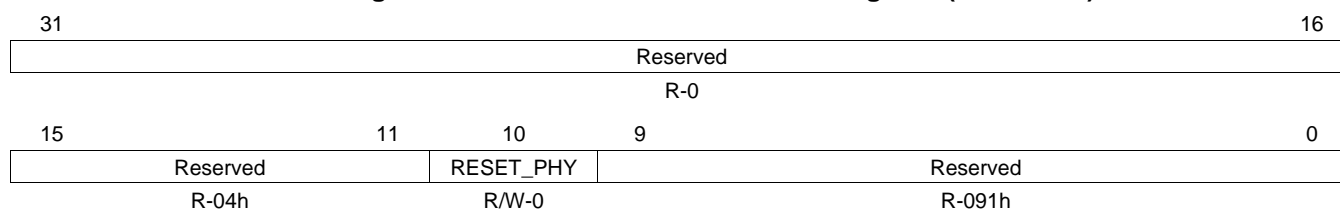
**Table 12-35. Performance Counter Time Register (PCT) Field Description**

Bit	Field	Value	Description
31-0	TOTAL_TIME	0-FFFF FFFFh	32-bit counter that continuously counts number for DDR_CLK cycles elapsed after the DDR2/mDDR memory controller is brought out of reset.

### 12.4.12 DDR PHY Reset Control Register (DRPYRCR)

The DDR PHY reset control register (DRPYRCR) is used to reset the DDR PHY. The DRPYRCR is shown in [Figure 12-32](#) and described in [Table 12-36](#).

**Figure 12-32. DDR PHY Reset Control Register (DRPYRCR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

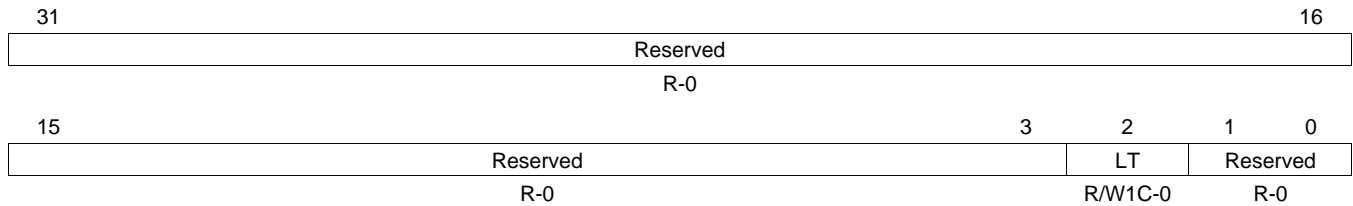
**Table 12-36. DDR PHY Reset Control Register (DRPYRCR)**

Bit	Field	Value	Description
31-11	Reserved	0000 04h	Always write the default value to these bits.
10	RESET_PHY	0 1	Reset DDR PHY. No effect. Resets DDR PHY.
9-0	Reserved	091h	Always write the default value to these bits.

### 12.4.13 Interrupt Raw Register (IRR)

The interrupt raw register (IRR) displays the raw status of the interrupt. If the interrupt condition occurs, the corresponding bit in IRR is set independent of whether or not the interrupt is enabled. The IRR is shown in [Figure 12-33](#) and described in [Table 12-37](#).

**Figure 12-33. Interrupt Raw Register (IRR)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

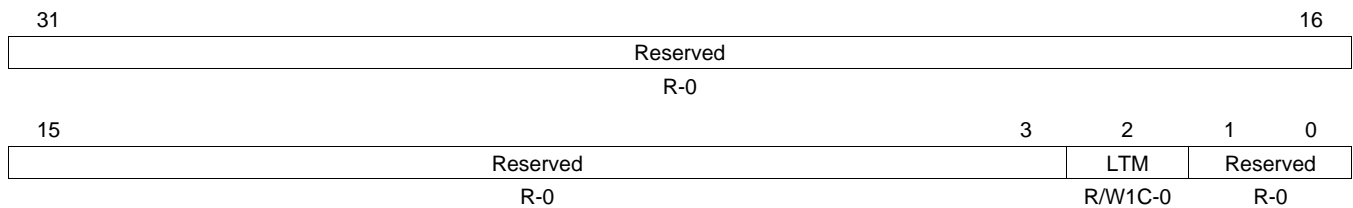
**Table 12-37. Interrupt Raw Register (IRR) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LT	0	Line trap. Write a 1 to clear LT and the LTM bit in the interrupt masked register (IMR); a write of 0 has no effect.
		0	A line trap condition has not occurred.
		1	Illegal memory access type. See <a href="#">Section 12.2.14</a> for more details.
1-0	Reserved	0	Reserved

### 12.4.14 Interrupt Masked Register (IMR)

The interrupt masked register (IMR) displays the status of the interrupt when it is enabled. If the interrupt condition occurs and the corresponding bit in the interrupt mask set register (IMSR) is set, then the IMR bit is set. The IMR bit is not set if the interrupt is not enabled in IMSR. The IMR is shown in [Figure 12-34](#) and described in [Table 12-38](#).

**Figure 12-34. Interrupt Masked Register (IMR)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 12-38. Interrupt Masked Register (IMR) Field Descriptions**

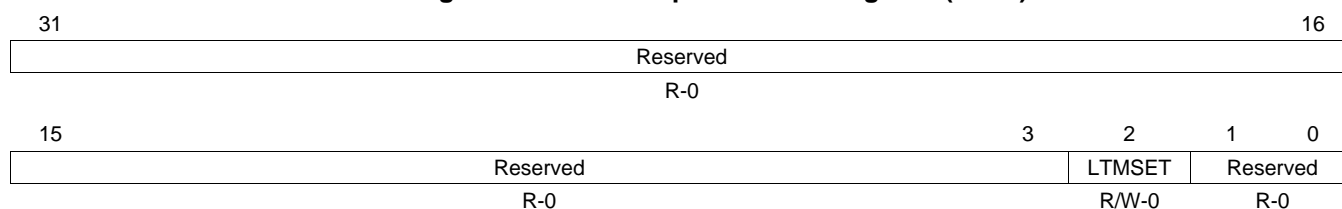
Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LTM	0	Line trap masked. Write a 1 to clear LTM and the LT bit in the interrupt raw register (IRR); a write of 0 has no effect.
		0	A line trap condition has not occurred.
		1	Illegal memory access type (only set if the LTMSET bit in IMSR is set). See <a href="#">Section 12.2.14</a> for more details.
1-0	Reserved	0	Reserved

### 12.4.15 Interrupt Mask Set Register (IMSR)

The interrupt mask set register (IMSR) enables the DDR2/mDDR memory controller interrupt. The IMSR is shown in Figure 12-35 and described in Table 12-39.

**NOTE:** If the LTMSET bit in IMSR is set concurrently with the LTMCLR bit in the interrupt mask clear register (IMCR), the interrupt is not enabled and neither bit is set to 1.

**Figure 12-35. Interrupt Mask Set Register (IMSR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-39. Interrupt Mask Set Register (IMSR) Field Descriptions**

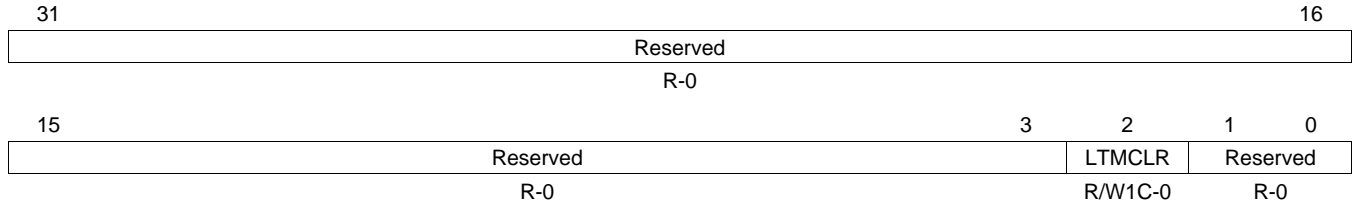
Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LTMSET	0	Line trap interrupt set. Write a 1 to set LTMSET and the LTMCLR bit in the interrupt mask clear register (IMCR); a write of 0 has no effect.
		0	Line trap interrupt is not enabled; a write of 1 to the LTMCLR bit in IMCR occurred.
		1	Line trap interrupt is enabled.
1-0	Reserved	0	Reserved

### 12.4.16 Interrupt Mask Clear Register (IMCR)

The interrupt mask clear register (IMCR) disables the DDR2/mDDR memory controller interrupt. Once an interrupt is enabled, it may be disabled by writing a 1 to the IMCR bit. The IMCR is shown in [Figure 12-36](#) and described in [Table 12-40](#).

**NOTE:** If the LTMCLR bit in IMCR is set concurrently with the LTMSET bit in the interrupt mask set register (IMSR), the interrupt is not enabled and neither bit is set to 1.

**Figure 12-36. Interrupt Mask Clear Register (IMCR)**



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 12-40. Interrupt Mask Clear Register (IMCR) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LTMCLR	0	Line trap interrupt clear. Write a 1 to clear LTMCLR and the LTMSET bit in the interrupt mask set register (IMSR); a write of 0 has no effect.
		1	Line trap interrupt is not enabled.
		1	Line trap interrupt is enabled; a write of 1 to the LTMSET bit in IMSR occurred.
1-0	Reserved	0	Reserved

### 12.4.17 DDR PHY Control Register (DRPYC1R)

The DDR PHY control register 1 (DRPYC1R) configures the DDR2/mDDR memory controller read latency. The DRPYC1R is shown in [Figure 12-37](#) and described in [Table 12-41](#).

**Figure 12-37. DDR PHY Control Register 1 (DRPYC1R)**

Reserved														
R-0														
31														16
15	14	12	11	8	7	6	5	3	2	0				
Rsvd	CONFIG_DLL_MODE	Reserved			EXT_STRBEN	PWRDNEN	Reserved		RL					
R-0	R/W-0	R-0			R/W-0	R/W-1	R-0		R/W-6h					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 12-41. DDR PHY Control Register 1 (DRPYC1R) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-12	CONFIG_DLL_MODE	0-1h 2h 3h-7h	DLL configuration. Controls the value assigned to the config_dll_mode input. DLL REFCLK is enabled. DLL REFCLK is disabled. Reserved
11-8	Reserved	0	Reserved
7	EXT_STRBEN	0 1	Internal/External strobe gating. Internal strobe gating mode. External strobe gating mode.
6	PWRDNEN	0 1	Power down receivers. Receivers powered up when idle. Receivers powered down when idle.
5-3	Reserved	0	Reserved
2-0	RL	0-7h	Read latency. Read latency is equal to CAS latency plus round trip board delay for data minus 1. The maximum value of read latency that is supported is CAS latency plus 2. The minimum read latency value that is supported is CAS latency plus 1. The read latency value is defined in number of MCLK/DDR_CLK cycles.

## Enhanced Capture (eCAP) Module

---

---

The enhanced capture (eCAP) module is essential in systems where accurate timing of external events is important. This chapter describes the eCAP module.

Topic	Page
<b>13.1 Introduction</b> .....	<b>294</b>
<b>13.2 Architecture</b> .....	<b>295</b>
<b>13.3 Applications</b> .....	<b>304</b>
<b>13.4 Registers</b> .....	<b>320</b>



## 13.1 Introduction

### 13.1.1 Purpose of the Peripheral

Uses for eCAP include:

- Sample rate measurements of audio inputs
- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

### 13.1.2 Features

The eCAP module includes the following features:

- 32-bit time base counter
- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single shot capture of up to four event time-stamps
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

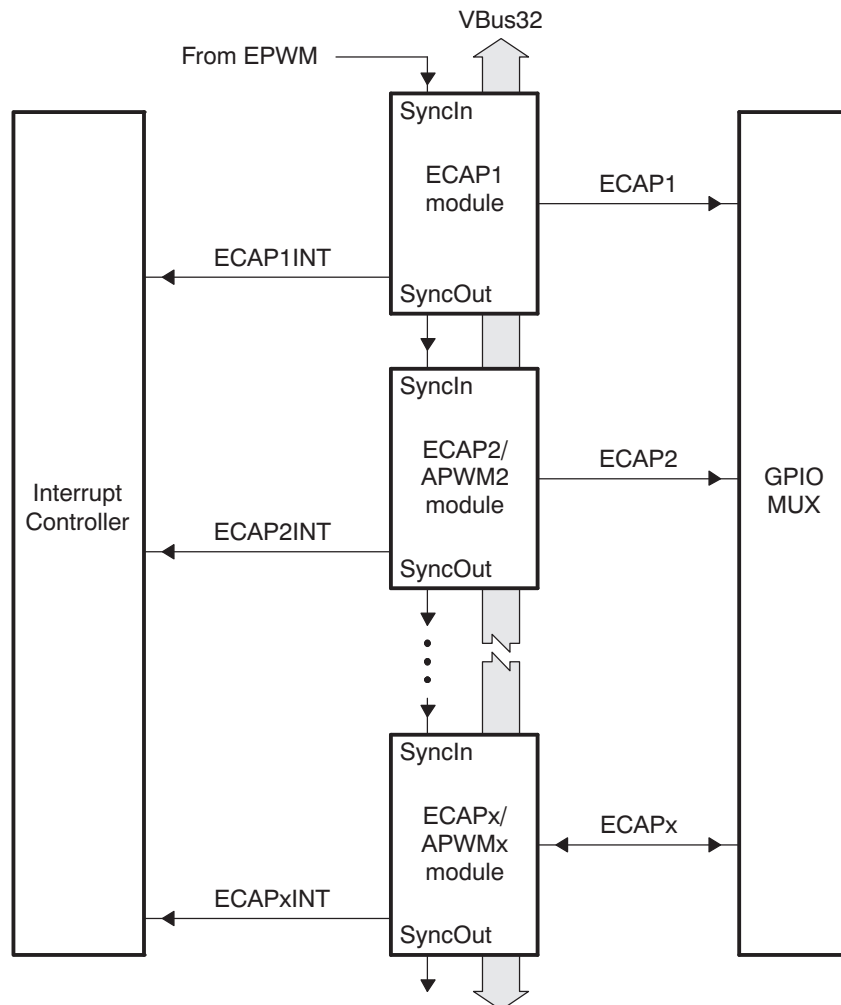
## 13.2 Architecture

The eCAP module represents one complete capture channel that can be instantiated multiple times depending on the target device. In the context of this guide, one eCAP channel has the following independent key resources:

- Dedicated input capture pin
- 32-bit time base counter
- 4 × 32-bit time-stamp capture registers (CAP1-CAP4)
- 4-stage sequencer (Modulo4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Independent edge polarity (rising/falling edge) selection for all 4 events
- Input capture signal prescaling (from 2-62)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Control for continuous time-stamp captures using a 4-deep circular buffer (CAP1-CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

Multiple identical eCAP modules can be contained in a system as shown in [Figure 13-1](#). The number of modules is device-dependent and is based on target application needs. In this chapter, the letter x within a signal or module name is used to indicate a generic eCAP instance on a device.

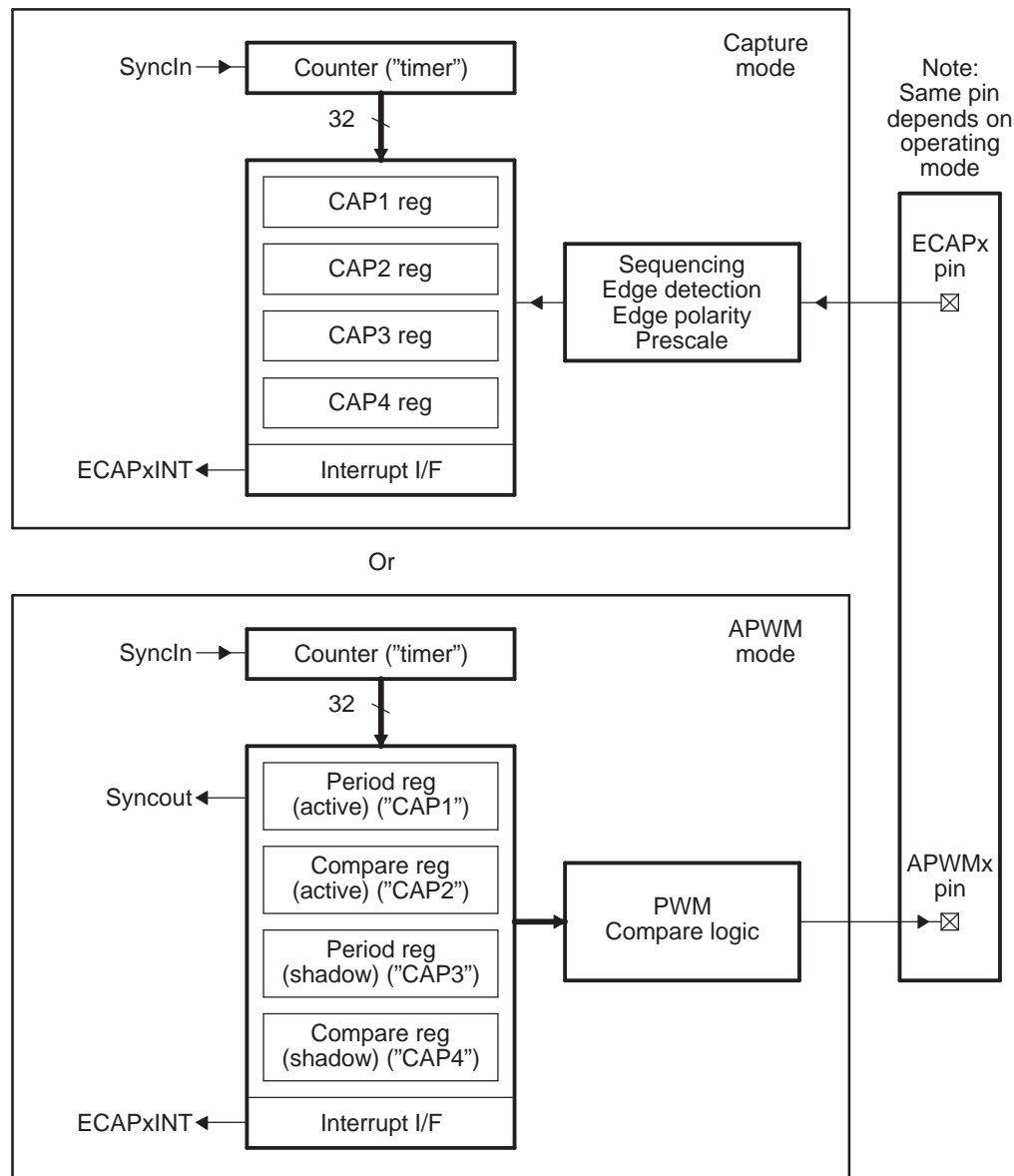
**Figure 13-1. Multiple eCAP Modules**



### 13.2.1 Capture and APWM Operating Mode

You can use the eCAP module resources to implement a single-channel PWM generator (with 32 bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. Figure 13-2 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

**Figure 13-2. Capture and APWM Modes of Operation**

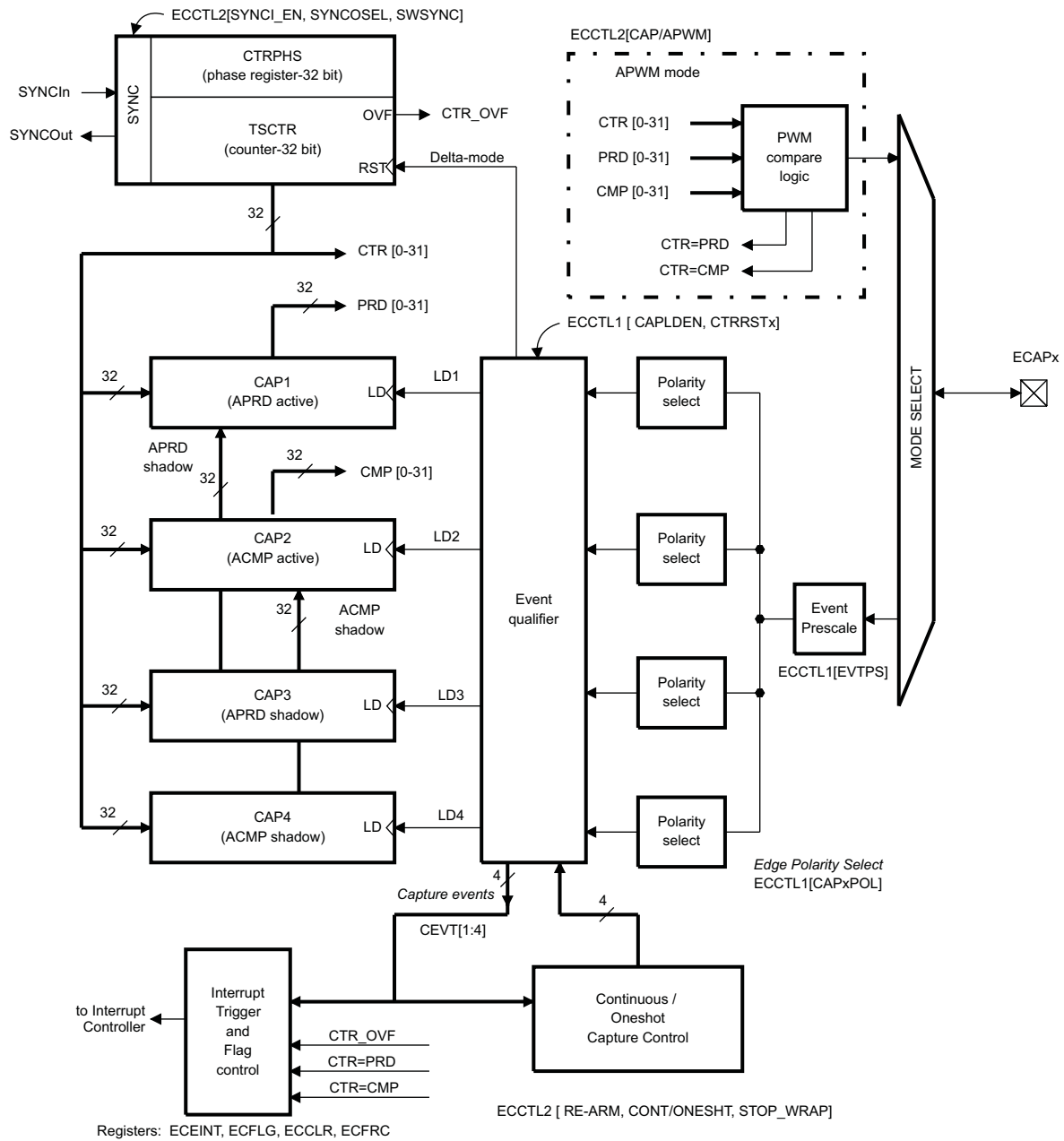


- (1) A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.
- (2) In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

### 13.2.2 Capture Mode Description

Figure 13-3 shows the various components that implement the capture function.

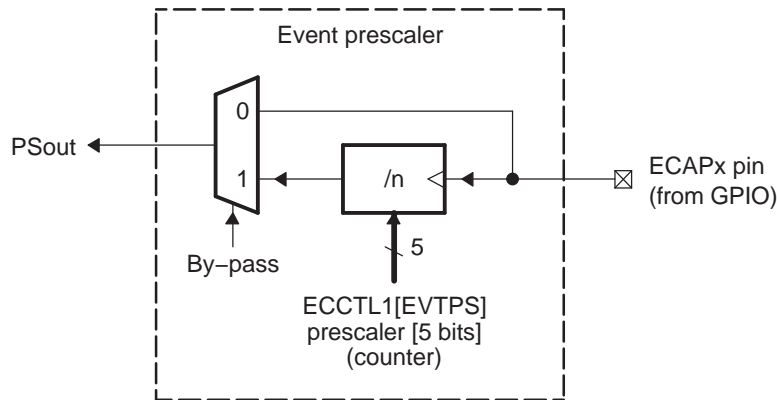
Figure 13-3. Capture Function Diagram



### 13.2.2.1 Event Prescaler

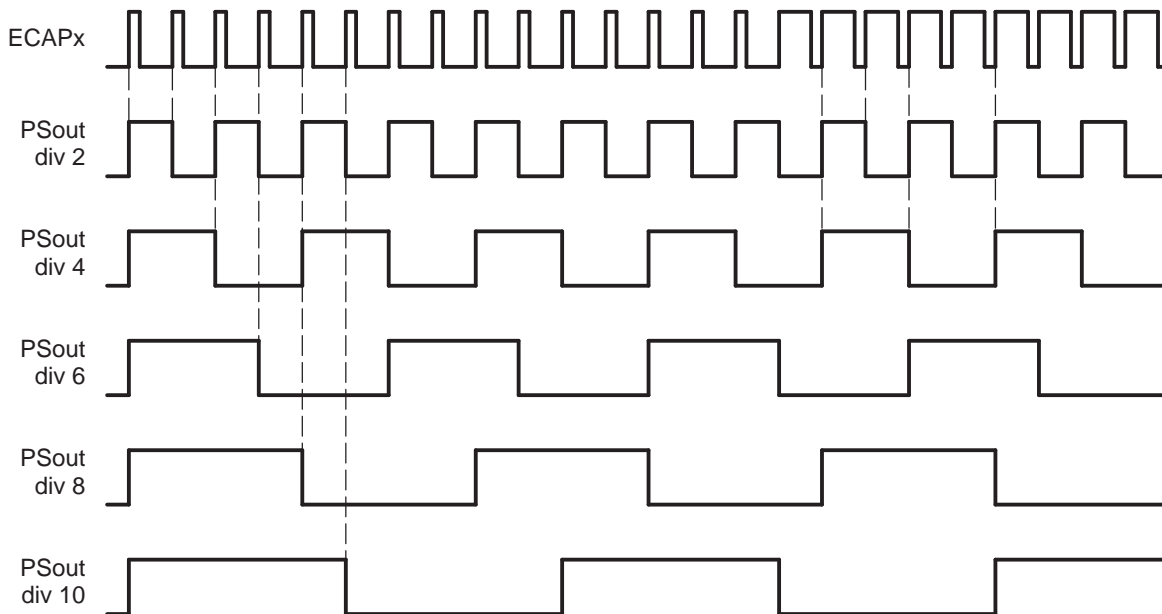
An input capture signal (pulse train) can be prescaled by  $N = 2-62$  (in multiples of 2) or can bypass the prescaler. This is useful when very high frequency signals are used as inputs. Figure 13-4 shows a functional diagram and Figure 13-5 shows the operation of the prescale function.

Figure 13-4. Event Prescale Control



- (1) When a prescale value of 1 is chosen (ECCTL1[13:9] = 0000) the input capture signal by-passes the prescale logic completely.

Figure 13-5. Prescale Function Waveforms



### 13.2.2.2 Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection multiplexers are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to its respective CAP $n$  register by the Mod4 counter. The CAP $n$  register is loaded on the falling edge.

### 13.2.2.3 Continuous/One-Shot Control

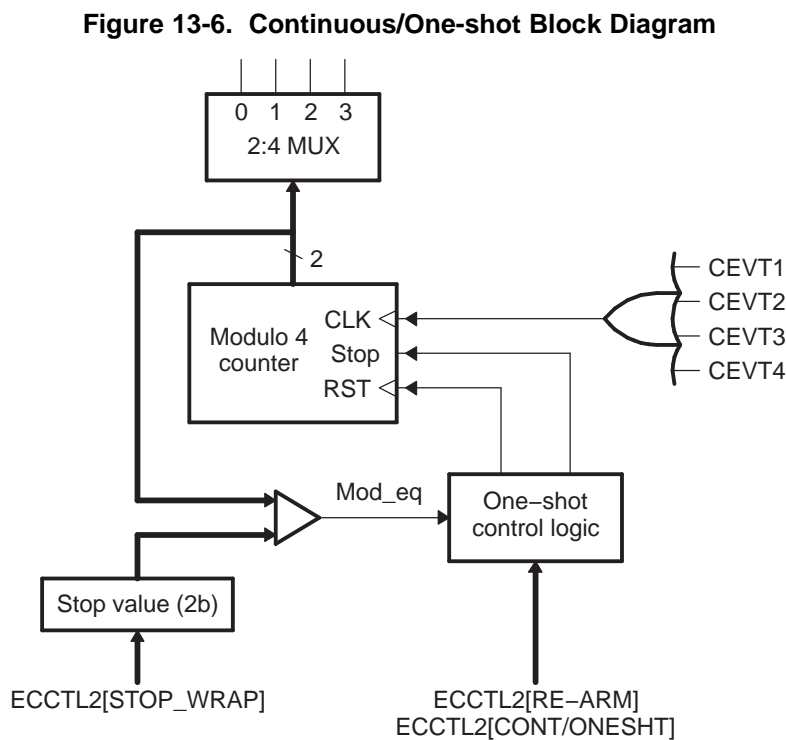
- The Mod4 (2 bit) counter is incremented via edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output, and when equal stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. This occurs during one-shot operation.

The continuous/one-shot block (Figure 13-6) controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the eCAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of CAP1-4 registers (time-stamps).

Re-arming prepares the eCAP module for another capture sequence. Also re-arming clears (to zero) the Mod4 counter and permits loading of CAP1-4 registers again, providing the CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0, the one-shot action is ignored, and capture values continue to be written to CAP1-4 in a circular buffer sequence.



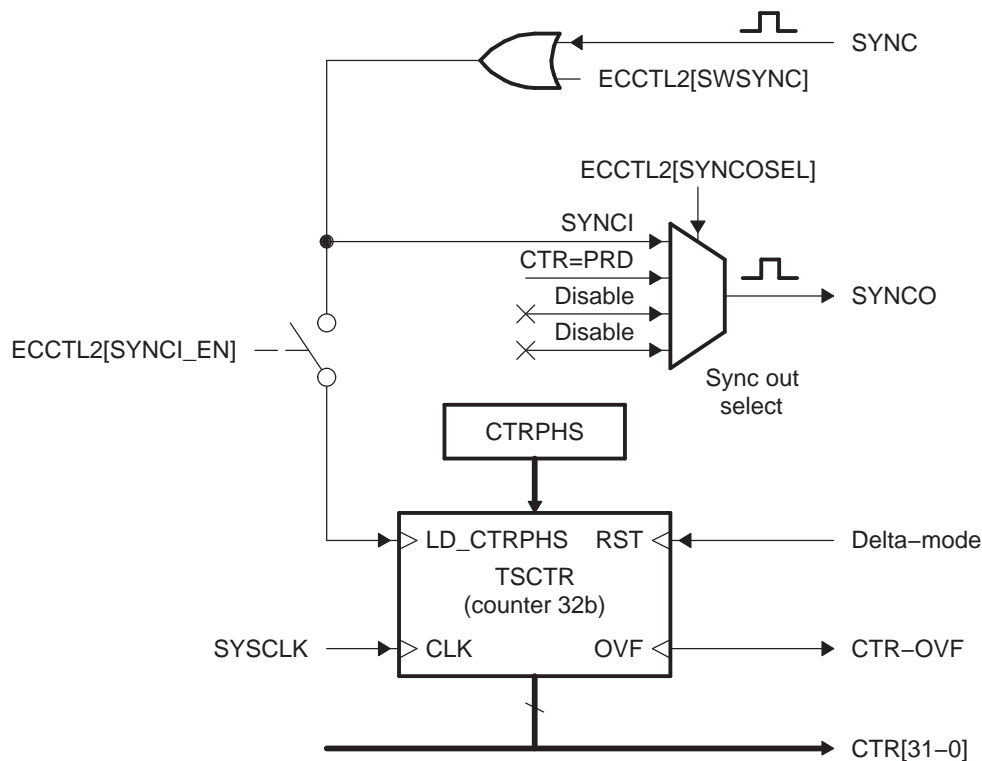
### 13.2.2.4 32-Bit Counter and Phase Control

This counter (Figure 13-7) provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1-LD4 signals.

Figure 13-7. Counter and Synchronization Block Diagram



### 13.2.2.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via control bit CAPLDEN. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACMP) for CAP1 and CAP2 during APWM operation.

### 13.2.2.6 Interrupt Control

An Interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP). See [Figure 13-8](#).

A counter overflow event (FFFF FFFFh->0000 0000h) is also provided as an interrupt source (CTROVF).

The capture events are edge and sequencer qualified (that is, ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the eCAP $n$  module) going to the interrupt controller.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR = PRD, CTR = CMP) can be generated. The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the interrupt controller only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECCLR) before any other interrupt pulses are generated. You can force an interrupt event via the interrupt force register (ECFRC). This is useful for test purposes.

### 13.2.2.7 Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of CAP1 or CAP2 from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, CTR[31:0] = PRD[31:0]

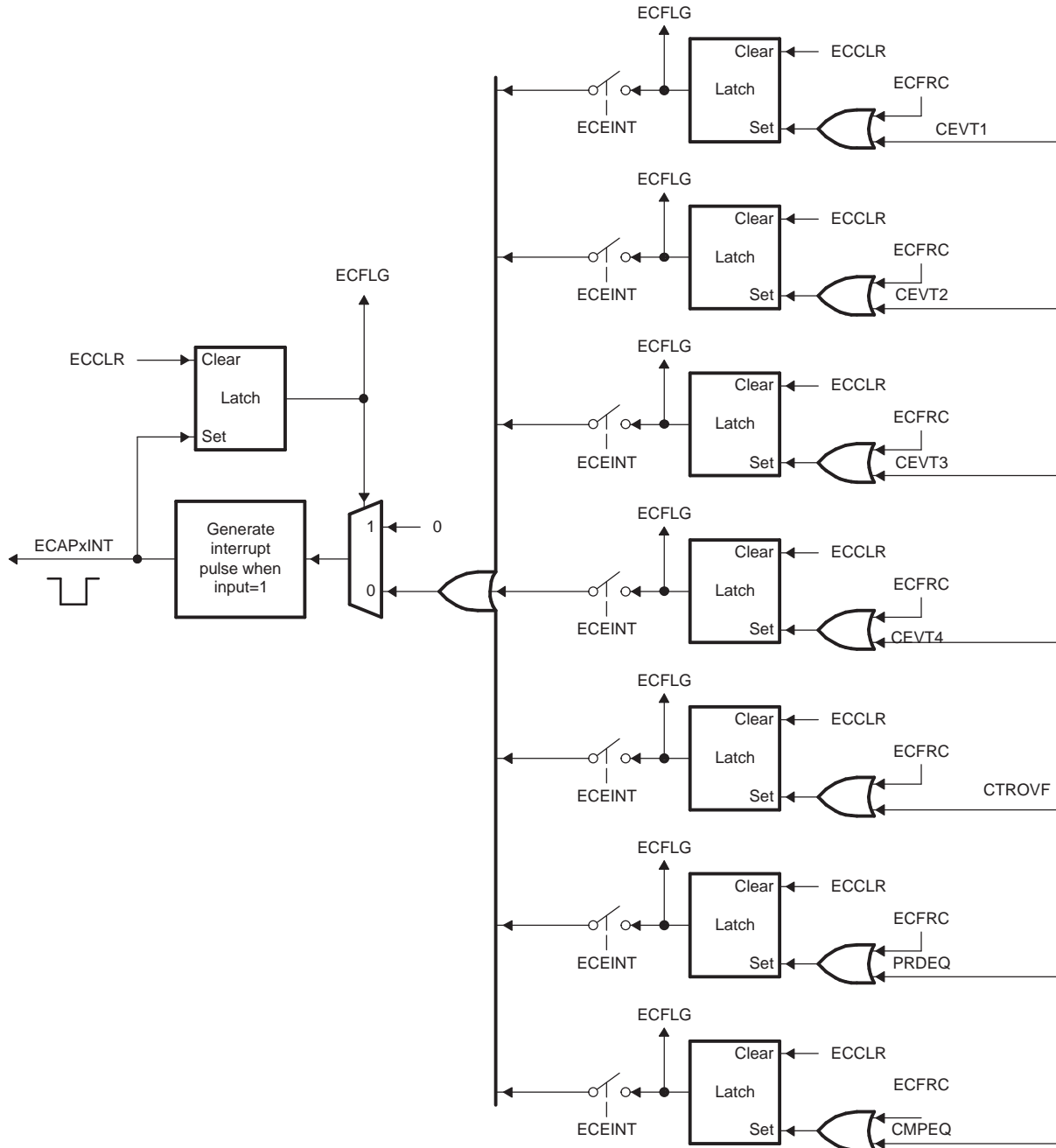
---

**NOTE:** The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR = PRD, CTR = CMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

---



Figure 13-8. Interrupts in eCAP Module

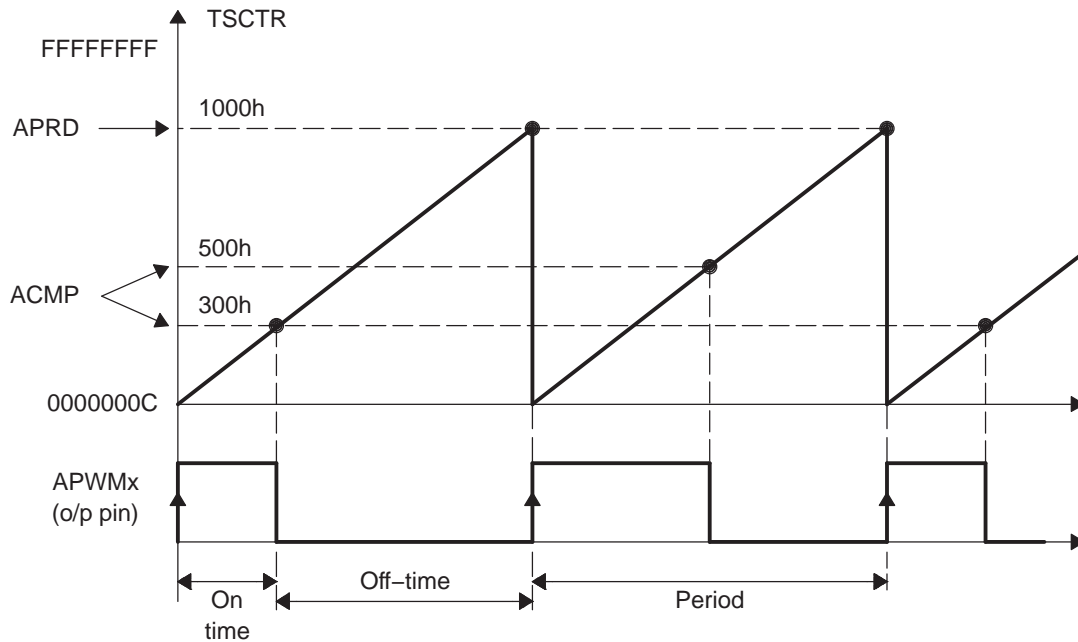


### 13.2.2.8 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via shadow registers APRD and ACMP (CAP3/4). The shadow register contents are transferred over to CAP1/2 registers either immediately upon a write, or on a CTR = PRD trigger.
- In APWM mode, writing to CAP1/CAP2 active registers will also write the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 will invoke the shadow mode.
- During initialization, you must write to the active registers for both period and compare. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, you only need to use the shadow registers.

**Figure 13-9. PWM Waveform Details Of APWM Mode Operation**



The behavior of APWM active-high mode (APWMPOL == 0) is:

- CMP = 0x00000000, output low for duration of period (0% duty)
- CMP = 0x00000001, output high 1 cycle
- CMP = 0x00000002, output high 2 cycles
- CMP = PERIOD, output high except for 1 cycle (<100% duty)
- CMP = PERIOD+1, output high for complete period (100% duty)
- CMP > PERIOD+1, output high for complete period

The behavior of APWM active-low mode (APWMPOL == 1) is:

- CMP = 0x00000000, output high for duration of period (0% duty)
- CMP = 0x00000001, output low 1 cycle
- CMP = 0x00000002, output low 2 cycles
- CMP = PERIOD, output low except for 1 cycle (<100% duty)
- CMP = PERIOD+1, output low for complete period (100% duty)

CMP > PERIOD+1, output low for complete period

### 13.3 Applications

The following sections will provide Applications examples and code snippets to show how to configure and operate the eCAP module. For clarity and ease of use, below are useful #defines which will help in the understanding of the examples.

```

// ECCTL1 ( ECAP Control Reg 1)
//=====
// CAPxPOL bits
#define    EC_RISING           0x0
#define    EC_FALLING         0x1

// CTRRSTx bits
#define    EC_ABS_MODE        0x0
#define    EC_DELTA_MODE     0x1

// PRESCALE bits
#define    EC_BYPASS          0x0
#define    EC_DIV1            0x0
#define    EC_DIV2            0x1
#define    EC_DIV4            0x2
#define    EC_DIV6            0x3
#define    EC_DIV8            0x4
#define    EC_DIV10           0x5

// ECCTL2 ( ECAP Control Reg 2)
//=====
// CONT/ONESHOT bit
#define    EC_CONTINUOUS      0x0
#define    EC_ONESHOT        0x1

// STOPVALUE bit
#define    EC_EVENT1          0x0
#define    EC_EVENT2          0x1
#define    EC_EVENT3          0x2
#define    EC_EVENT4          0x3

// RE-ARM bit
#define    EC_ARM              0x1

// TSCTRSTOP bit
#define    EC_FREEZE          0x0
#define    EC_RUN              0x1

// SYNCO_SEL bit
#define    EC_SYNCIN          0x0
#define    EC_CTR_PRD         0x1
#define    EC_SYNCO_DIS       0x2

// CAP/APWM mode bit
#define    EC_CAP_MODE        0x0
#define    EC_APWM_MODE       0x1

// APWMPOL bit
#define    EC_ACTV_HI         0x0
#define    EC_ACTV_LO         0x1

// Generic
#define    EC_DISABLE         0x0
#define    EC_ENABLE          0x1
#define    EC_FORCE           0x1

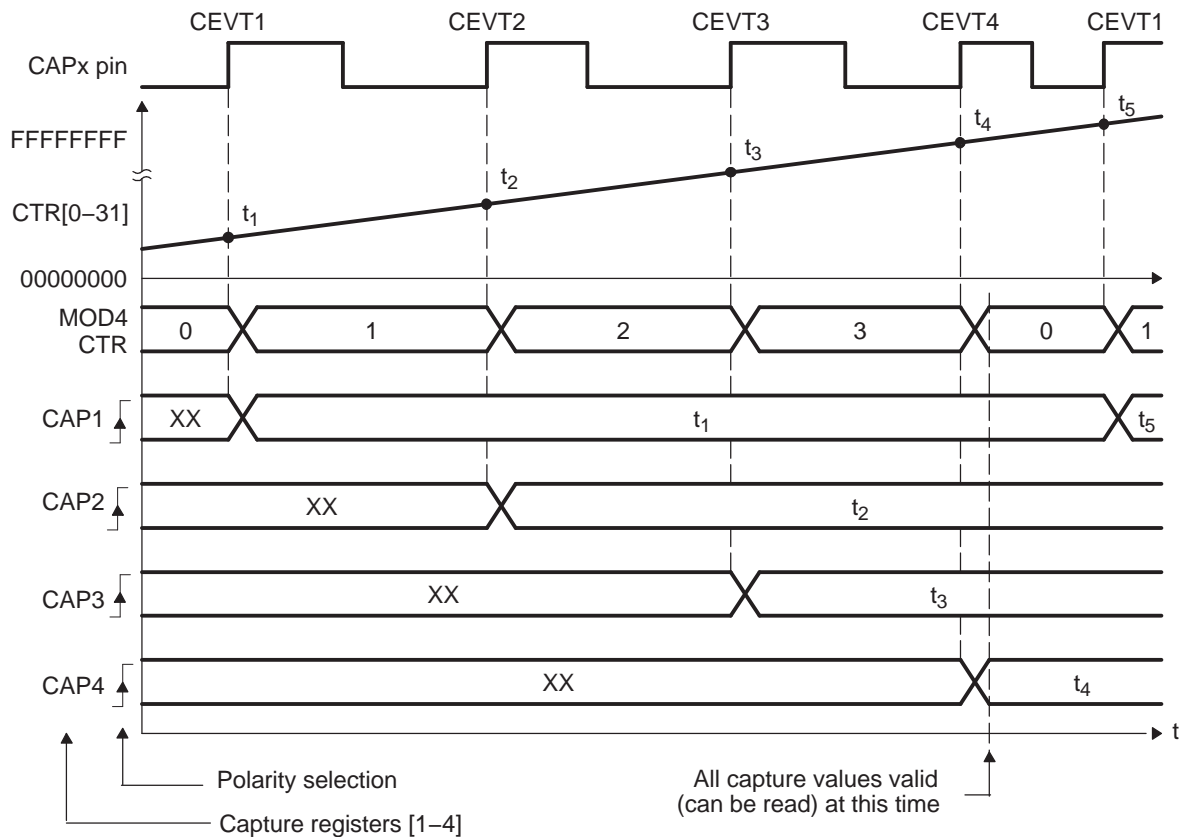
```

### 13.3.1 Absolute Time-Stamp Operation Rising Edge Trigger Example

Figure 13-10 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFF FFFFh (maximum value), it wraps around to 0000 0000h (not shown in Figure 13-10), if this occurs, the CTROVF (counter overflow) flag is set, and an interrupt (if enabled) occurs, CTROVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. Captured time-stamps are valid at the point indicated by the diagram, after the 4th event, hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAP $n$  registers.

Figure 13-10. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect



**Table 13-1. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

**Example 13-1. Code Snippet for CAP Mode Absolute Time, Rising Edge Trigger**

```

// Code snippet for CAP mode Absolute Time, Rising edge trigger

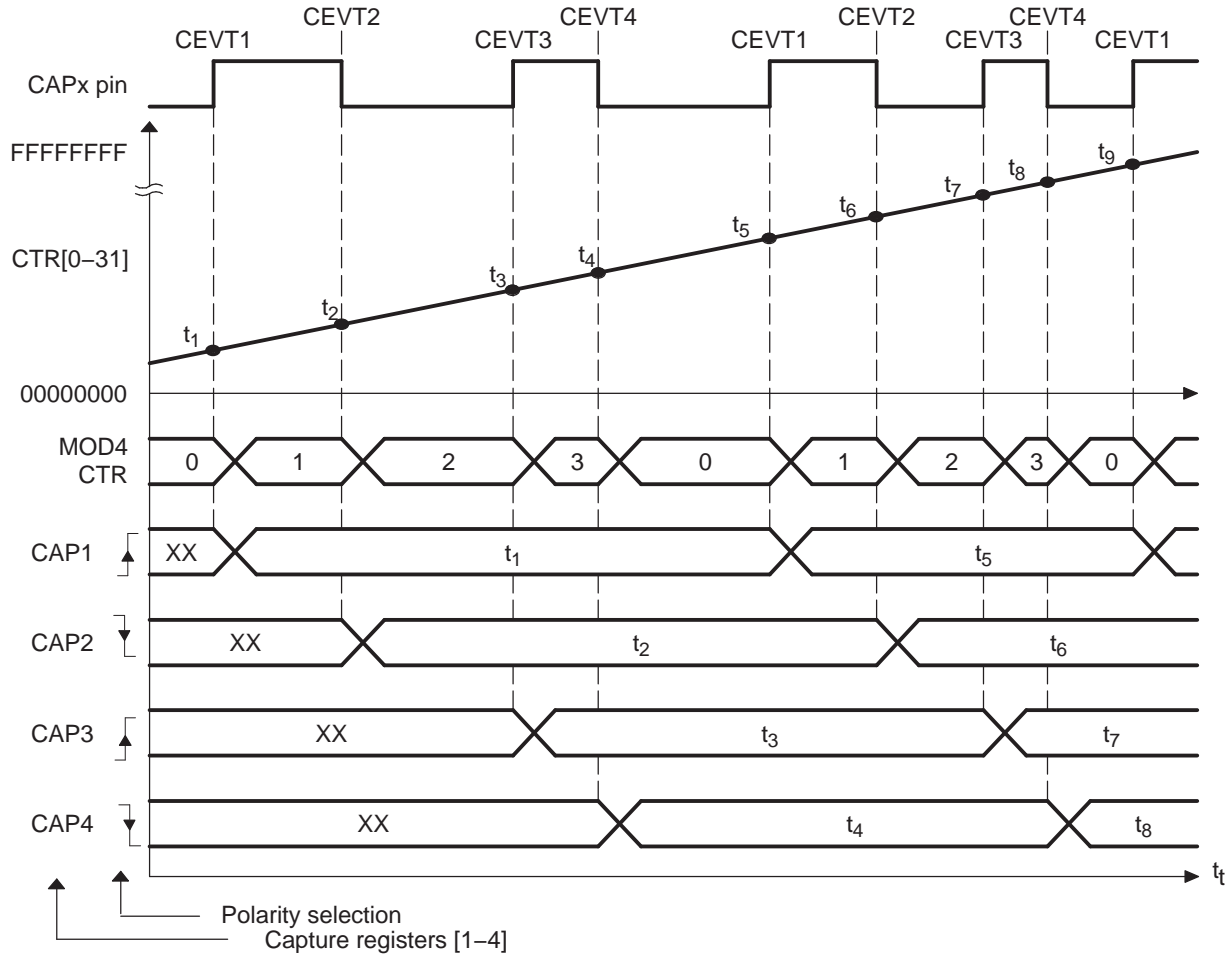
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECAPxRegs.CAP1;      // Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;      // Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;      // Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;      // Fetch Time-Stamp captured at t4

Period1 = TSt2-TSt1;        // Calculate 1st period
Period2 = TSt3-TSt2;        // Calculate 2nd period
Period3 = TSt4-TSt3;        // Calculate 3rd period
    
```

### 13.3.2 Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example

In Figure 13-11 the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information:  $\text{Period1} = t_3 - t_1$ ,  $\text{Period2} = t_5 - t_3$ , ...etc.  $\text{Duty Cycle1 (on-time \%)} = (t_2 - t_1) / \text{Period1} \times 100\%$ , etc.  $\text{Duty Cycle1 (off-time \%)} = (t_3 - t_2) / \text{Period1} \times 100\%$ , etc.

Figure 13-11. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect



**Table 13-2. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCL_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

**Example 13-2. Code Snippet for CAP Mode Absolute Time, Rising and Falling Edge Trigger**

```

// Code snippet for CAP mode Absolute Time, Rising & Falling edge triggers

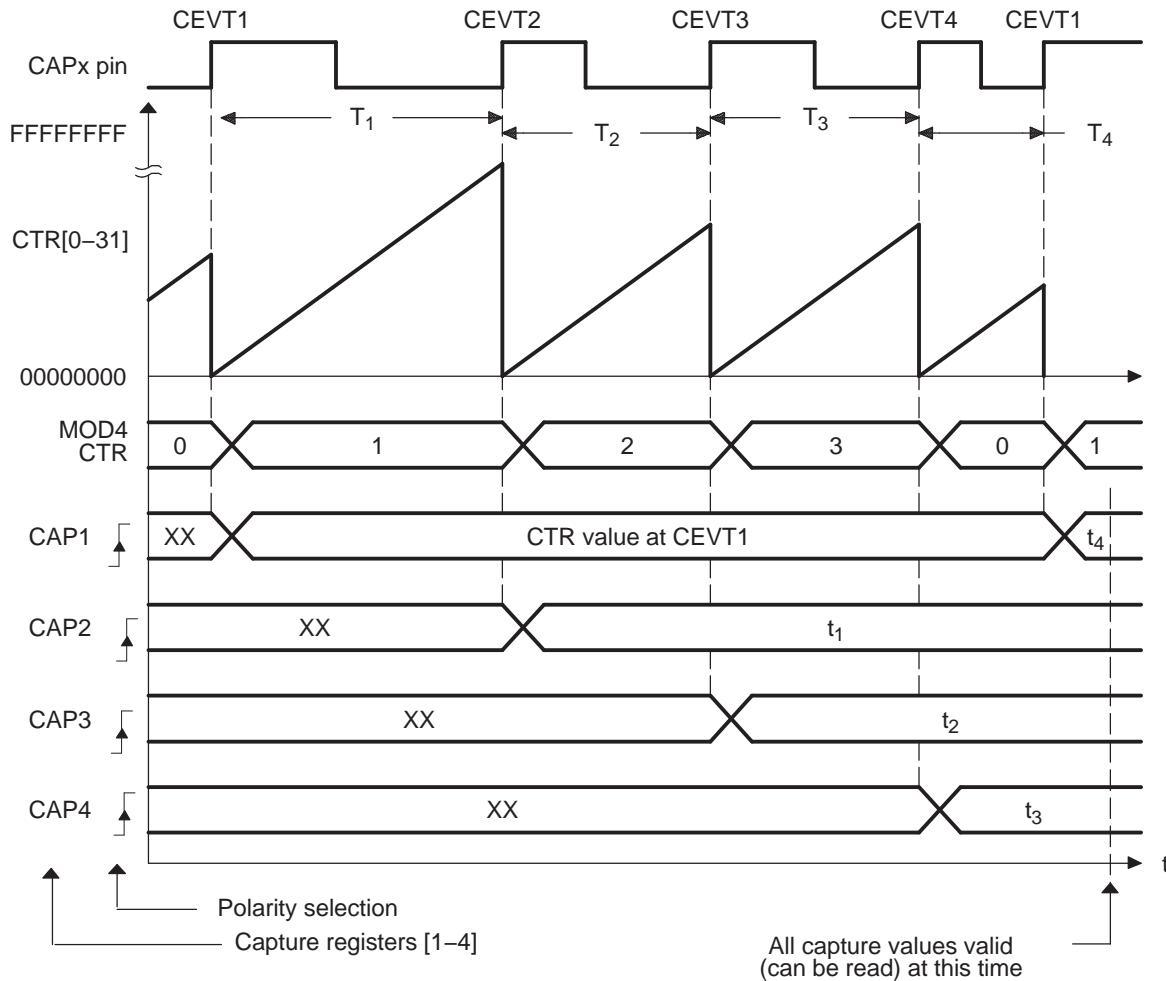
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECAPxRegs.CAP1;      // Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;      // Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;      // Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;      // Fetch Time-Stamp captured at t4

Period1 = TSt3-TSt1;        // Calculate 1st period
DutyOnTime1 = TSt2-TSt1;    // Calculate On time
DutyOffTime1 = TSt3-TSt2;   // Calculate Off time
    
```

### 13.3.3 Time Difference (Delta) Operation Rising Edge Trigger Example

Figure 13-12 shows how the eCAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (time-stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFF FFFFh (maximum value), before the next event, it wraps around to 0000 0000h and continues, a CANTOVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAPn contents directly give timing data without the need for CPU calculations:  $Period1 = T_1$ ,  $Period2 = T_2, \dots$  etc. As shown in Figure 13-12, the CEVT1 event is a good trigger point to read the timing data,  $T_1, T_2, T_3, T_4$  are all valid here.

Figure 13-12. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect





**Table 13-3. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSTRSTOP	EC_RUN

**Example 13-3. Code Snippet for CAP Mode Delta Time, Rising Edge Trigger**

```

// Code snippet for CAP mode Delta Time, Rising edge trigger

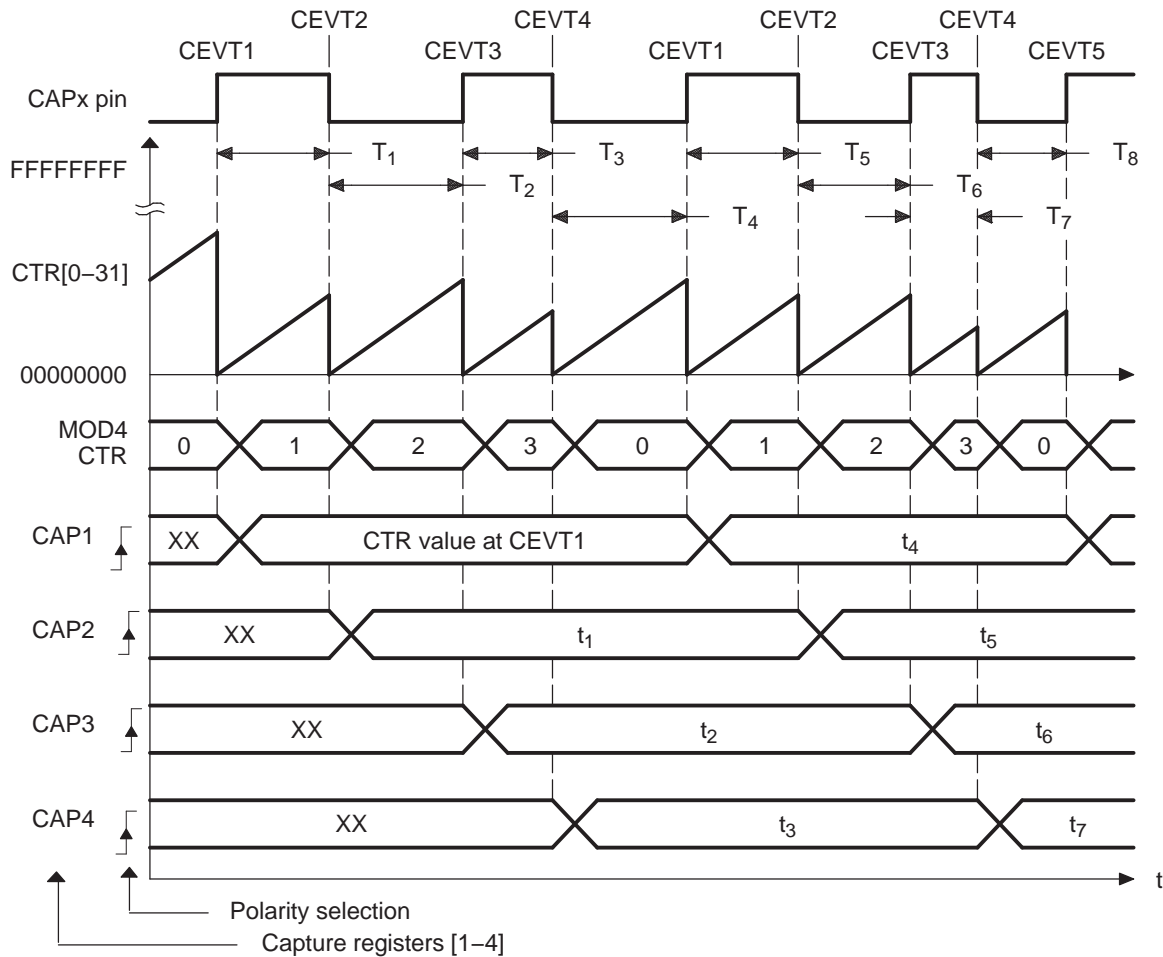
// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Period value.
Period4 = ECAPxRegs.CAP1;    // Fetch Time-Stamp captured at T1
Period1 = ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at T2
Period2 = ECAPxRegs.CAP3;    // Fetch Time-Stamp captured at T3
Period3 = ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at T4
    
```

### 13.3.4 Time Difference (Delta) Operation Rising and Falling Edge Trigger Example

In Figure 13-13 the eCAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information:  $\text{Period1} = T_1 + T_2$ ,  $\text{Period2} = T_3 + T_4$ , ...etc  $\text{Duty Cycle1 (on-time \%)} = T_1 / \text{Period1} \times 100\%$ , etc  $\text{Duty Cycle1 (off-time \%)} = T_2 / \text{Period1} \times 100\%$ , etc

During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, that is, during run-time, only the shadow registers must be used.

Figure 13-13. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect



**Table 13-4. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers**

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCl_EN	EC_DISABLE
ECCTL2	TSTRSTOP	EC_RUN

**Example 13-4. Code Snippet for CAP Mode Delta Time, Rising and Falling Edge Triggers**

```
// Code snippet for CAP mode Delta Time, Rising and Falling edge triggers

// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Duty cycle values.
DutyOnTime1 = ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at T2
DutyOffTime1 = ECAPxRegs.CAP3;   // Fetch Time-Stamp captured at T3
DutyOnTime2 = ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at T4
DutyOffTime2 = ECAPxRegs.CAP1;   // Fetch Time-Stamp captured at T1

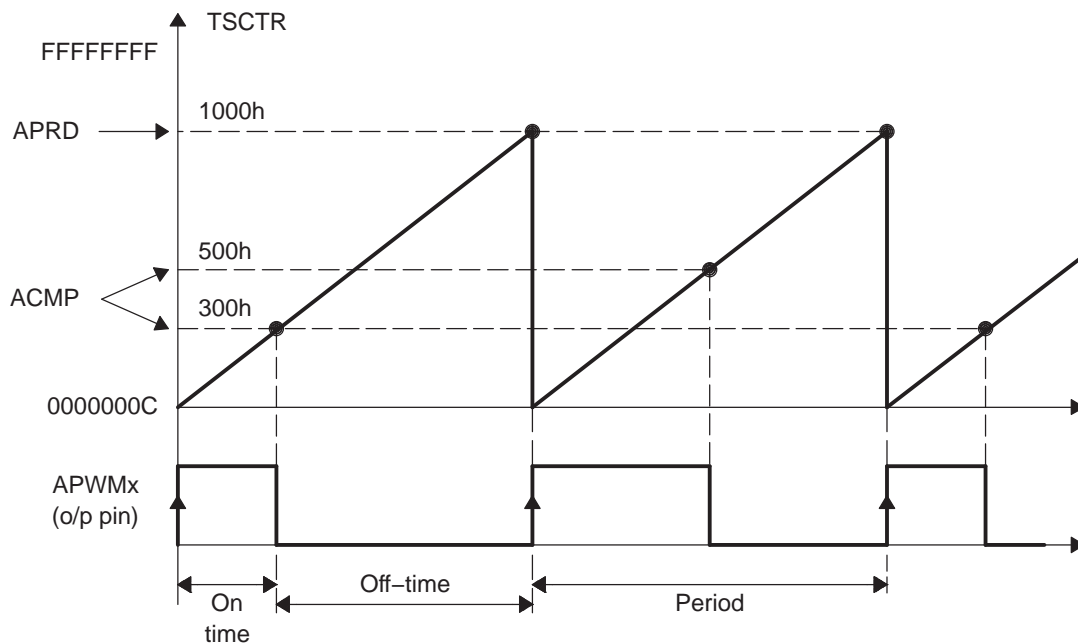
Period1 = DutyOnTime1 + DutyOffTime1;
Period2 = DutyOnTime2 + DutyOffTime2;
```

### 13.3.5 Application of the APWM Mode

#### 13.3.5.1 Simple PWM Generation (Independent Channel/s) Example

In this example, the eCAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWM $n$ . The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time.

Figure 13-14. PWM Waveform Details of APWM Mode Operation



**Table 13-5. ECAP Initialization for APWM Mode**

Register	Bit	Value
CAP1	CAP1	0x1000
CTRPHS	CTRPHS	0x0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSTRSTOP	EC_RUN

**Example 13-5. Code Snippet for APWM Mode**

```

// Code snippet for APWM mode Example 1

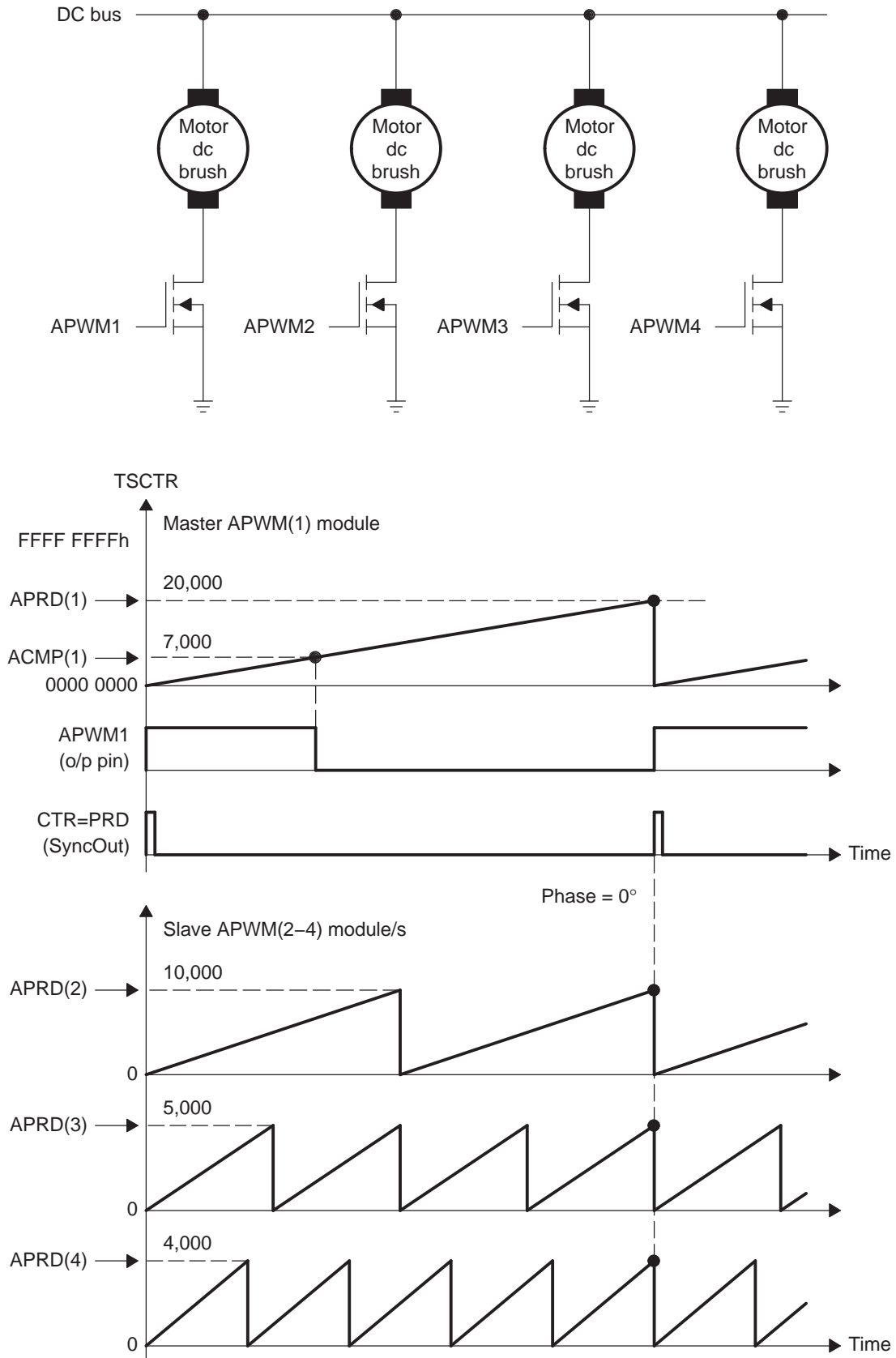
// Run Time (Instant 1, e.g. ISR call)
//=====
    ECAPxRegs.CAP2 = 0x300;      // Set Duty cycle i.e. compare value

// Run Time (Instant 2, e.g. another ISR call)
//=====
    ECAPxRegs.CAP2 = 0x500;      // Set Duty cycle i.e. compare value
    
```

**13.3.5.2 Multichannel PWM Generation with Synchronization Example**

Figure 13-15 takes advantage of the synchronization feature between eCAP modules. Here 4 independent PWM channels are required with different frequencies, but at integer multiples of each other to avoid "beat" frequencies. Hence one eCAP module is configured as the Master and the remaining 3 are Slaves all receiving their synch pulse (CTR = PRD) from the master. Note the Master is chosen to have the lower frequency ( $F_1 = 1/20,000$ ) requirement. Here Slave2 Freq =  $2 \times F_1$ , Slave3 Freq =  $4 \times F_1$  and Slave4 Freq =  $5 \times F_1$ . Note here values are in decimal notation. Also, only the APWM1 output waveform is shown.

Figure 13-15. Multichannel PWM Example Using 4 eCAP Modules



**Table 13-6. ECAP1 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	20000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

**Table 13-7. ECAP2 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	10000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCL
ECCTL2	TSCTRSTOP	EC_RUN

**Table 13-8. ECAP3 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	5000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCL
ECCTL2	TSCTRSTOP	EC_RUN

**Table 13-9. ECAP4 Initialization for Multichannel PWM Generation with Synchronization**

Register	Bit	Value
CAP1	CAP1	4000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

**Example 13-6. Code Snippet for Multichannel PWM Generation with Synchronization**

```
// Code snippet for APWM mode Example 2

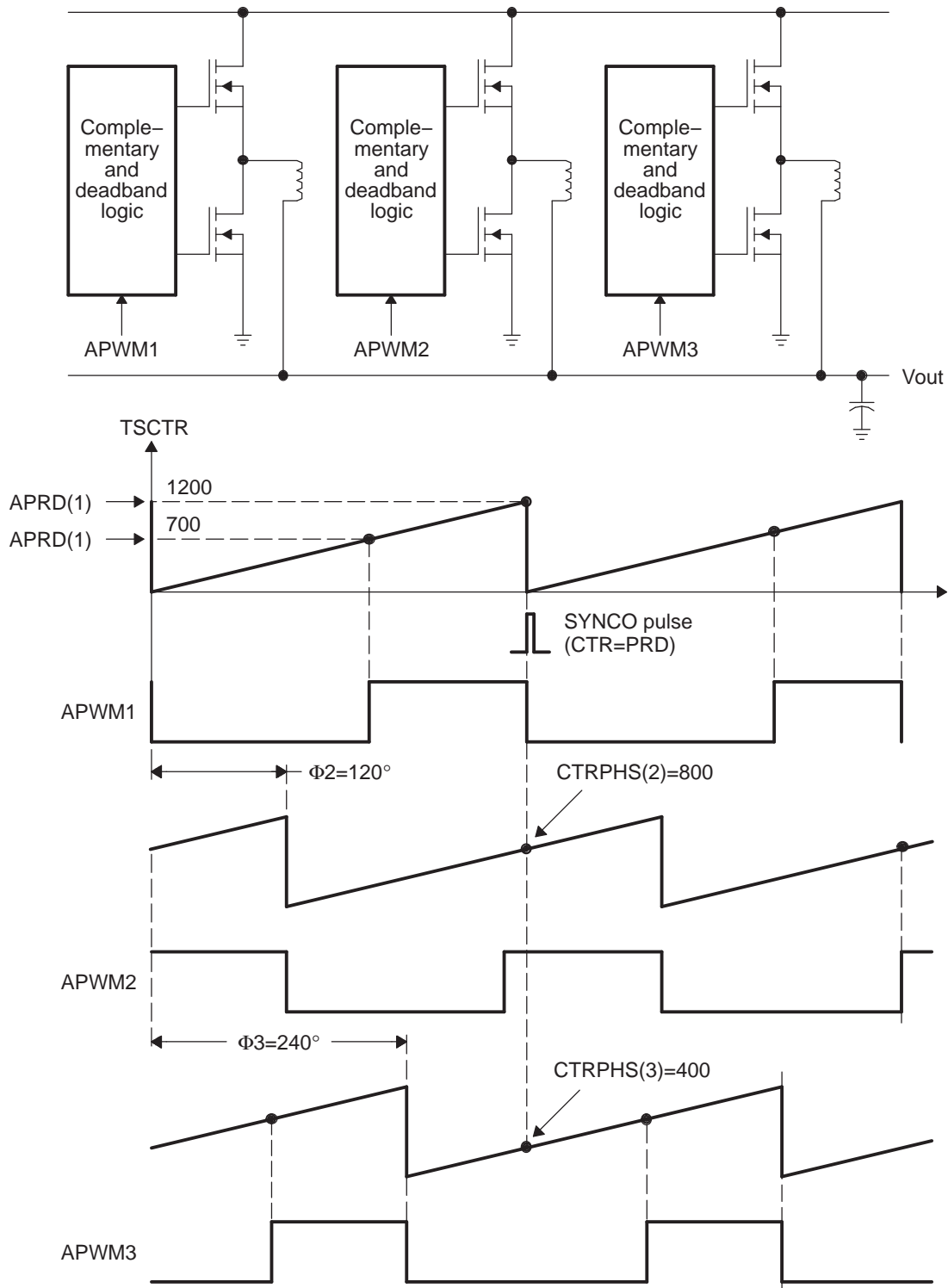
// Run Time (Note: Example execution of one run-time instant)
//=====
ECAP1Regs.CAP2 = 7000;    // Set Duty cycle i.e., compare value = 7000
ECAP2Regs.CAP2 = 2000;    // Set Duty cycle i.e., compare value = 2000
ECAP3Regs.CAP2 = 550;     // Set Duty cycle i.e., compare value = 550
ECAP4Regs.CAP2 = 6500;    // Set Duty cycle i.e., compare value = 6500
```

**13.3.5.3 Multichannel PWM Generation with Phase Control Example**

In [Figure 13-16](#), the Phase control feature of the APWM mode is used to control a 3 phase Interleaved DC/DC converter topology. This topology requires each phase to be off-set by 120° from each other. Hence if “Leg” 1 (controlled by APWM1) is the reference Leg (or phase), that is, 0°, then Leg 2 need 120° off-set and Leg 3 needs 240° off-set. The waveforms in [Figure 13-16](#) show the timing relationship between each of the phases (Legs). Note eCAP1 module is the Master and issues a sync out pulse to the slaves (modules 2, 3) whenever TSCTR = Period value.



Figure 13-16. Multiphase (channel) Interleaved PWM Example Using 3 eCAP Modules



**Table 13-10. ECAP1 Initialization for Multichannel PWM Generation with Phase Control**

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

**Table 13-11. ECAP2 Initialization for Multichannel PWM Generation with Phase Control**

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	800
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCL
ECCTL2	TSCTRSTOP	EC_RUN

**Table 13-12. ECAP3 Initialization for Multichannel PWM Generation with Phase Control**

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	400
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCL_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

**Example 13-7. Code Snippet for Multichannel PWM Generation with Phase Control**

```
// Code snippet for APWM mode Example 3

// Run Time (Note: Example execution of one run-time instant)
//=====
// All phases are set to the same duty cycle
ECAP1Regs.CAP2 = 700;    // Set Duty cycle i.e. compare value = 700
ECAP2Regs.CAP2 = 700;    // Set Duty cycle i.e. compare value = 700
ECAP3Regs.CAP2 = 700;    // Set Duty cycle i.e. compare value = 700
```

## 13.4 Registers

Table 13-13 shows the eCAP module control and status register set. All 32-bit registers are aligned on even address boundaries and are organized in little-endian mode. The 16 least-significant bits of a 32-bit register are located on lowest address (even address).

**NOTE:** In APWM mode, writing to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

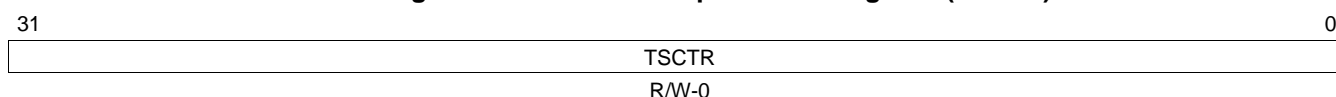
**Table 13-13. Control and Status Register Set**

Offset	Acronym	Description	Size (x16)	Section
0h	TSCTR	Time-Stamp Counter Register	2	<a href="#">Section 13.4.1</a>
4h	CTRPHS	Counter Phase Offset Value Register	2	<a href="#">Section 13.4.2</a>
8h	CAP1	Capture 1 Register	2	<a href="#">Section 13.4.3</a>
Ch	CAP2	Capture 2 Register	2	<a href="#">Section 13.4.4</a>
10h	CAP3	Capture 3 Register	2	<a href="#">Section 13.4.5</a>
14h	CAP4	Capture 4 Register	2	<a href="#">Section 13.4.6</a>
28h	ECCTL1	Capture Control Register 1	1	<a href="#">Section 13.4.7</a>
2Ah	ECCTL2	Capture Control Register 2	1	<a href="#">Section 13.4.8</a>
2Ch	ECEINT	Capture Interrupt Enable Register	1	<a href="#">Section 13.4.9</a>
2Eh	ECFLG	Capture Interrupt Flag Register	1	<a href="#">Section 13.4.10</a>
30h	ECCLR	Capture Interrupt Clear Register	1	<a href="#">Section 13.4.11</a>
32h	ECFRC	Capture Interrupt Force Register	1	<a href="#">Section 13.4.12</a>
5Ch	REVID	Revision ID Register	2	<a href="#">Section 13.4.13</a>

### 13.4.1 Time-Stamp Counter Register (TSCTR)

The time-stamp counter register (TSCTR) is shown in [Figure 13-17](#) and described in [Table 13-14](#).

**Figure 13-17. Time-Stamp Counter Register (TSCTR)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 13-14. Time-Stamp Counter Register (TSCTR) Field Descriptions**

Bit	Field	Value	Description
31-0	TSCTR	0-FFFF FFFFh	Active 32-bit counter register that is used as the capture time-base

### 13.4.2 Counter Phase Control Register (CTRPHS)

The counter phase control register (CTRPHS) is shown in [Figure 13-18](#) and described in [Table 13-15](#).

**Figure 13-18. Counter Phase Control Register (CTRPHS)**

31	0
CTRPHS	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 13-15. Counter Phase Control Register (CTRPHS) Field Descriptions**

Bit	Field	Value	Description
31-0	CTRPHS	0-FFFF FFFFh	Counter phase value register that can be programmed for phase lag/lead. This register shadows TSCTR and is loaded into TSCTR upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases.

### 13.4.3 Capture 1 Register (CAP1)

The capture 1 register (CAP1) is shown in [Figure 13-19](#) and described in [Table 13-16](#).

**Figure 13-19. Capture 1 Register (CAP1)**

31	0
CAP1	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

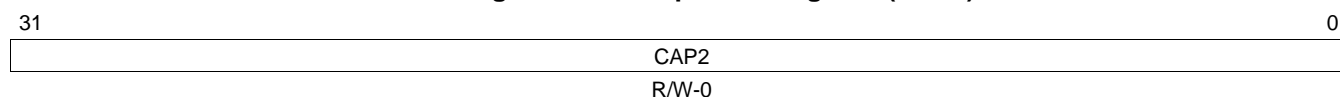
**Table 13-16. Capture 1 Register (CAP1) Field Descriptions**

Bit	Field	Value	Description
31-0	CAP1	0-FFFF FFFFh	This register can be loaded (written) by: <ul style="list-style-type: none"> <li>• Time-Stamp (i.e., counter value) during a capture event</li> <li>• Software - may be useful for test purposes</li> <li>• APRD active register when used in APWM mode</li> </ul>

### 13.4.4 Capture 2 Register (CAP2)

The capture 2 register (CAP2) is shown in [Figure 13-20](#) and described in [Table 13-17](#).

**Figure 13-20. Capture 2 Register (CAP2)**



LEGEND: R/W = Read/Write; -n = value after reset

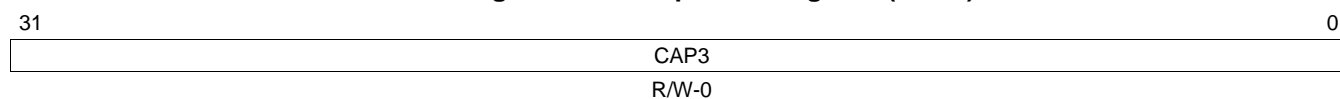
**Table 13-17. Capture 2 Register (CAP2) Field Descriptions**

Bit	Field	Value	Description
31-0	CAP2	0-FFFF FFFFh	This register can be loaded (written) by: <ul style="list-style-type: none"> <li>• Time-Stamp (i.e., counter value) during a capture event</li> <li>• Software - may be useful for test purposes</li> <li>• ACMP active register when used in APWM mode</li> </ul>

### 13.4.5 Capture 3 Register (CAP3)

The capture 3 register (CAP3) is shown in [Figure 13-21](#) and described in [Table 13-18](#).

**Figure 13-21. Capture 3 Register (CAP3)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 13-18. Capture 3 Register (CAP3) Field Descriptions**

Bit	Field	Value	Description
31-0	CAP3	0-FFFF FFFFh	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. You update the PWM period value through this register. In this mode, CAP3 shadows CAP1.

### 13.4.6 Capture 4 Register (CAP4)

The capture 4 register (CAP4) is shown in [Figure 13-22](#) and described in [Table 13-19](#).

**Figure 13-22. Capture 4 Register (CAP4)**

31	0
CAP4	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 13-19. Capture 4 Register (CAP4) Field Descriptions**

Bit	Field	Value	Description
31-0	CAP4	0-FFFF FFFFh	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. You update the PWM compare value through this register. In this mode, CAP4 shadows CAP2.

### 13.4.7 ECAP Control Register 1 (ECCTL1)

The ECAP control register 1 (ECCTL1) is shown in [Figure 13-23](#) and described in [Table 13-20](#).

**Figure 13-23. ECAP Control Register 1 (ECCTL1)**

15	14	13	9	8			
FREE/SOFT		PRESCALE		CAPLDEN			
R/W-0		R/W-0		R/W-0			
7	6	5	4	3	2	1	0
CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL	CTRRST1	CAP1POL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 13-20. ECAP Control Register 1 (ECCTL1) Field Descriptions**

Bit	Field	Value	Description
15-14	FREE/SOFT	0-3h 0 1h 2h-3h	Emulation Control TSCTR counter stops immediately on emulation suspend TSCTR counter runs until = 0 TSCTR counter is unaffected by emulation suspend (Run Free)
13-9	PRESCALE	0-1Fh 0 1 2h 3h 4h 5h ... 1Eh 1Fh	Event Filter prescale select Divide by 1 (i.e., no prescale, by-pass the prescaler) Divide by 2 Divide by 4 Divide by 6 Divide by 8 Divide by 10 ... Divide by 60 Divide by 62
8	CAPLDEN	0 1	Enable Loading of CAP1-4 registers on a capture event Disable CAP1-4 register loads at capture event time. Enable CAP1-4 register loads at capture event time.

**Table 13-20. ECAP Control Register 1 (ECCTL1) Field Descriptions (continued)**

Bit	Field	Value	Description
7	CTRRST4	0	Counter Reset on Capture Event 4 <i>Do not</i> reset counter on Capture Event 4 (absolute time stamp operation)
		1	Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation)
6	CAP4POL	0	Capture Event 4 Polarity select Capture Event 4 triggered on a rising edge (RE)
		1	Capture Event 4 triggered on a falling edge (FE)
5	CTRRST3	0	Counter Reset on Capture Event 3 <i>Do not</i> reset counter on Capture Event 3 (absolute time stamp)
		1	Reset counter after Event 3 time-stamp has been captured (used in difference mode operation)
4	CAP3POL	0	Capture Event 3 Polarity select Capture Event 3 triggered on a rising edge (RE)
		1	Capture Event 3 triggered on a falling edge (FE)
3	CTRRST2	0	Counter Reset on Capture Event 2 <i>Do not</i> reset counter on Capture Event 2 (absolute time stamp)
		1	Reset counter after Event 2 time-stamp has been captured (used in difference mode operation)
2	CAP2POL	0	Capture Event 2 Polarity select Capture Event 2 triggered on a rising edge (RE)
		1	Capture Event 2 triggered on a falling edge (FE)
1	CTRRST1	0	Counter Reset on Capture Event 1 <i>Do not</i> reset counter on Capture Event 1 (absolute time stamp)
		1	Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)
0	CAP1POL	0	Capture Event 1 Polarity select Capture Event 1 triggered on a rising edge (RE)
		1	Capture Event 1 triggered on a falling edge (FE)

### 13.4.8 ECAP Control Register 2 (ECCTL2)

The ECAP control register 2 (ECCTL2) is shown in [Figure 13-24](#) and described in [Table 13-21](#).

**Figure 13-24. ECAP Control Register 2 (ECCTL2)**

15			11			10		9		8	
Reserved						APWMPOL	CAP/APWM	SWSYNC			
R-0						R/W-0	R/W-0	R/W-0			
7		6		5		4		3		2	
SYNCO_SEL		SYNCl_EN		TSCTRSTOP		RE-ARM		STOP_WRAP		CONT/ONESHT	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-1		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 13-21. ECAP Control Register 2 (ECCTL2) Field Descriptions**

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10	APWMPOL	0 1	APWM output polarity select. This is applicable only in APWM operating mode 0 Output is active high (Compare value defines high time) 1 Output is active low (Compare value defines low time)
9	CAP/APWM	0 1	CAP/APWM operating mode select 0 ECAP module operates in capture mode. This mode forces the following configuration: <ul style="list-style-type: none"> <li>Inhibits TSCTR resets via CTR = PRD event</li> <li>Inhibits shadow loads on CAP1 and 2 registers</li> <li>Permits user to enable CAP1-4 register load</li> <li>ECAPn/APWMn pin operates as a capture input</li> </ul> 1 ECAP module operates in APWM mode. This mode forces the following configuration: <ul style="list-style-type: none"> <li>Resets TSCTR on CTR = PRD event (period boundary)</li> <li>Permits shadow loading on CAP1 and 2 registers</li> <li>Disables loading of time-stamps into CAP1-4 registers</li> <li>ECAPn/APWMn pin operates as a APWM output</li> </ul>
8	SWSYNC	0 1	Software-forced Counter (TSCTR) Synchronizing. This provides a convenient software method to synchronize some or all ECAP time bases. In APWM mode, the synchronizing can also be done via the CTR = PRD event. 0 Writing a zero has no effect. Reading always returns a zero 1 Writing a one forces a TSCTR shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 0,0. After writing a 1, this bit returns to a zero. Note: Selection CTR = PRD is meaningful only in APWM mode; however, you can choose it in CAP mode if you find doing so useful.
7-6	SYNCO_SEL	0-3h 0 1h 2h 3h	Sync-Out Select 0 Select sync-in event to be the sync-out signal (pass through) 1h Select CTR = PRD event to be the sync-out signal 2h Disable sync out signal 3h Disable sync out signal
5	SYNCl_EN	0 1	Counter (TSCTR) Sync-In select mode 0 Disable sync-in option 1 Enable counter (TSCTR) to be loaded from CTRPHS register upon either a SYNCl signal or a S/W force event.
4	TSCTRSTOP	0 1	Time Stamp (TSCTR) Counter Stop (freeze) Control 0 TSCTR stopped 1 TSCTR free-running



**Table 13-21. ECAP Control Register 2 (ECCTL2) Field Descriptions (continued)**

Bit	Field	Value	Description
3	RE-ARM	0 1	One-Shot Re-Arming Control, that is, wait for stop trigger. Note: The re-arm function is valid in one shot or continuous mode. 0 Has no effect (reading always returns a 0) 1 Arms the one-shot sequence as follows: 1) Resets the Mod4 counter to zero 2) Unfreezes the Mod4 counter 3) Enables capture register loads
2-1	STOP_WRAP	0-3h 0 1h 2h 3h	Stop value for one-shot mode. This is the number (between 1-4) of captures allowed to occur before the CAP(1-4) registers are frozen, that is, capture sequence is stopped. Wrap value for continuous mode. This is the number (between 1-4) of the capture register in which the circular buffer wraps around and starts again. 0 Stop after Capture Event 1 in one-shot mode. Wrap after Capture Event 1 in continuous mode. 1h Stop after Capture Event 2 in one-shot mode. Wrap after Capture Event 2 in continuous mode. 2h Stop after Capture Event 3 in one-shot mode. Wrap after Capture Event 3 in continuous mode. 3h Stop after Capture Event 4 in one-shot mode. Wrap after Capture Event 4 in continuous mode. Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur: <ul style="list-style-type: none"> <li>• Mod4 counter is stopped (frozen)</li> <li>• Capture register loads are inhibited</li> </ul> In one-shot mode, further interrupt events are blocked until re-armed.
0	CONT/ONESHT	0 1	Continuous or one-shot mode control (applicable only in capture mode) 0 Operate in continuous mode 1 Operate in one-shot mode

### 13.4.9 ECAP Interrupt Enable Register (ECEINT)

The ECAP interrupt enable register (ECEINT) is shown in [Figure 13-25](#) and described in [Table 13-22](#).

The interrupt enable bits (CEVT $n$ ) block any of the selected events from generating an interrupt. Events will still be latched into the flag bit (ECFLG register) and can be forced/cleared via the ECFRC/ECCLR registers.

The proper procedure for configuring peripheral modes and interrupts is:

1. Disable global interrupts
2. Stop eCAP counter
3. Disable eCAP interrupts
4. Configure peripheral registers
5. Clear spurious eCAP interrupt flags
6. Enable eCAP interrupts
7. Start eCAP counter
8. Enable global interrupts

**Figure 13-25. ECAP Interrupt Enable Register (ECEINT)**

Reserved							
R-0							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 13-22. ECAP Interrupt Enable Register (ECEINT) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	CTR=CMP	0 1	Counter Equal Compare Interrupt Enable Disable Compare Equal as an Interrupt source Enable Compare Equal as an Interrupt source
6	CTR=PRD	0 1	Counter Equal Period Interrupt Enable Disable Period Equal as an Interrupt source Enable Period Equal as an Interrupt source
5	CTROVF	0 1	Counter Overflow Interrupt Enable Disable counter Overflow as an Interrupt source Enable counter Overflow as an Interrupt source
4	CEVT4	0 1	Capture Event 4 Interrupt Enable Disable Capture Event 4 as an Interrupt source Enable Capture Event 4 as an Interrupt source
3	CEVT3	0 1	Capture Event 3 Interrupt Enable Disable Capture Event 3 as an Interrupt source Enable Capture Event 3 as an Interrupt source
2	CEVT2	0 1	Capture Event 2 Interrupt Enable Disable Capture Event 2 as an Interrupt source Enable Capture Event 2 as an Interrupt source
1	CEVT1	0 1	Capture Event 1 Interrupt Enable Disable Capture Event 1 as an Interrupt source Enable Capture Event 1 as an Interrupt source
0	Reserved	0	Reserved

### 13.4.10 ECAP Interrupt Flag Register (ECFLG)

The ECAP interrupt flag register (ECFLG) is shown in [Figure 13-26](#) and described in [Table 13-23](#).

**Figure 13-26. ECAP Interrupt Flag Register (ECFLG)**

Reserved							
R-0							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CETV3	CEVT2	CETV1	INT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 13-23. ECAP Interrupt Flag Register (ECFLG) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	CTR=CMP	0	Compare Equal Compare Status Flag. This flag is only active in APWM mode. Indicates no event occurred
		1	Indicates the counter (TSCTR) reached the compare register value (ACMP)
6	CTR=PRD	0	Counter Equal Period Status Flag. This flag is only active in APWM mode. Indicates no event occurred
		1	Indicates the counter (TSCTR) reached the period register value (APRD) and was reset.
5	CTROVF	0	Counter Overflow Status Flag. This flag is active in CAP and APWM mode. Indicates no event occurred.
		1	Indicates the counter (TSCTR) has made the transition from 0xFFFFFFFF to 0x00000000
4	CEVT4	0	Capture Event 4 Status Flag This flag is only active in CAP mode. Indicates no event occurred
		1	Indicates the fourth event occurred at ECAP <sub>n</sub> pin
3	CEVT3	0	Capture Event 3 Status Flag. This flag is active only in CAP mode. Indicates no event occurred.
		1	Indicates the third event occurred at ECAP <sub>n</sub> pin.
2	CEVT2	0	Capture Event 2 Status Flag. This flag is only active in CAP mode. Indicates no event occurred.
		1	Indicates the second event occurred at ECAP <sub>n</sub> pin.
1	CEVT1	0	Capture Event 1 Status Flag. This flag is only active in CAP mode. Indicates no event occurred.
		1	Indicates the first event occurred at ECAP <sub>n</sub> pin.
0	INT	0	Global Interrupt Status Flag Indicates no interrupt generated.
		1	Indicates that an interrupt was generated.

### 13.4.11 ECAP Interrupt Clear Register (ECCLR)

The ECAP interrupt clear register (ECCLR) is shown in [Figure 13-27](#) and described in [Table 13-24](#).

**Figure 13-27. ECAP Interrupt Clear Register (ECCLR)**

Reserved							
R-0							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CETV3	CETV2	CETV1	INT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 13-24. ECAP Interrupt Clear Register (ECCLR) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	CTR=CMP	0 1	Counter Equal Compare Status Flag Writing a 0 has no effect. Always reads back a 0 Writing a 1 clears the CTR=CMP flag condition
6	CTR=PRD	0 1	Counter Equal Period Status Flag Writing a 0 has no effect. Always reads back a 0 Writing a 1 clears the CTR=PRD flag condition
5	CTROVF	0 1	Counter Overflow Status Flag Writing a 0 has no effect. Always reads back a 0 Writing a 1 clears the CTROVF flag condition
4	CEVT4	0 1	Capture Event 4 Status Flag Writing a 0 has no effect. Always reads back a 0. Writing a 1 clears the CEVT3 flag condition.
3	CEVT3	0 1	Capture Event 3 Status Flag Writing a 0 has no effect. Always reads back a 0. Writing a 1 clears the CEVT3 flag condition.
2	CEVT2	0 0	Capture Event 2 Status Flag Writing a 0 has no effect. Always reads back a 0. Writing a 1 clears the CEVT2 flag condition.
1	CEVT1	0 1	Capture Event 1 Status Flag Writing a 0 has no effect. Always reads back a 0. Writing a 1 clears the CEVT1 flag condition.
0	INT	0 1	Global Interrupt Clear Flag Writing a 0 has no effect. Always reads back a 0. Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1.

### 13.4.12 ECAP Interrupt Forcing Register (ECFRC)

The ECAP interrupt forcing register (ECFRC) is shown in [Figure 13-28](#) and described in [Table 13-25](#).

**Figure 13-28. ECAP Interrupt Forcing Register (ECFRC)**

15	14	13	12	11	10	9	8
Reserved							
R-0							
7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CETV3	CETV2	CETV1	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

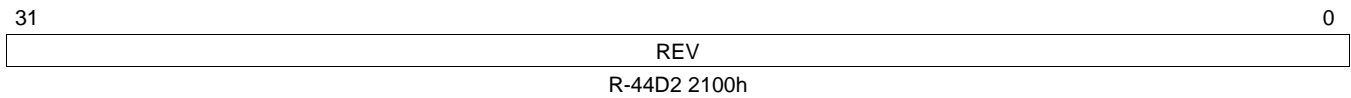
**Table 13-25. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	CTR=CMP	0	Force Counter Equal Compare Interrupt No effect. Always reads back a 0.
		1	Writing a 1 sets the CTR=CMP flag bit.
6	CTR=PRD	0	Force Counter Equal Period Interrupt No effect. Always reads back a 0.
		1	Writing a 1 sets the CTR=PRD flag bit.
5	CTROVF	0	Force Counter Overflow No effect. Always reads back a 0.
		1	Writing a 1 to this bit sets the CTROVF flag bit.
4	CEVT4	0	Force Capture Event 4 No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT4 flag bit
3	CEVT3	0	Force Capture Event 3 No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT3 flag bit
2	CEVT2	0	Force Capture Event 2 No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT2 flag bit.
1	CEVT1	0	Force Capture Event 1 No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT1 flag bit.
0	Reserved	0	Reserved

### 13.4.13 Revision ID Register (REVID)

The revision ID register (REVID) is shown in [Figure 13-29](#) and described in [Table 13-26](#).

**Figure 13-29. Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

**Table 13-26. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	44D2 2100h	Revision ID.

## ***Enhanced High-Resolution Pulse-Width Modulator (eHRPWM)***

---

---

---

This chapter describes the enhanced high-resolution pulse-width modulator (eHRPWM).

<b>Topic</b>	<b>Page</b>
<b>14.1 Introduction .....</b>	<b>333</b>
<b>14.2 Architecture .....</b>	<b>338</b>
<b>14.3 Applications to Power Topologies .....</b>	<b>397</b>
<b>14.4 Registers .....</b>	<b>421</b>

## 14.1 Introduction

### 14.1.1 Introduction

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel modules with separate resources and that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In this chapter, the letter x within a signal or module name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and, likewise, EPWM4A and EPWM4B belong to ePWM4.

### 14.1.2 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in [Figure 14-1](#). Each ePWM instance is identical with one exception. Some instances include a hardware extension that allows more precise control of the PWM outputs. This extension is the high-resolution pulse width modulator (HRPWM) and is described in [Section 14.2.10](#). See your device-specific data manual to determine which ePWM instances include this feature. Each ePWM module is indicated by a numerical value starting with 1. For example ePWM1 is the first instance and ePWM3 is the 3rd instance in the system and ePWMx indicates any instance.

The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral modules (eCAP). The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

Each ePWM module supports the following features:

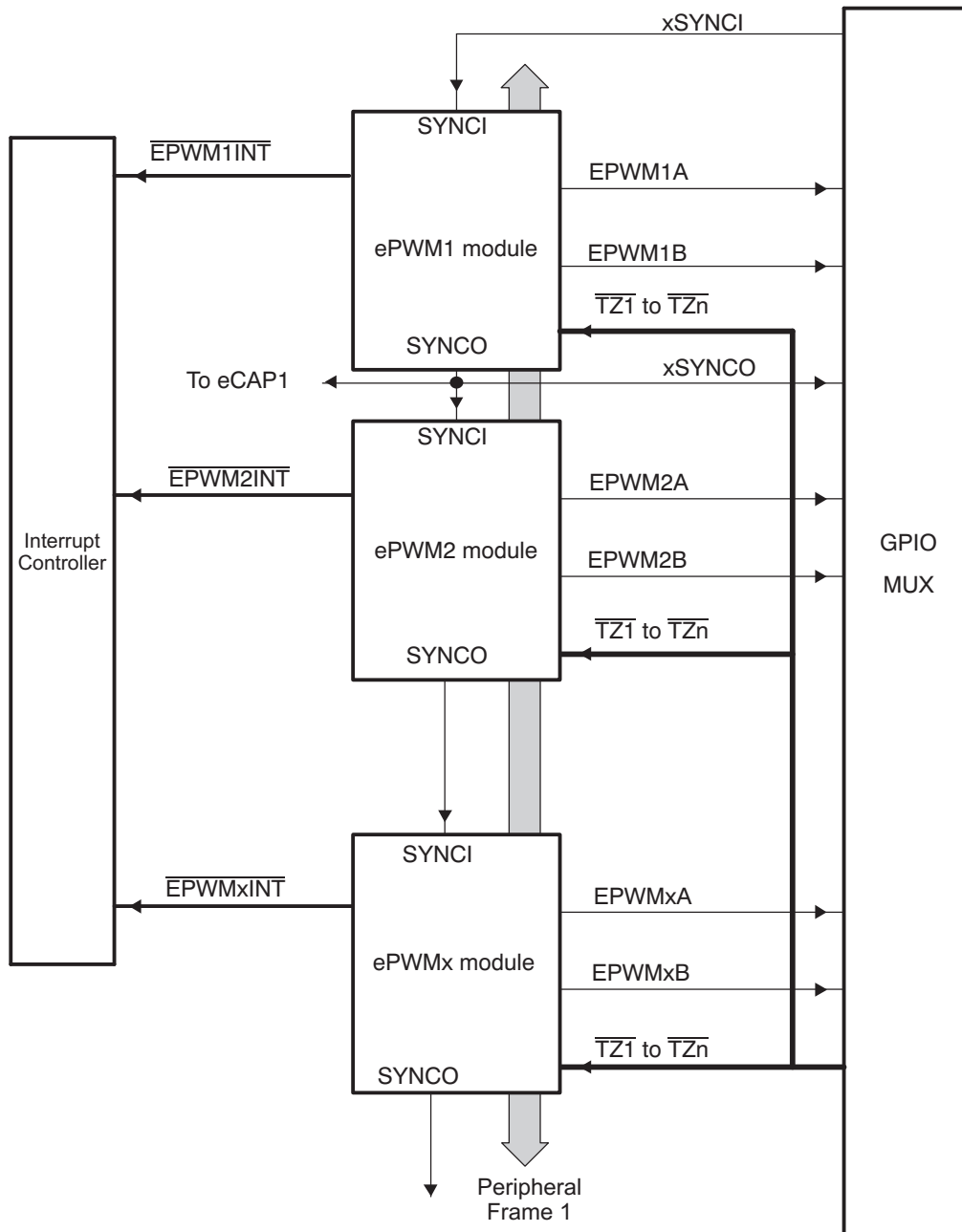
- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations::
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in [Figure 14-1](#). The signals are described in detail in subsequent sections.

The order in which the ePWM modules are connected may differ from what is shown in [Figure 14-1](#). See [Section 14.2.3.3.2](#) for the synchronization scheme for a particular device. Each ePWM module consists of seven submodules and is connected within a system via the signals shown in [Figure 14-2](#).



Figure 14-1. Multiple ePWM Modules



**Figure 14-2. Submodules and Signal Connections for an ePWM Module**

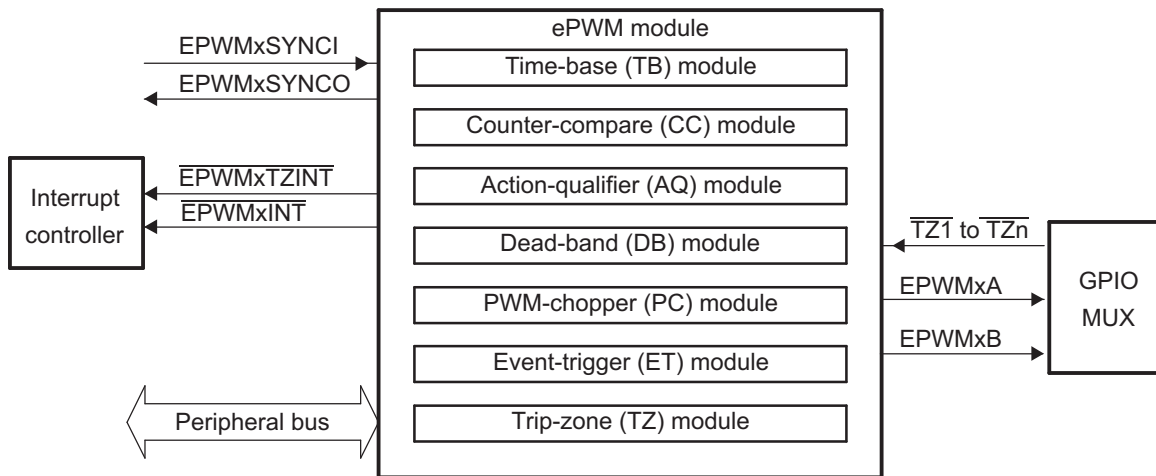
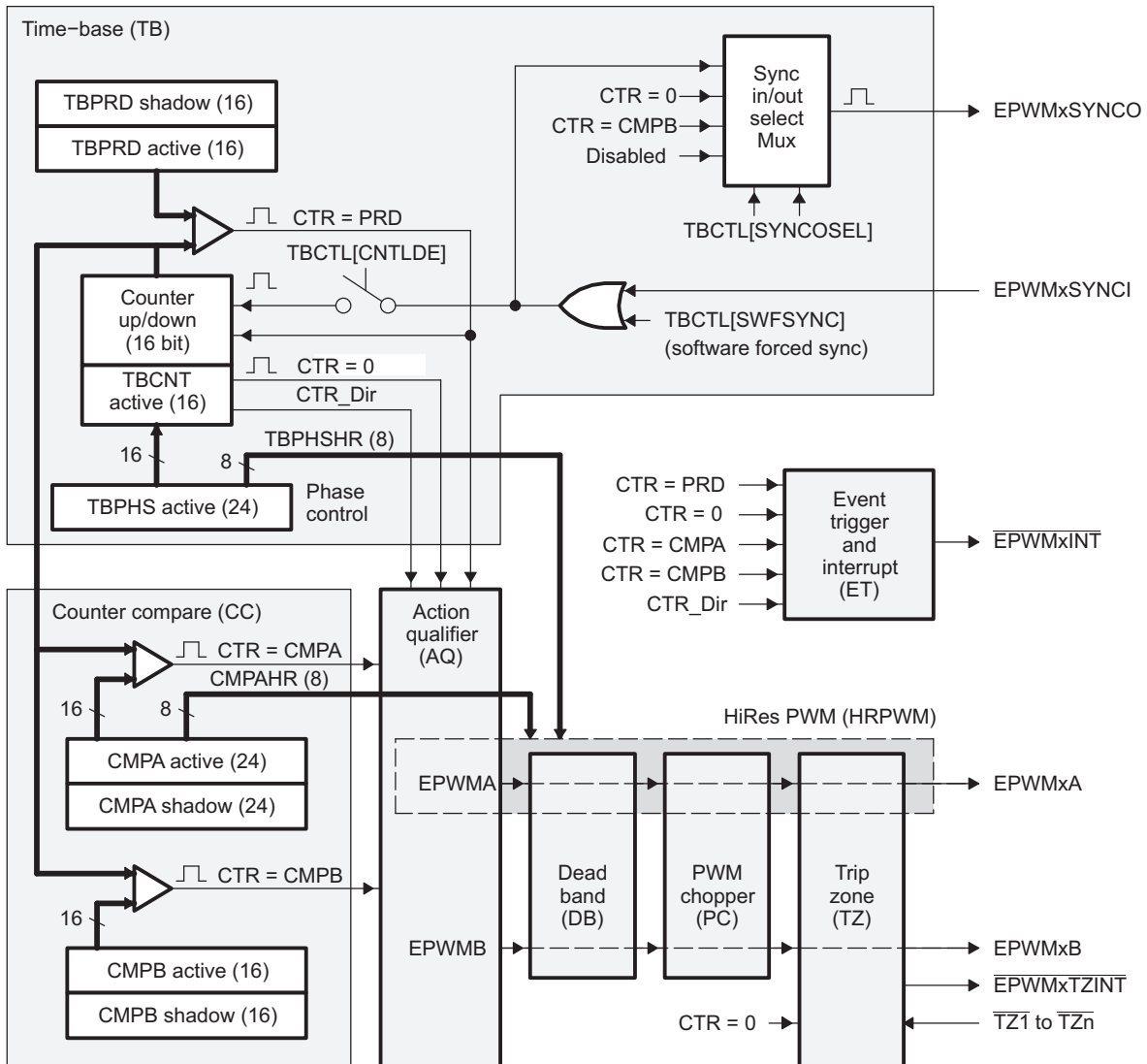


Figure 14-3 shows more internal details of a single ePWM module. The main signals used by the ePWM module are:

- **PWM output signals (EPWMxA and EPWMxB).** The PWM output signals are made available external to the device through the GPIO peripheral described in the system control and interrupts guide for your device.
- **Trip-zone signals ( $\overline{TZ1}$  to  $\overline{TZn}$ ).** These input signals alert the ePWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signal can be configured as an asynchronous input through the GPIO peripheral. See your device-specific data manual to determine how many trip-zone pins are available in the device.
- **Time-base synchronization input (EPWMxSYNCl) and output (EPWMxSYNCO) signals.** The synchronization signals daisy chain the ePWM modules together. Each module can be configured to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The synchronization output for ePWM1 (EPWM1SYNCO) is also connected to the SYNCl of the first enhanced capture module (eCAP1).
- **Peripheral Bus.** The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

Figure 14-3 also shows the key internal submodule interconnect signals. Each submodule is described in detail in Section 14.2.

Figure 14-3. ePWM Submodules and Critical Internal Signal Interconnects



### 14.1.3 Register Mapping

Table 14-1 shows the complete ePWM module control and status register set grouped by submodule. Each register set is duplicated for each instance of the ePWM module. The start address for each ePWM register file instance on a device is specified in the appropriate data manual.

**Table 14-1. ePWM Module Control and Status Registers Grouped by Submodule**

Acronym	Offset <sup>(1)</sup>	Size (x16)	Shadow	Register Description
<b>Time-Base Submodule Registers</b>				
TBCTL	0h	1	No	Time-Base Control Register
TBSTS	2h	1	No	Time-Base Status Register
TBPHSHR	4h	1	No	Extension for HRPWM Phase Register <sup>(2)</sup>
TBPHS	6h	1	No	Time-Base Phase Register
TBCNT	8h	1	No	Time-Base Counter Register
TBPRD	Ah	1	Yes	Time-Base Period Register
<b>Counter-Compare Submodule Registers</b>				
CMPCTL	Eh	1	No	Counter-Compare Control Register
CMPAHR	10h	1	No	Extension for HRPWM Counter-Compare A Register <sup>(2)</sup>
CMPA	12h	1	Yes	Counter-Compare A Register
CMPB	14h	1	Yes	Counter-Compare B Register
<b>Action-Qualifier Submodule Registers</b>				
AQCTLA	16h	1	No	Action-Qualifier Control Register for Output A (EPWMxA)
AQCTLB	18h	1	No	Action-Qualifier Control Register for Output B (EPWMxB)
AQSFRC	1Ah	1	No	Action-Qualifier Software Force Register
AQCSFRC	1Ch	1	Yes	Action-Qualifier Continuous S/W Force Register Set
<b>Dead-Band Generator Submodule Registers</b>				
DBCTL	1Eh	1	No	Dead-Band Generator Control Register
DBRED	20h	1	No	Dead-Band Generator Rising Edge Delay Count Register
DBFED	22h	1	No	Dead-Band Generator Falling Edge Delay Count Register
<b>PWM-Chopper Submodule Registers</b>				
PCCTL	3Ch	1	No	PWM-Chopper Control Register
<b>Trip-Zone Submodule Registers</b>				
TZSEL	24h	1	No	Trip-Zone Select Register
TZCTL	28h	1	No	Trip-Zone Control Register
TZEINT	2Ah	1	No	Trip-Zone Enable Interrupt Register
TZFLG	2Ch	1	No	Trip-Zone Flag Register
TZCLR	2Eh	1	No	Trip-Zone Clear Register
TZFRC	30h	1	No	Trip-Zone Force Register
<b>Event-Trigger Submodule Registers</b>				
ETSEL	32h	1	No	Event-Trigger Selection Register
ETPS	34h	1	No	Event-Trigger Pre-Scale Register
ETFLG	36h	1	No	Event-Trigger Flag Register
ETCLR	38h	1	No	Event-Trigger Clear Register
ETFRC	3Ah	1	No	Event-Trigger Force Register
<b>High-Resolution PWM (HRPWM) Submodule Registers</b>				
HRCNFG	1040h	1	No	HRPWM Configuration Register <sup>(2)</sup>

<sup>(1)</sup> Locations not shown are reserved.

<sup>(2)</sup> These registers are only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, these locations are reserved. See your device-specific data manual to determine which instances include the HRPWM.

## 14.2 Architecture

Seven submodules are included in every ePWM peripheral. There are some instances that include a high-resolution submodule that allows more precise control of the PWM outputs. Each of these submodules performs specific tasks that can be configured by software.

### 14.2.1 Overview

Table 14-2 lists the eight key submodules together with a list of their main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, then you should see the counter-compare submodule in Section 14.2.4 for relevant details.

**Table 14-2. Submodule Configuration Parameters**

Submodule	Configuration Parameter or Option	Reference
Time-base (TB)	<ul style="list-style-type: none"> <li>• Scale the time-base clock (TBCLK) relative to the system clock (SYSCLKOUT).</li> <li>• Configure the PWM time-base counter (TBCNT) frequency or period.</li> <li>• Set the mode for the time-base counter:               <ul style="list-style-type: none"> <li>– count-up mode: used for asymmetric PWM</li> <li>– count-down mode: used for asymmetric PWM</li> <li>– count-up-and-down mode: used for symmetric PWM</li> </ul> </li> <li>• Configure the time-base phase relative to another ePWM module.</li> <li>• Synchronize the time-base counter between modules through hardware or software.</li> <li>• Configure the direction (up or down) of the time-base counter after a synchronization event.</li> <li>• Configure how the time-base counter will behave when the device is halted by an emulator.</li> <li>• Specify the source for the synchronization output of the ePWM module:               <ul style="list-style-type: none"> <li>– Synchronization input signal</li> <li>– Time-base counter equal to zero</li> <li>– Time-base counter equal to counter-compare B (CMPB)</li> <li>– No output synchronization signal generated.</li> </ul> </li> </ul>	Section 14.2.3
Counter-compare (CC)	<ul style="list-style-type: none"> <li>• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB</li> <li>• Specify the time at which switching events occur on the EPWMxA or EPWMxB output</li> </ul>	Section 14.2.4
Action-qualifier (AQ)	<ul style="list-style-type: none"> <li>• Specify the type of action taken when a time-base or counter-compare submodule event occurs:               <ul style="list-style-type: none"> <li>– No action taken</li> <li>– Output EPWMxA and/or EPWMxB switched high</li> <li>– Output EPWMxA and/or EPWMxB switched low</li> <li>– Output EPWMxA and/or EPWMxB toggled</li> </ul> </li> <li>• Force the PWM output state through software control</li> <li>• Configure and control the PWM dead-band through software</li> </ul>	Section 14.2.5
Dead-band (DB)	<ul style="list-style-type: none"> <li>• Control of traditional complementary dead-band relationship between upper and lower switches</li> <li>• Specify the output rising-edge-delay value</li> <li>• Specify the output falling-edge delay value</li> <li>• Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>	Section 14.2.6
PWM-chopper (PC)	<ul style="list-style-type: none"> <li>• Create a chopping (carrier) frequency.</li> <li>• Pulse width of the first pulse in the chopped pulse train.</li> <li>• Duty cycle of the second and subsequent pulses.</li> <li>• Bypass the PWM-chopper module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>	Section 14.2.7

**Table 14-2. Submodule Configuration Parameters (continued)**

Submodule	Configuration Parameter or Option	Reference
Trip-zone (TZ)	<ul style="list-style-type: none"> <li>• Configure the ePWM module to react to one, all, or none of the trip-zone pins.</li> <li>• Specify the tripping action taken when a fault occurs:               <ul style="list-style-type: none"> <li>– Force EPWMxA and/or EPWMxB high</li> <li>– Force EPWMxA and/or EPWMxB low</li> <li>– Force EPWMxA and/or EPWMxB to a high-impedance state</li> <li>– Configure EPWMxA and/or EPWMxB to ignore any trip condition.</li> </ul> </li> <li>• Configure how often the ePWM will react to each trip-zone pin:               <ul style="list-style-type: none"> <li>– One-shot</li> <li>– Cycle-by-cycle</li> </ul> </li> <li>• Enable the trip-zone to initiate an interrupt.</li> <li>• Bypass the trip-zone module entirely.</li> </ul>	<a href="#">Section 14.2.8</a>
Event-trigger (ET)	<ul style="list-style-type: none"> <li>• Enable the ePWM events that will trigger an interrupt.</li> <li>• Specify the rate at which events cause triggers (every occurrence or every second or third occurrence)</li> <li>• Poll, set, or clear event flags</li> </ul>	<a href="#">Section 14.2.9</a>
High-Resolution PWM (HRPWM)	<ul style="list-style-type: none"> <li>• Enable extended time resolution capabilities</li> <li>• Configure finer time granularity control or edge positioning</li> </ul>	<a href="#">Section 14.2.10</a>

Code examples are provided in the remainder of this chapter that show how to implement various ePWM module configurations. These examples use the constant definitions shown in [Example 14-1](#).

### Example 14-1. Constant Definitions Used in the Code Examples

```

// TBCTL (Time-Base Control)
// = = = = =
// TBCNT MODE bit
#define TB_COUNT_UP 0x0
#define TB_COUNT_DOWN 0x1
#define TB_COUNT_UPDOWN 0x2
#define TB_FREEZE 0x3
// PHSEN bit
#define TB_DISABLE 0x0
#define TB_ENABLE 0x1
// PRDL bit
#define TB_SHADOW 0x0
#define TB_IMMEDIATE 0x1
// SYNCSEL bit
#define TB_SYNC_IN 0x0
#define TB_CTR_ZERO 0x1
#define TB_CTR_CMPB 0x2
#define TB_SYNC_DISABLE 0x3
// HSPCLKDIV and CLKDIV bits
#define TB_DIV1 0x0
#define TB_DIV2 0x1
#define TB_DIV4 0x2
// PHSDIR bit
#define TB_DOWN 0x0
#define TB_UP 0x1
// CMPCTL (Compare Control)
// = = = = =
// LOADAMODE and LOADBMODE bits
#define CC_CTR_ZERO 0x0
#define CC_CTR_PRD 0x1
#define CC_CTR_ZERO_PRD 0x2
#define CC_LD_DISABLE 0x3
// SHDWAMODE and SHDWBMODE bits
#define CC_SHADOW 0x0
#define CC_IMMEDIATE 0x1
// AQCTLA and AQCTLB (Action-qualifier Control)
// = = = = =
// ZRO, PRD, CAU, CAD, CBU, CBD bits
#define AQ_NO_ACTION 0x0
#define AQ_CLEAR 0x1
#define AQ_SET 0x2
#define AQ_TOGGLE 0x3
// DBCTL (Dead-Band Control)
// = = = = =
// MODE bit
#define DB_DISABLE 0x0
#define DBA_ENABLE 0x1
#define DBB_ENABLE 0x2
#define DB_FULL_ENABLE 0x3
// POLSEL bit
#define DB_ACTV_HI 0x0
#define DB_ACTV_LO 0x1
#define DB_ACTV_HIC 0x2
#define DB_ACTV_LO 0x3
// PCCTL (chopper control)
// = = = = =
// CHPEN bit
#define CHP_DISABLE 0x0
#define CHP_ENABLE 0x1

```

**Example 14-1. Constant Definitions Used in the Code Examples (continued)**

```

// CHPFREQ bit
#define      CHP_DIV1      0x0
#define      CHP_DIV2      0x1
#define      CHP_DIV3      0x2
#define      CHP_DIV4      0x3
#define      CHP_DIV5      0x4
#define      CHP_DIV6      0x5
#define      CHP_DIV7      0x6
#define      CHP_DIV8      0x7
// CHPDUTY bit
#define      CHP1_8TH      0x0
#define      CHP2_8TH      0x1
#define      CHP3_8TH      0x2
#define      CHP4_8TH      0x3
#define      CHP5_8TH      0x4
#define      CHP6_8TH      0x5
#define      CHP7_8TH      0x6
// TZSEL (Trip-zone Select)
// = = = = =
// CBCn and OSHTn bits
#define      TZ_DISABLE    0x0
#define      TZ_ENABLE     0x1
// TZCTL (Trip-zone Control)
// = = = = =
// TZA and TZB bits
#define      TZ_HIZ        0x0
#define      TZ_FORCE_HI   0x1
#define      TZ_FORCE_LO   0x2
#define      TZ_NONE       0x3
// ETSEL (Event-trigger Select)
// = = = = =
// INTSEL bit
#define      ET_CTR_ZERO   0x1
#define      ET_CTR_PRD    0x2
#define      ET_CTRU_CMPA  0x4
#define      ET_CTRD_CMPA  0x5
#define      ET_CTRU_CMPB  0x6
#define      ET_CTRD_CMPB  0x7
// ETPS (Event-trigger Prescale)
// = = = = =
// INTPRD bit
#define      ET_DISABLE    0x0
#define      ET_1ST        0x1
#define      ET_2ND        0x2
#define      ET_3RD        0x3

```

### 14.2.2 Proper Interrupt Initialization Procedure

When the ePWM peripheral clock is enabled it may be possible that interrupt flags may be set due to spurious events due to the ePWM registers not being properly initialized. The proper procedure for initializing the ePWM peripheral is:

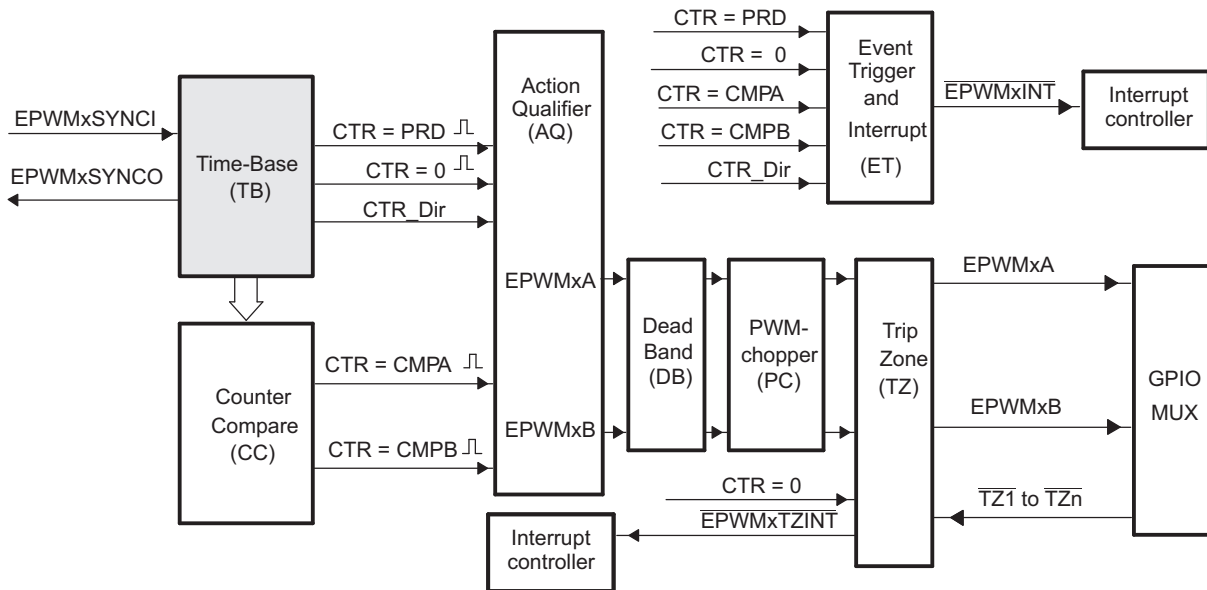
1. Disable global interrupts (CPU INTM flag)
2. Disable ePWM interrupts
3. Initialize peripheral registers
4. Clear any spurious ePWM flags
5. Enable ePWM interrupts
6. Enable global interrupts



### 14.2.3 Time-Base (TB) Submodule

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. Figure 14-4 illustrates the time-base module's place within the ePWM.

**Figure 14-4. Time-Base Submodule Block Diagram**



#### 14.2.3.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCNT) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCNT = TBPRD) .
  - CTR = 0: Time-base counter equal to zero (TBCNT = 0000h).
- Configure the rate of the time-base clock; a prescaled version of the CPU system clock (SYSCLKOUT). This allows the time-base counter to increment/decrement at a slower rate.

### 14.2.3.2 Controlling and Monitoring the Time-Base Submodule

Table 14-3 lists the registers used to control and monitor the time-base submodule.

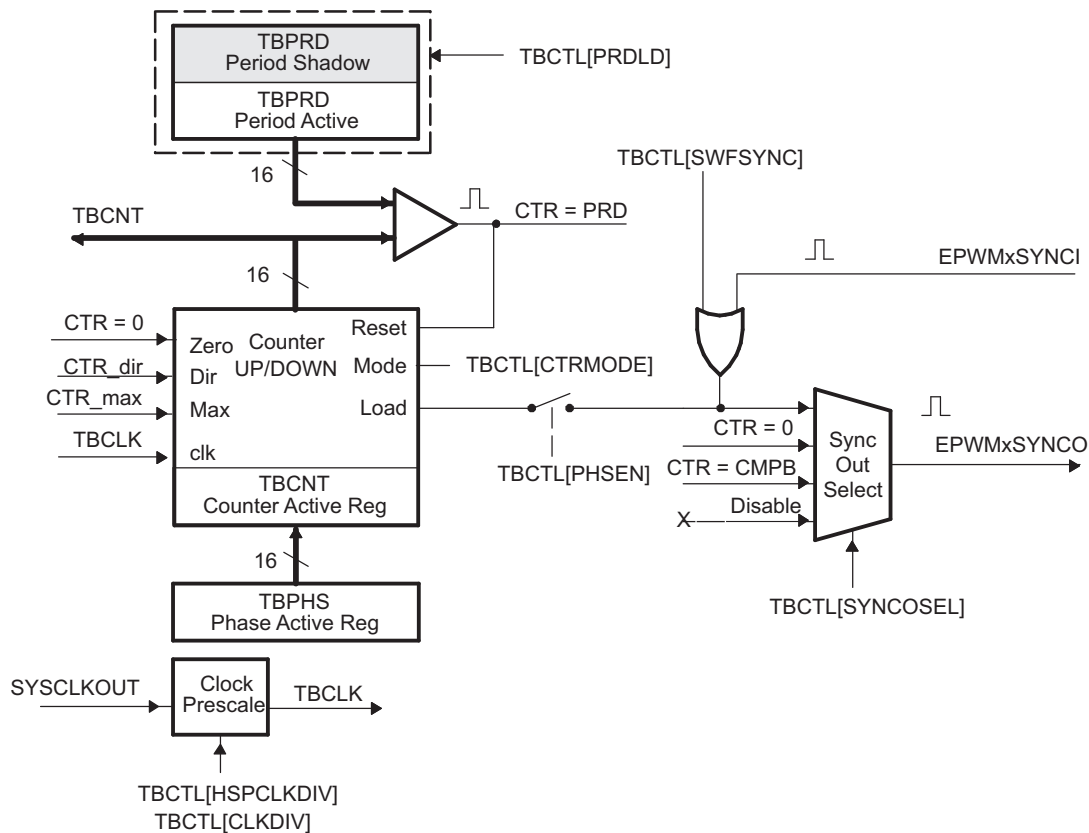
**Table 14-3. Time-Base Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
TBCTL	Time-Base Control Register	0h	No
TBSTS	Time-Base Status Register	2h	No
TBPHSHR	HRPWM extension Phase Register <sup>(1)</sup>	4h	No
TBPHS	Time-Base Phase Register	6h	No
TBCNT	Time-Base Counter Register	8h	No
TBPRD	Time-Base Period Register	Ah	Yes

<sup>(1)</sup> This register is available only on ePWM instances that include the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM, this location is reserved. See your device-specific data manual to determine which ePWM instances include this feature.

Figure 14-5 shows the critical signals and registers of the time-base submodule. Table 14-4 provides descriptions of the key signals associated with the time-base submodule.

**Figure 14-5. Time-Base Submodule Signals and Registers**



**Table 14-4. Key Time-Base Signals**

Signal	Description
EPWMxSYNCl	Time-base synchronization input.  Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module (EPWM1) this signal comes from a device pin. For subsequent ePWM modules this signal is passed from another ePWM peripheral. For example, EPWM2SYNCl is generated by the ePWM1 peripheral, EPWM3SYNCl is generated by ePWM2 and so forth. See <a href="#">Section 14.2.3.3.2</a> for information on the synchronization order of a particular device.
EPWMxSYNCO	Time-base synchronization output.  This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources: <ol style="list-style-type: none"> <li>1. EPWMxSYNCl (Synchronization input pulse)</li> <li>2. CTR = 0: The time-base counter equal to zero (TBCNT = 0000h).</li> <li>3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCNT = CMPB) register.</li> </ol>
CTR = PRD	Time-base counter equal to the specified period.  This signal is generated whenever the counter value is equal to the active period register value. That is when TBCNT = TBPRD.
CTR = 0	Time-base counter equal to zero.  This signal is generated whenever the counter value is zero. That is when TBCNT equals 0000h.
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCNT = CMPB).  This event is generated by the counter-compare submodule and used by the synchronization out logic.
CTR_dir	Time-base counter direction.  Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CTR_max	Time-base counter equal max value. (TBCNT = FFFFh)  Generated event when the TBCNT value reaches its maximum value. This signal is only used only as a status bit.
TBCLK	Time-base clock.  This is a prescaled version of the system clock (SYSCLKOUT) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.

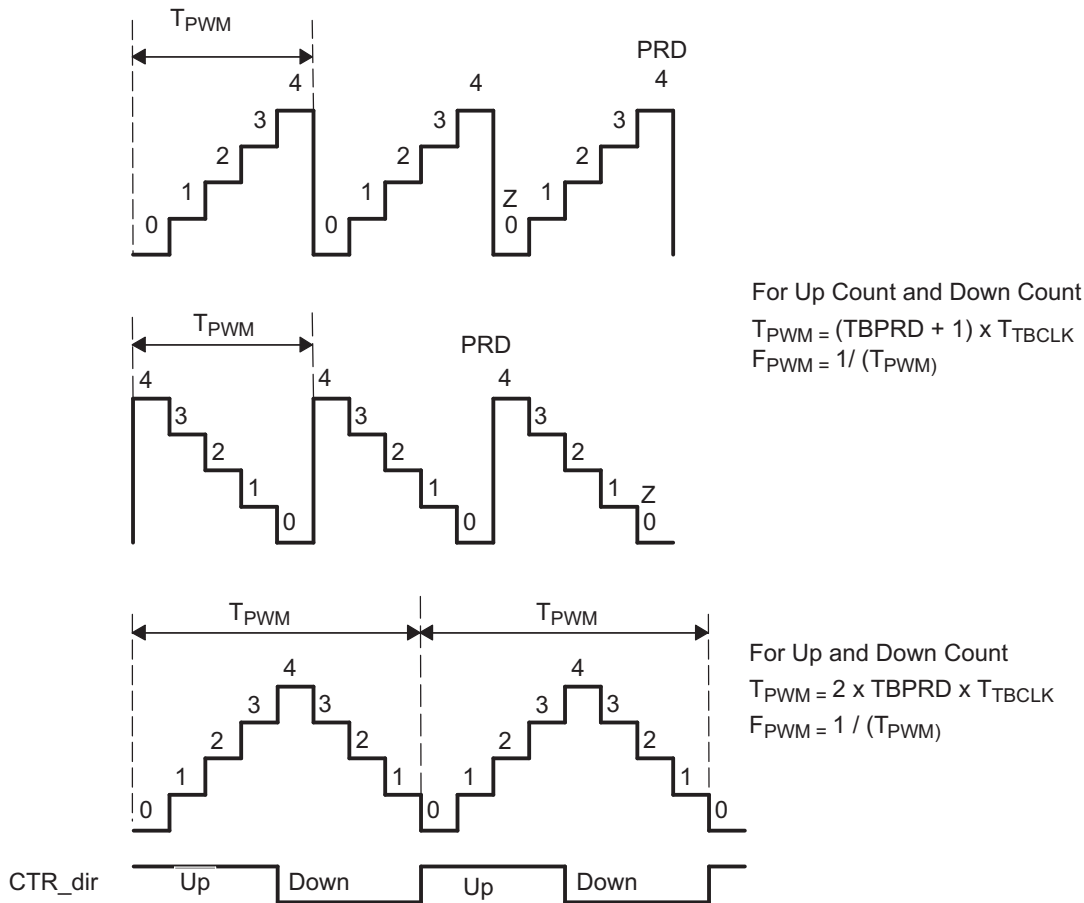
### 14.2.3.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. [Figure 14-6](#) shows the period ( $T_{pwm}$ ) and frequency ( $F_{pwm}$ ) relationships for the up-count, down-count, and up-down-count time-base counter modes when the period is set to 4 (TBPRD = 4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the system clock (SYSCLKOUT).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

- **Up-Down-Count Mode:** In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.
- **Up-Count Mode:** In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.
- **Down-Count Mode:** In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

Figure 14-6. Time-Base Frequency and Period



#### 14.2.3.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register:** The active register controls the hardware and is responsible for actions that the hardware causes or invokes.
- **Shadow Register:** The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

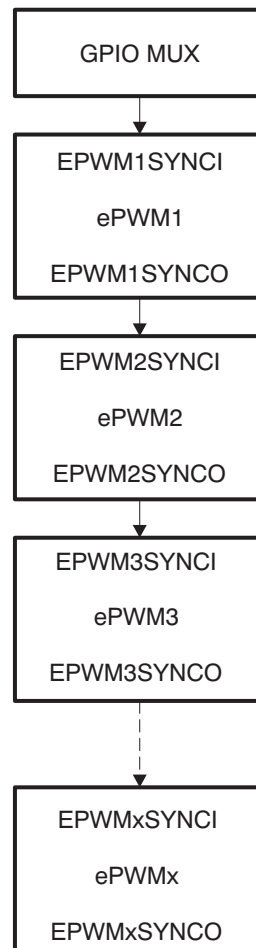
The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[PRDL] bit. This bit enables and disables the TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:** The TBPRD shadow register is enabled when TBCTL[PRDL] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active) ← TBPRD (shadow)) when the time-base counter equals zero (TBCNT = 0000h). By default the TBPRD shadow register is enabled.
- **Time-Base Period Immediate Load Mode:** If immediate load mode is selected (TBCTL[PRDL] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

### 14.2.3.3.2 Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the ePWM modules on a device. Each ePWM module has a synchronization input (EPWMxSYNCI) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (ePWM1) comes from an external pin. The possible synchronization connections for the remaining ePWM modules is shown in [Figure 14-7](#).

**Figure 14-7. Time-Base Counter Synchronization Scheme 1**



Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCNT) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCl: Synchronization Input Pulse:** The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCNT). This operation occurs on the next valid time-base clock (TBCLK) edge.
- **Software Forced Synchronization Pulse:** Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCl.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The TBPHS bit is ignored in count-up or count-down modes. See [Figure 14-8](#) through [Figure 14-11](#) for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, you can set up a master time-base (for example, ePWM1) and downstream modules (ePWM2 - ePWMx) may elect to run in synchronization with the master.

#### 14.2.3.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit in the chip configuration register 1 (CFGCHIP1) in the System Module can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. The TBCLKSYNC bit is part of the chip configuration registers and is described in the device-specific data manual. When TBCLKSYNC = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKSYNC = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is as follows:

1. Enable the ePWM module clocks.
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

#### 14.2.3.5 Time-Base Counter Modes and Timing Waveforms

The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical.
- Down-count mode which is asymmetrical.
- Up-down-count which is symmetrical.
- Frozen where the time-base counter is held constant at the current value.

To illustrate the operation of the first three modes, [Figure 14-8](#) to [Figure 14-11](#) show when events are generated and how the time-base responds to an EPWMxSYNCl signal.

Figure 14-8. Time-Base Up-Count Mode Waveforms

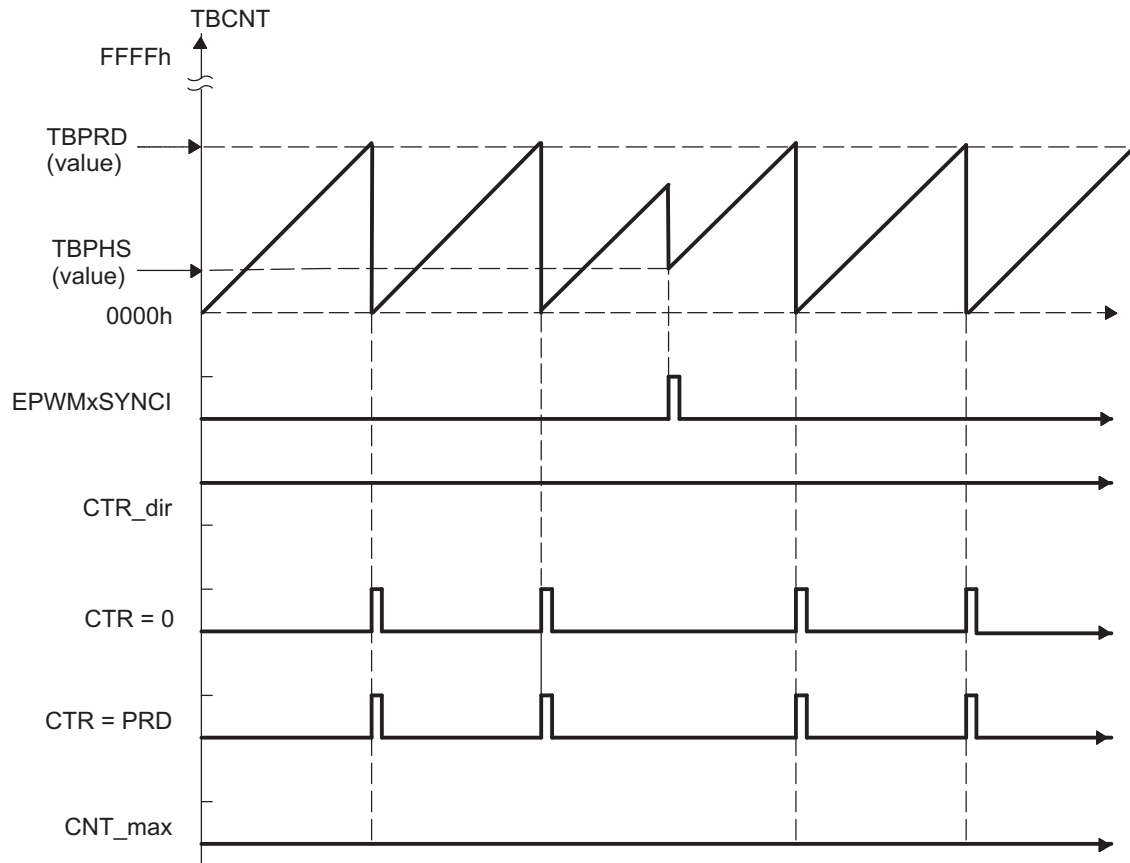


Figure 14-9. Time-Base Down-Count Mode Waveforms

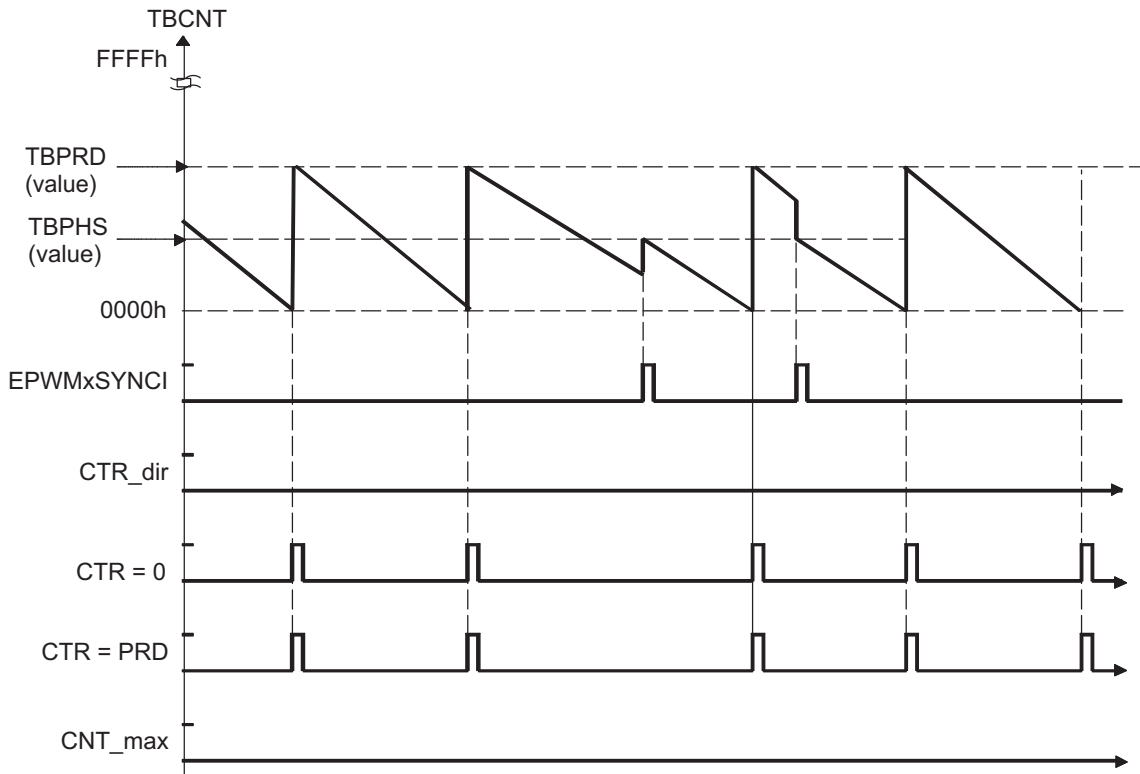
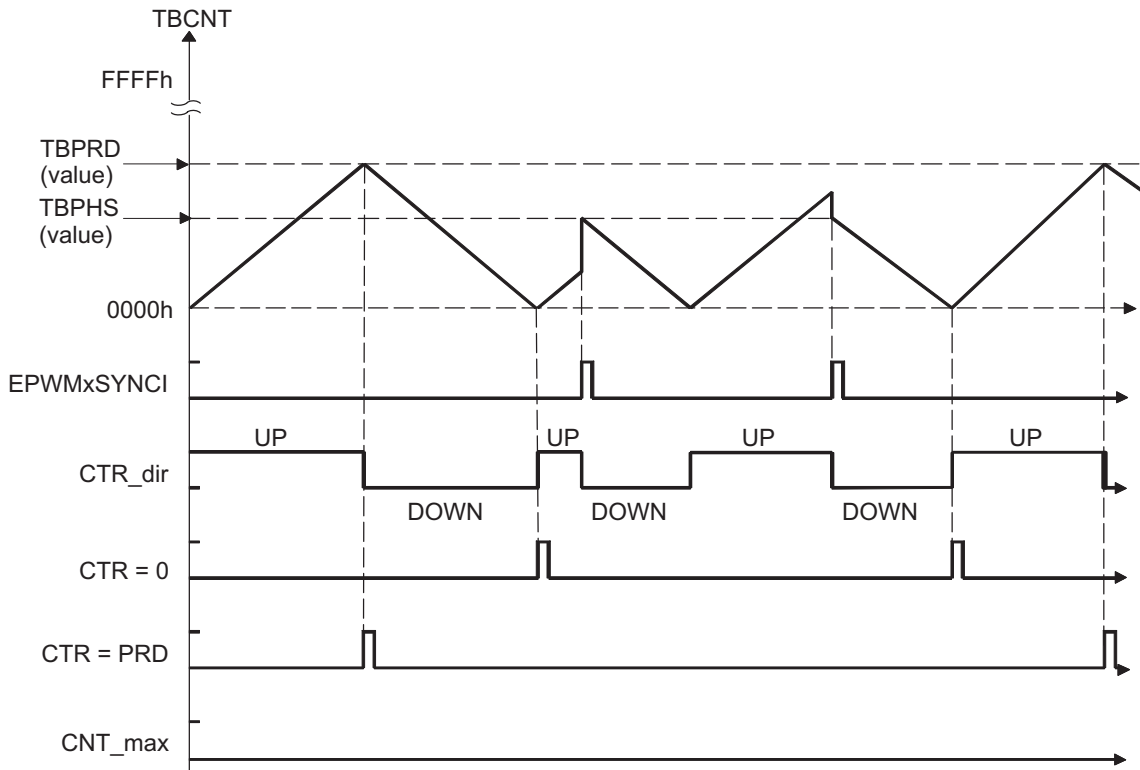
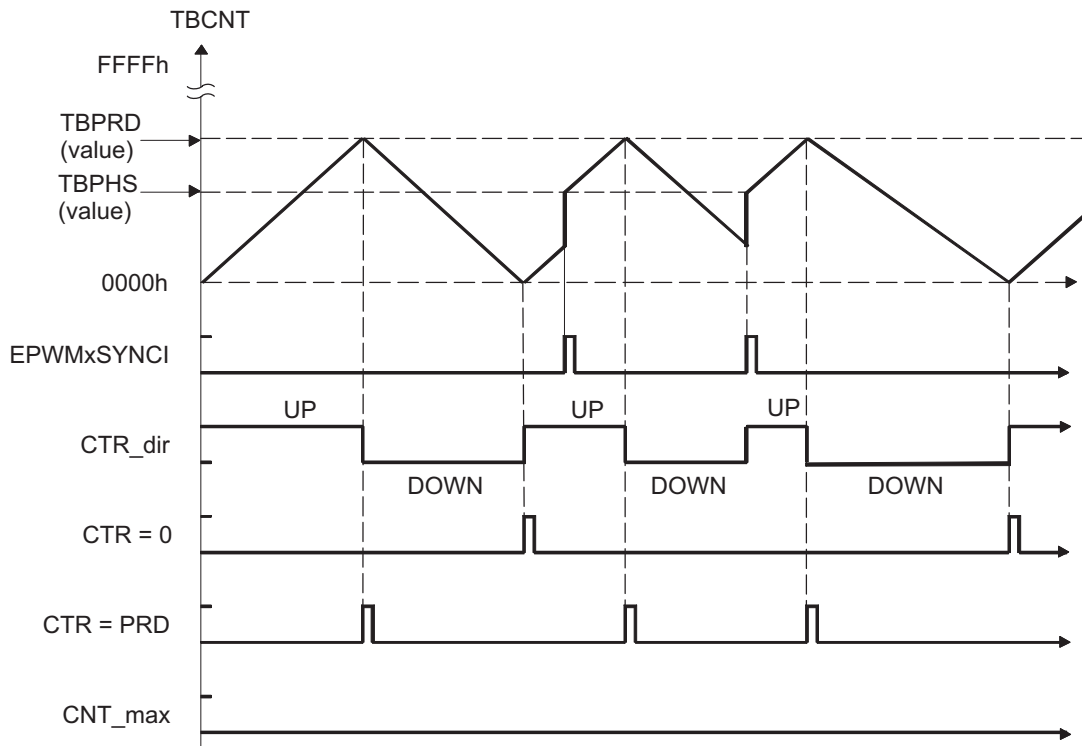


Figure 14-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down on Synchronization Event





**Figure 14-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up on Synchronization Event**



### 14.2.4 Counter-Compare (CC) Submodule

Figure 14-12 illustrates the counter-compare submodule within the ePWM. Figure 14-13 shows the basic structure of the counter-compare submodule.

Figure 14-12. Counter-Compare Submodule

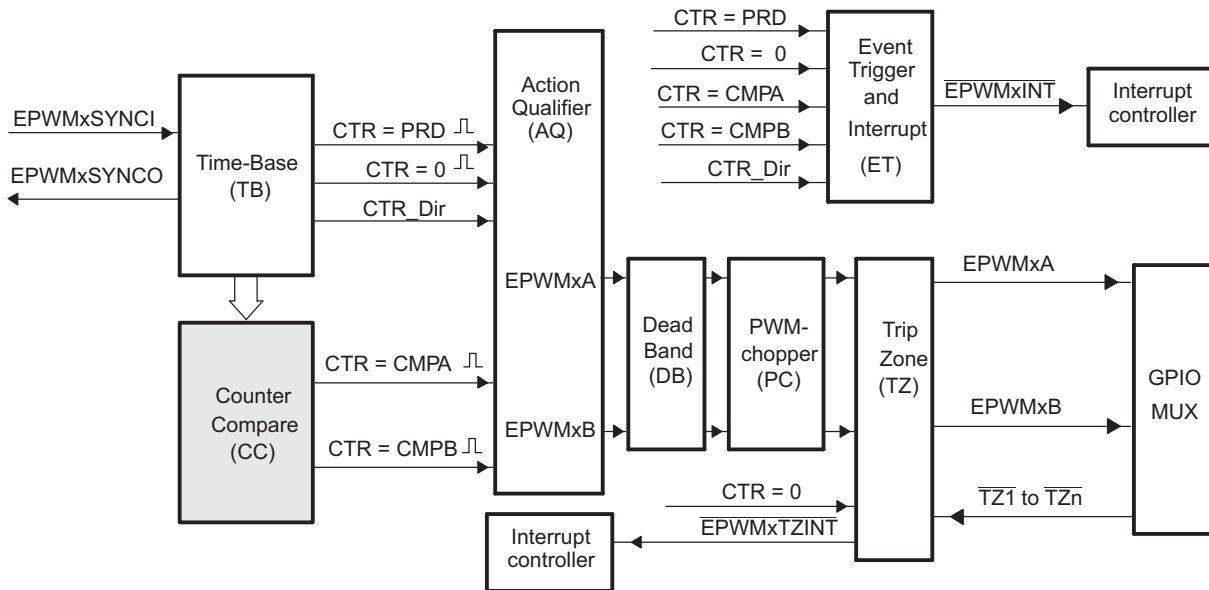
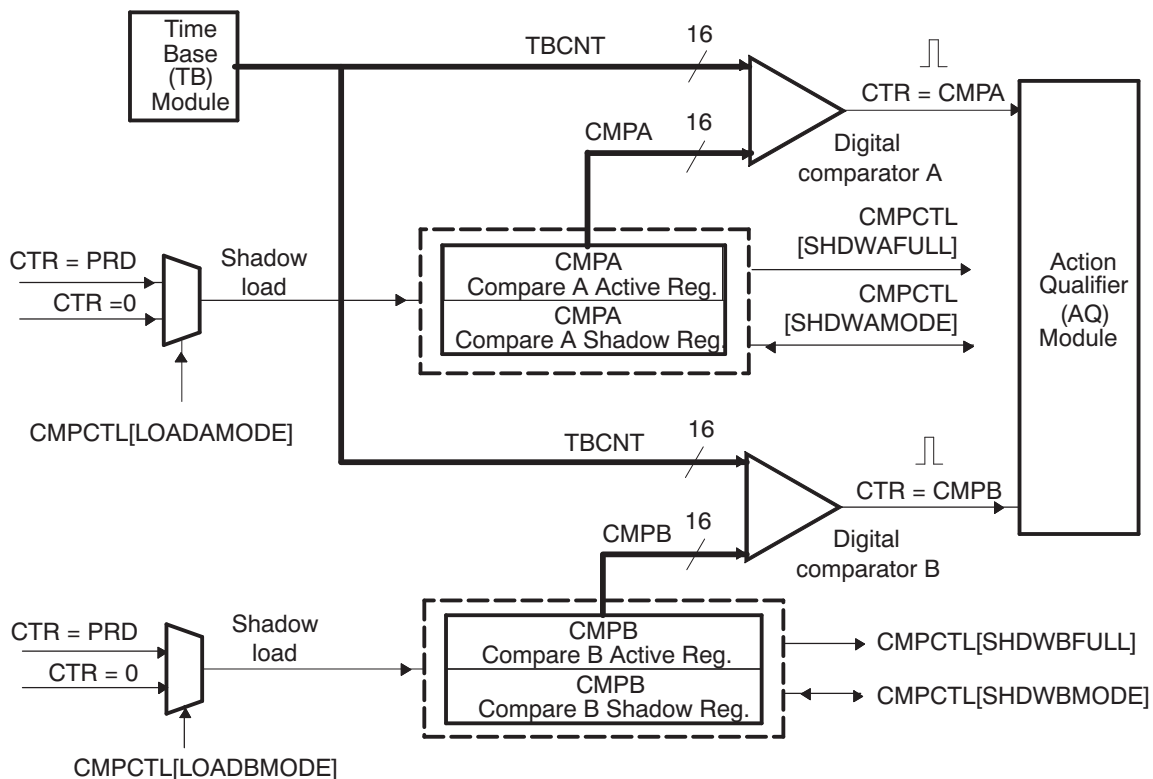


Figure 14-13. Counter-Compare Submodule Signals and Registers



#### 14.2.4.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) and counter-compare B (CMPB) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

The counter-compare submodule:

- Generates events based on programmable time stamps using the CMPA and CMPB registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCNT = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCNT = CMPB)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

#### 14.2.4.2 Controlling and Monitoring the Counter-Compare Submodule

Table 14-5 lists the registers used to control and monitor the counter-compare submodule. Table 14-6 lists the key signals associated with the counter-compare submodule.

**Table 14-5. Counter-Compare Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
CMPCTL	Counter-Compare Control Register.	Eh	No
CMPAHR	HRPWM Counter-Compare A Extension Register <sup>(1)</sup>	10h	Yes
CMPA	Counter-Compare A Register	12h	Yes
CMPB	Counter-Compare B Register	14h	Yes

<sup>(1)</sup> This register is available only on ePWM modules with the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM, this location is reserved. Refer to the device-specific data manual to determine which ePWM instances include this feature.

**Table 14-6. Counter-Compare Submodule Key Signals**

Signal	Description of Event	Registers Compared
CTR = CMPA	Time-base counter equal to the active counter-compare A value	TBCNT = CMPA
CTR = CMPB	Time-base counter equal to the active counter-compare B value	TBCNT = CMPB
CTR = PRD	Time-base counter equal to the active period. Used to load active counter-compare A and B registers from the shadow register	TBCNT = TBPRD
CTR = 0	Time-base counter equal to zero. Used to load active counter-compare A and B registers from the shadow register	TBCNT = 0000h

#### 14.2.4.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCNT = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCNT = CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle, if the compare value is between 0000h and TBPRD; and occurs once per cycle, if the compare value is equal to 0000h or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to [Section 14.2.5.1](#) for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occurs at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

- **Shadow Mode:** The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events:

- CTR = PRD: Time-base counter equal to the period (TBCNT = TBPRD).
- CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
- Both CTR = PRD and CTR = 0

Which of these three events is specified by the CMPCTL[LOADAMODE] and CMPCTL[LOADBMODE] register bits. Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

- **Immediate Load Mode:** If immediate load mode is selected (TBCTL[SHADWAMODE] = 1 or TBCTL[SHADWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

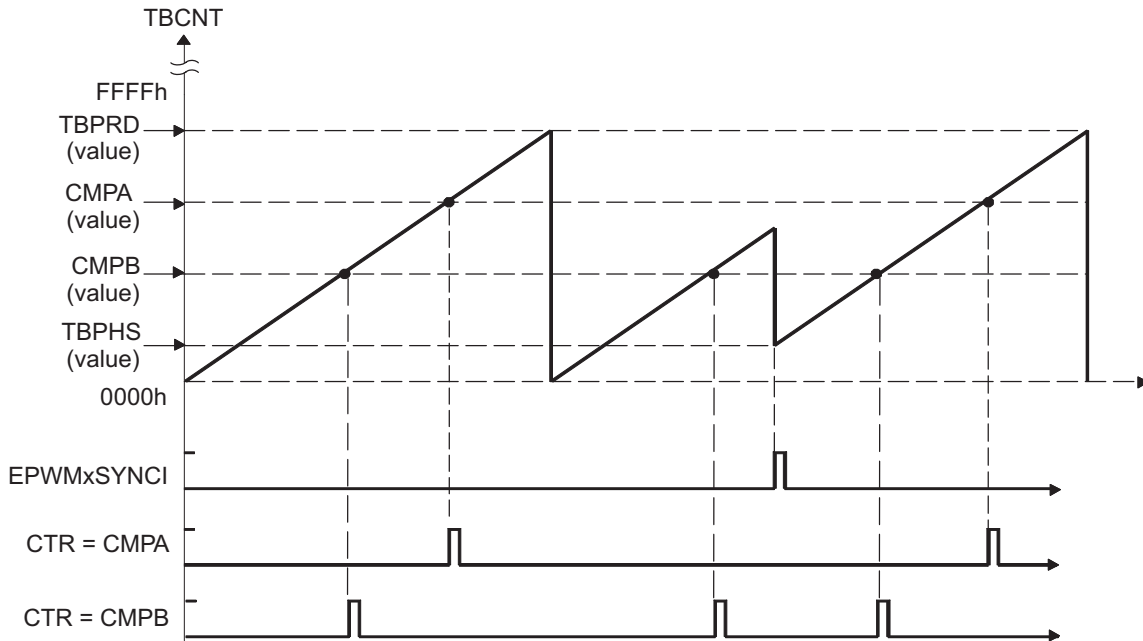
#### 14.2.4.4 Count Mode Timing Waveforms

The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

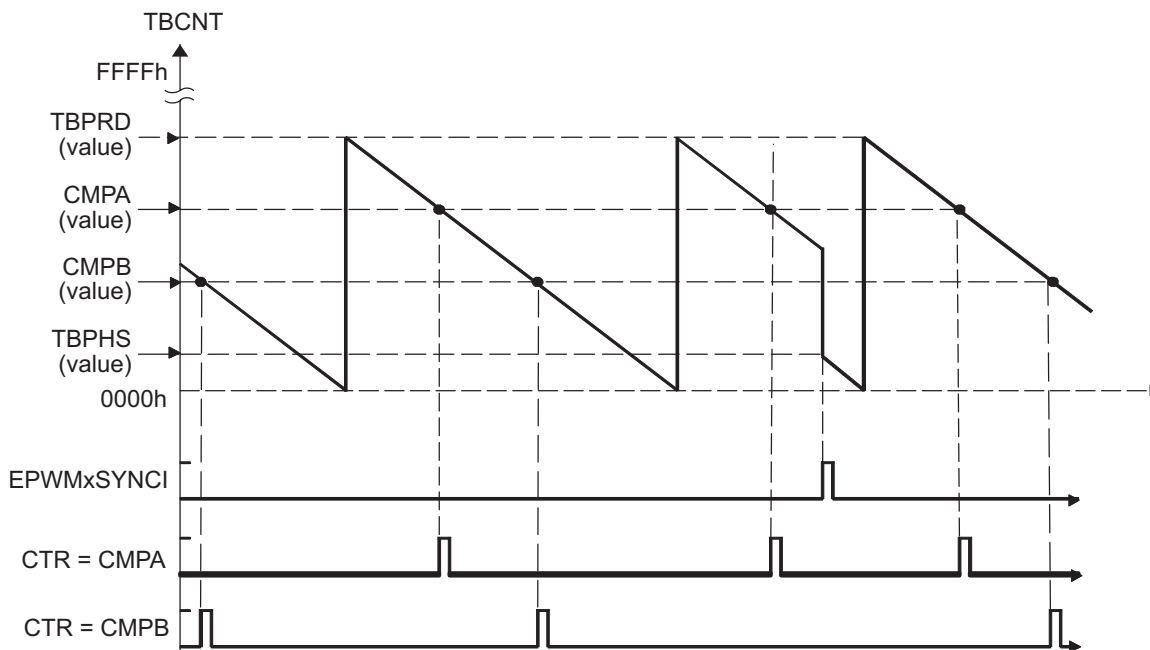
To best illustrate the operation of the first three modes, the timing diagrams in [Figure 14-14](#) to [Figure 14-17](#) show when events are generated and how the EPWMxSYNCl signal interacts.

**Figure 14-14. Counter-Compare Event Waveforms in Up-Count Mode**

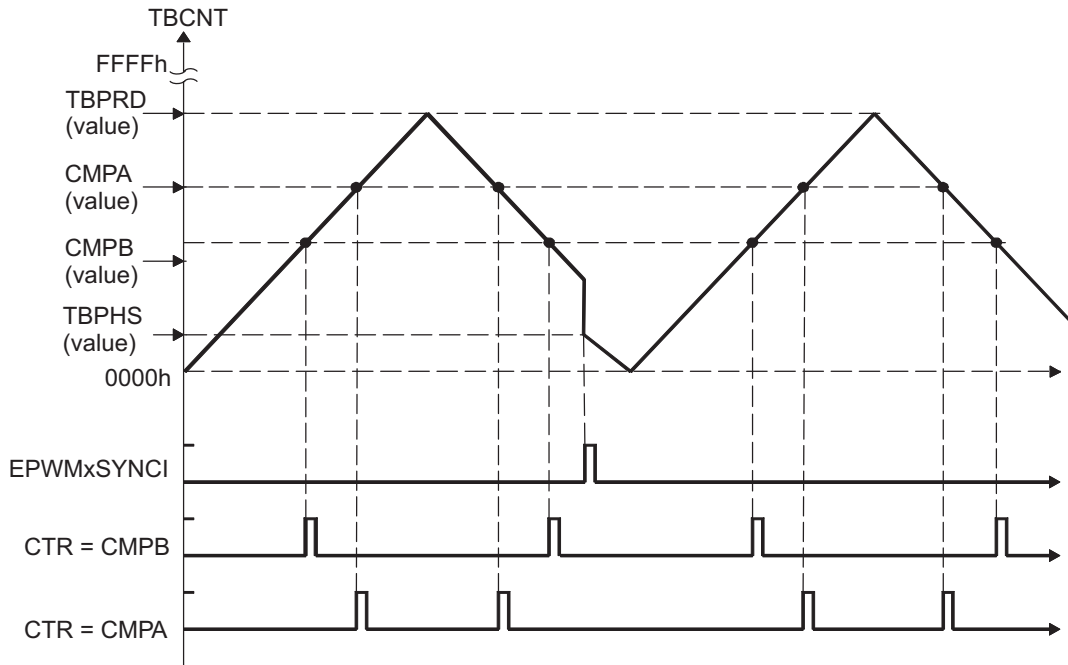


NOTE: An EPWMxSYNCl external synchronization event can cause a discontinuity in the TBCNT count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

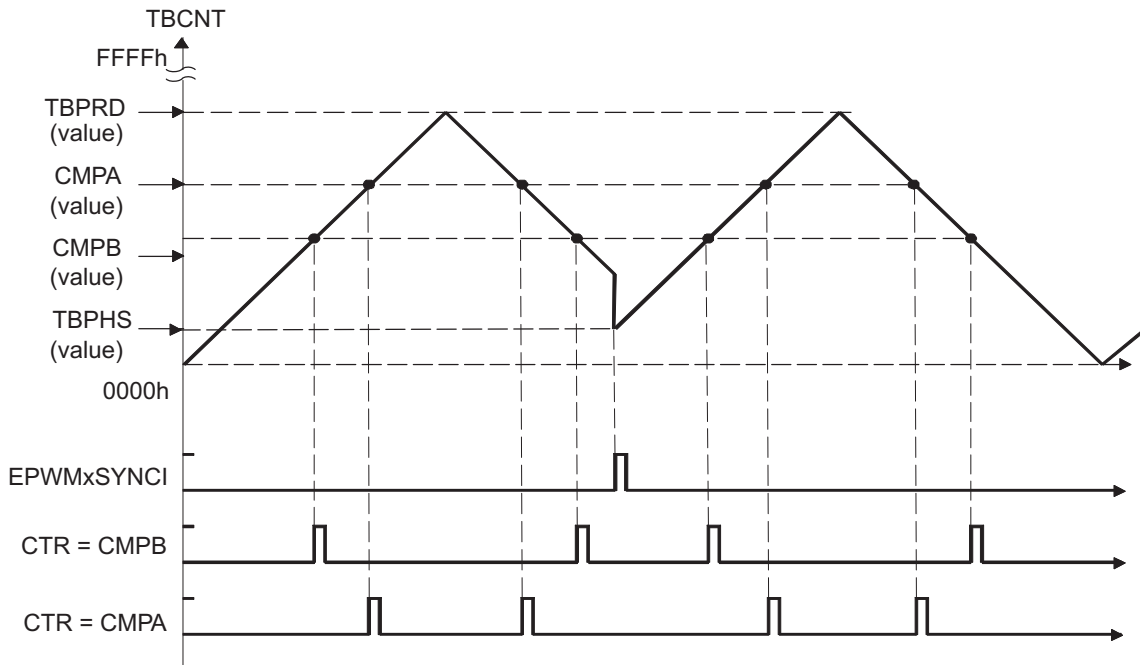
**Figure 14-15. Counter-Compare Events in Down-Count Mode**



**Figure 14-16. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down on Synchronization Event**



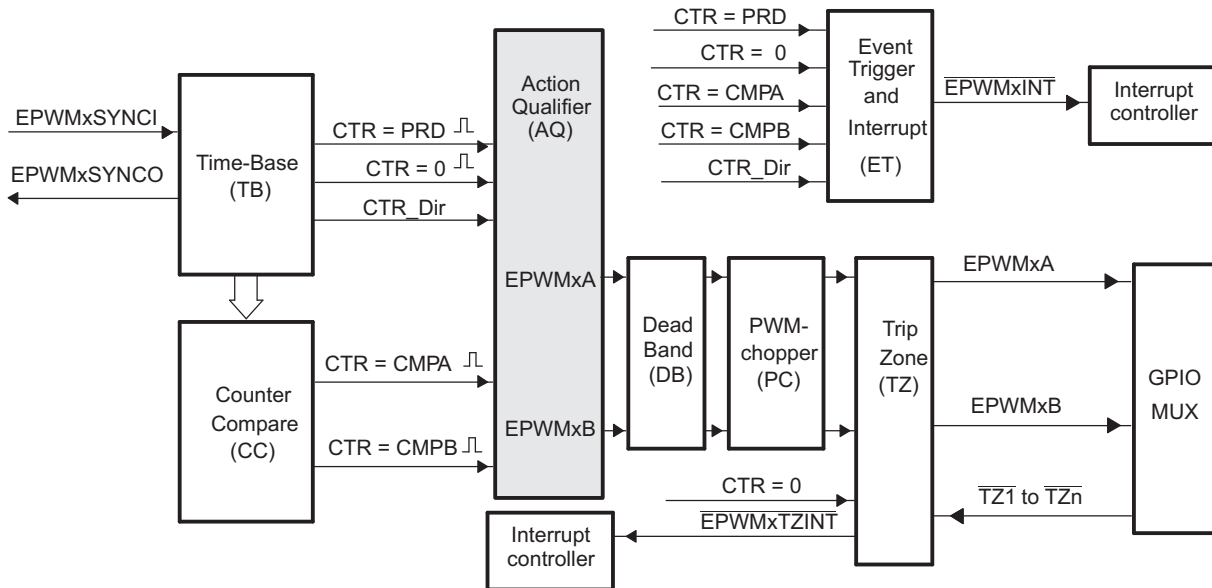
**Figure 14-17. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up on Synchronization Event**



### 14.2.5 Action-Qualifier (AQ) Submodule

Figure 14-18 shows the action-qualifier (AQ) submodule (see shaded block) in the ePWM system. The action-qualifier submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

**Figure 14-18. Action-Qualifier Submodule**



#### 14.2.5.1 Purpose of the Action-Qualifier Submodule

The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCNT = TBPRD)
  - CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCNT = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCNT = CMPB)
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing.

#### 14.2.5.2 Controlling and Monitoring the Action-Qualifier Submodule

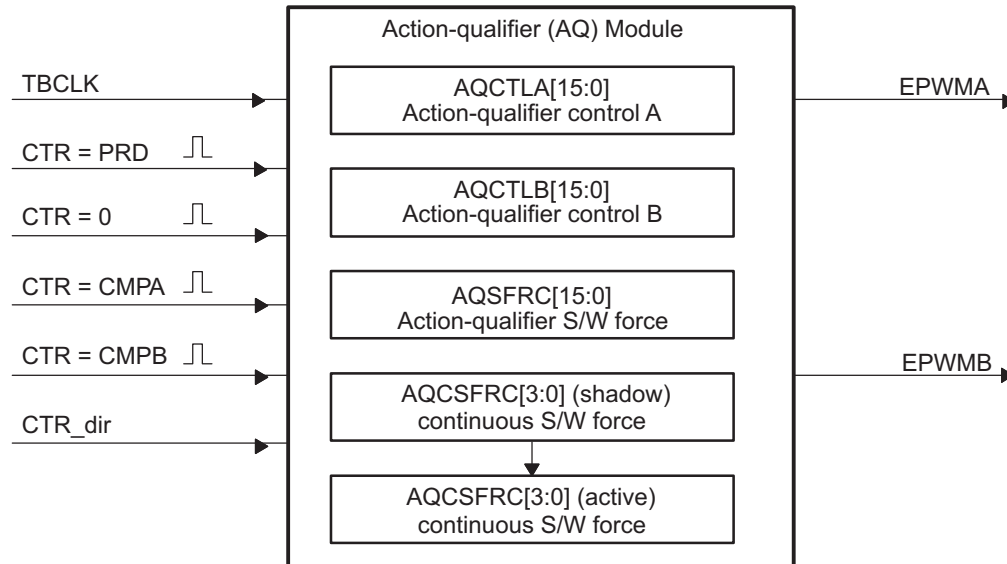
Table 14-7 lists the registers used to control and monitor the action-qualifier submodule.

**Table 14-7. Action-Qualifier Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
AQCTLA	Action-Qualifier Control Register For Output A (EPWMxA)	16h	No
AQCTLB	Action-Qualifier Control Register For Output B (EPWMxB)	18h	No
AQSFRC	Action-Qualifier Software Force Register	1Ah	No
AQCSFRC	Action-Qualifier Continuous Software Force	1Ch	Yes

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers shown in [Figure 14-19](#). The possible input events are summarized again in [Table 14-8](#).

**Figure 14-19. Action-Qualifier Submodule Inputs and Outputs**



**Table 14-8. Action-Qualifier Submodule Possible Input Events**

Signal	Description	Registers Compared
CTR = PRD	Time-base counter equal to the period value	TBCNT = TBPRD
CTR = 0	Time-base counter equal to zero	TBCNT = 0000h
CTR = CMPA	Time-base counter equal to the counter-compare A	TBCNT = CMPA
CTR = CMPB	Time-base counter equal to the counter-compare B	TBCNT = CMPB
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by registers AQSFRC and AQCSFRC.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:





















- **Set High:** Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:** Set output EPWMxA or EPWMxB to a low level.
- **Toggle:** If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:** Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts. See the event-trigger submodule description in [Section 14.2.9](#) for details.



Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this chapter use a set of symbolic actions. These symbols are summarized in [Figure 14-20](#). Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

**Figure 14-20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
					Do Nothing
					Clear Low
					Set High
					Toggle

### 14.2.5.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in [Table 14-9](#). A priority level of 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCNT.

**Table 14-9. Action-Qualifier Event Priority for Up-Down-Count Mode**

Priority Level	Event if TBCNT is Incrementing TBCNT = 0 up to TBCNT = TBPRD	Event if TBCNT is Decrementing TBCNT = TBPRD down to TBCNT = 1
1 (Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals zero	Counter equals period (TBPRD)
5	Counter equals CMPB on down-count (CBD) <sup>(1)</sup>	Counter equals CMPB on up-count (CBU) <sup>(1)</sup>
6 (Lowest)	Counter equals CMPA on down-count (CAD) <sup>(1)</sup>	Counter equals CMPA on up-count (CBU) <sup>(1)</sup>

<sup>(1)</sup> To maintain symmetry for up-down-count mode, both up-events (CAU/CBU) and down-events (CAD/CBD) can be generated for TBPRD. Otherwise, up-events can occur only when the counter is incrementing and down-events can occur only when the counter is decrementing.

[Table 14-10](#) shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

**Table 14-10. Action-Qualifier Event Priority for Up-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to period (TBPRD)
3	Counter equal to CMPB on up-count (CBU)
4	Counter equal to CMPA on up-count (CAU)
5 (Lowest)	Counter equal to Zero

[Table 14-11](#) shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

**Table 14-11. Action-Qualifier Event Priority for Down-Count Mode**

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (Lowest)	Counter equal to period (TBPRD)

It is possible to set the compare value greater than the period. In this case the action will take place as shown in [Table 14-12](#).

**Table 14-12. Behavior if CMPA/CMPB is Greater than the Period**

Counter Mode	Compare on Up-Count Event CAU/CBU	Compare on Down-Count Event CAU/CBU
Up-Count Mode	<p>If <math>CMPA/CMPB \leq TBPRD</math> period, then the event occurs on a compare match (<math>TBCNT = CMPA</math> or <math>CMPB</math>).</p> <p>If <math>CMPA/CMPB &gt; TBPRD</math>, then the event will not occur.</p>	Never occurs.
Down-Count Mode	Never occurs.	<p>If <math>CMPA/CMPB &lt; TBPRD</math>, the event will occur on a compare match (<math>TBCNT = CMPA</math> or <math>CMPB</math>).</p> <p>If <math>CMPA/CMPB \geq TBPRD</math>, the event will occur on a period match (<math>TBCNT = TBPRD</math>).</p>
Up-Down-Count Mode	<p>If <math>CMPA/CMPB &lt; TBPRD</math> and the counter is incrementing, the event occurs on a compare match (<math>TBCNT = CMPA</math> or <math>CMPB</math>).</p> <p>If <math>CMPA/CMPB \geq TBPRD</math>, the event will occur on a period match (<math>TBCNT = TBPRD</math>).</p>	<p>If <math>CMPA/CMPB &lt; TBPRD</math> and the counter is decrementing, the event occurs on a compare match (<math>TBCNT = CMPA</math> or <math>CMPB</math>).</p> <p>If <math>CMPA/CMPB \geq TBPRD</math>, the event occurs on a period match (<math>TBCNT = TBPRD</math>).</p>

#### 14.2.5.4 Waveforms for Common Configurations

**NOTE:** The waveforms in this chapter show the ePWMs behavior for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

**Use up-down-count mode to generate a symmetric PWM:**

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to  $TBPRD - 1$ .

This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

**Use up-down-count mode to generate an asymmetric PWM:**

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

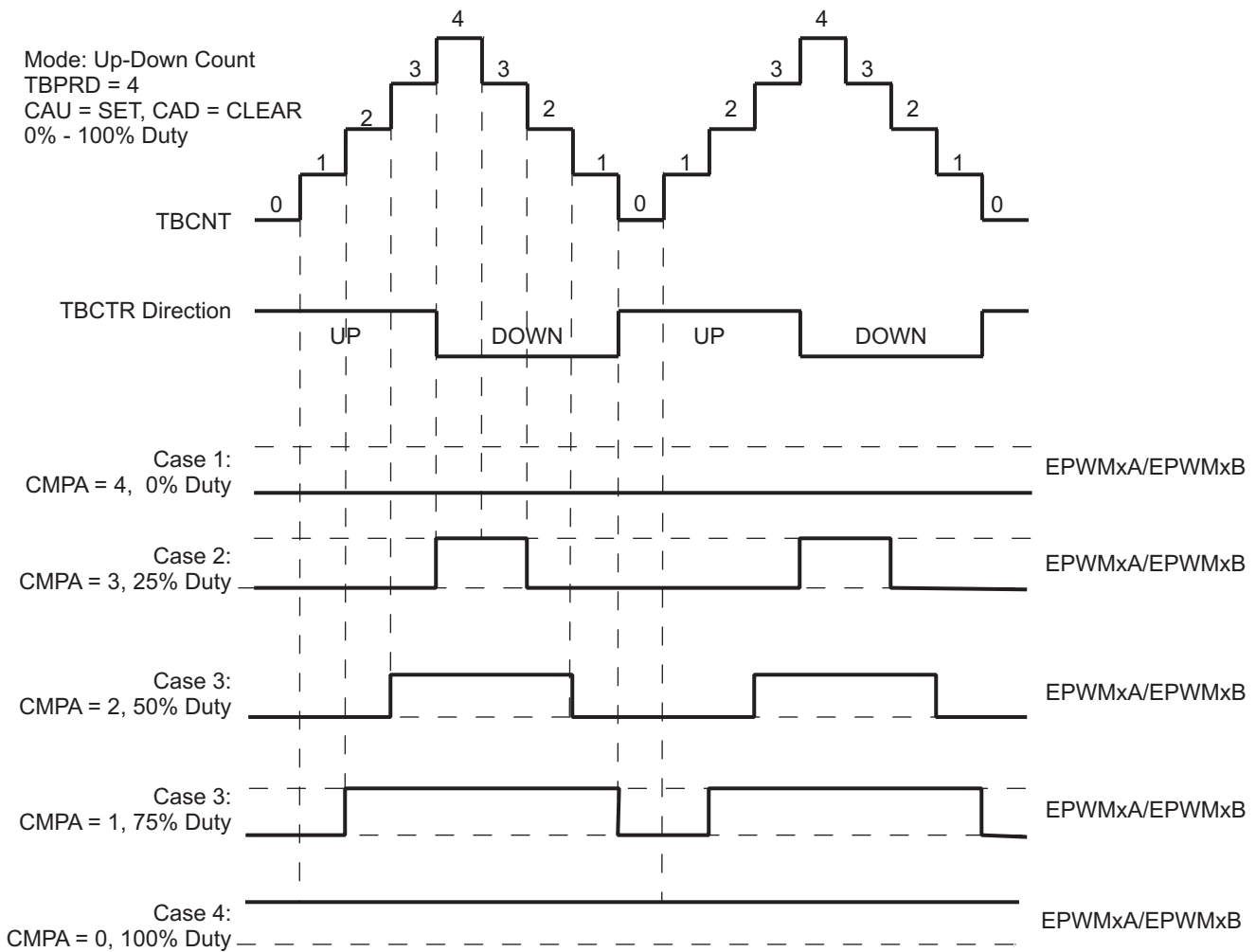
**When using up-count mode to generate an asymmetric PWM:**

- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to  $TBPRD+1$  to achieve 0-100% PWM duty.

Figure 14-21 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCNT. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When  $CMPA = 0$ , the PWM signal is low for the entire period giving the 0% duty waveform. When  $CMPA = TBPRD$ , the PWM signal is high achieving 100% duty.

When using this configuration in practice, if you load  $CMPA/CMPB$  on zero, then use  $CMPA/CMPB$  values greater than or equal to 1. If you load  $CMPA/CMPB$  on period, then use  $CMPA/CMPB$  values less than or equal to  $TBPRD-1$ . This means there will always be a pulse of at least one  $TBCLK$  cycle in a PWM period which, when very short, tend to be ignored by the system.

Figure 14-21. Up-Down-Count Mode Symmetrical Waveform

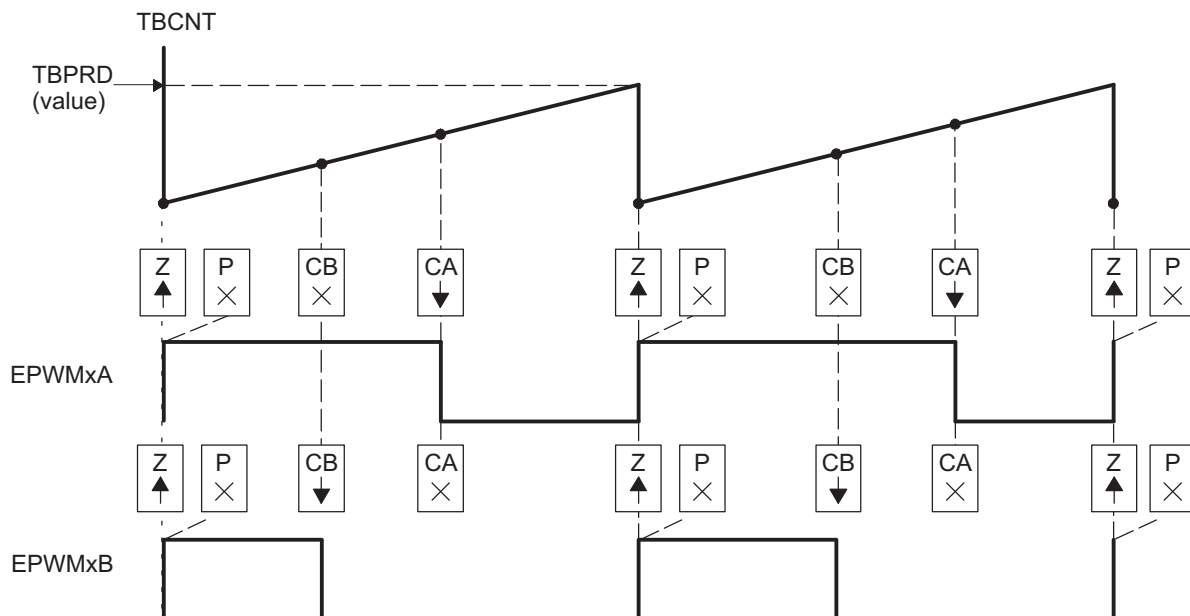


The PWM waveforms in [Figure 14-22](#) through [Figure 14-27](#) show some common action-qualifier configurations. Some conventions used in the figures are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

[Table 14-13](#) and [Table 14-14](#) contains initialization and runtime register configurations for the waveforms in [Figure 14-22](#).

**Figure 14-22. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High**



- (1)  $\text{PWM period} = (\text{TBPRD} + 1) \times T_{\text{TBCLK}}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
- (3) Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
- (4) The "Do Nothing" actions ( X ) are shown for completeness, but will not be shown on subsequent diagrams.
- (5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

**Table 14-13. EPWMx Initialization for Figure 14-22**

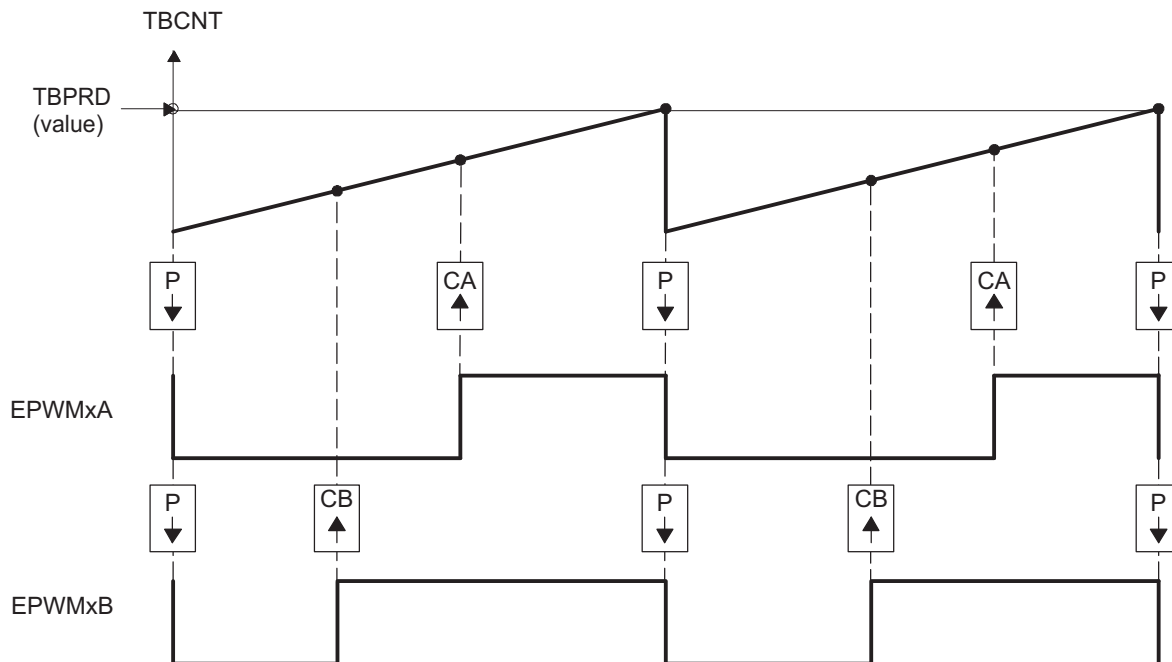
Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	
	CAU	AQ_CLEAR	
AQCTLB	ZRO	AQ_SET	
	CBU	AQ_CLEAR	

**Table 14-14. EPWMx Run Time Changes for Figure 14-22**

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 14-15 and Table 14-16 contains initialization and runtime register configurations for the waveforms in Figure 14-23.

**Figure 14-23. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low**



- (1)  $\text{PWM period} = (\text{TBPRD} + 1) \times T_{\text{TBCLK}}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- (3) Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- (4) The Do Nothing actions ( X ) are shown for completeness here, but will not be shown on subsequent diagrams.
- (5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

**Table 14-15. EPWMx Initialization for Figure 14-23**

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	
AQCTLB	PRD	AQ_CLEAR	
	CBU	AQ_SET	

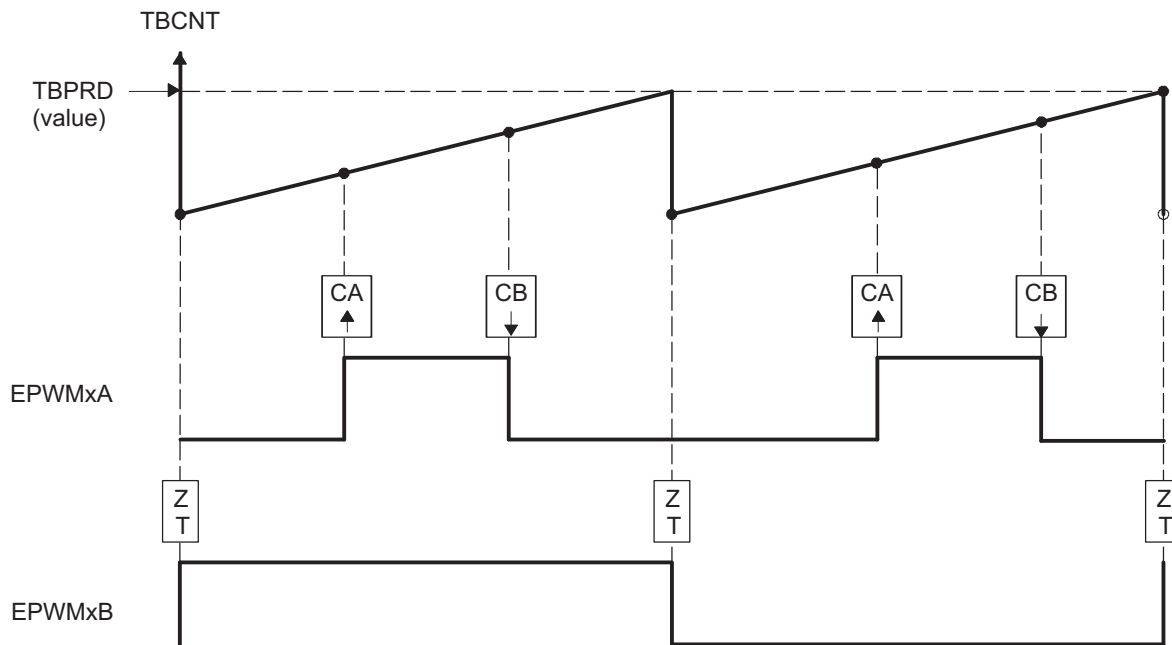
**Table 14-16. EPWMx Run Time Changes for Figure 14-23**

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B



Table 14-17 and Table 14-18 contains initialization and runtime register configurations for the waveforms Figure 14-24. Use the code in Example 14-1 to define the headers.

**Figure 14-24. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**



- (1)  $\text{PWM frequency} = 1 / ((\text{TBPRD} + 1) \times T_{\text{TBCLK}})$
- (2) Pulse can be placed anywhere within the PWM cycle (0000h - TBPRD)
- (3) High time duty proportional to (CMPB - CMPA)
- (4) EPWMxB can be used to generate a 50% duty square wave with frequency =  $1/2 \times ((\text{TBPRD} + 1) \times \text{TBCLK})$

**Table 14-17. EPWMx Initialization for Figure 14-24**

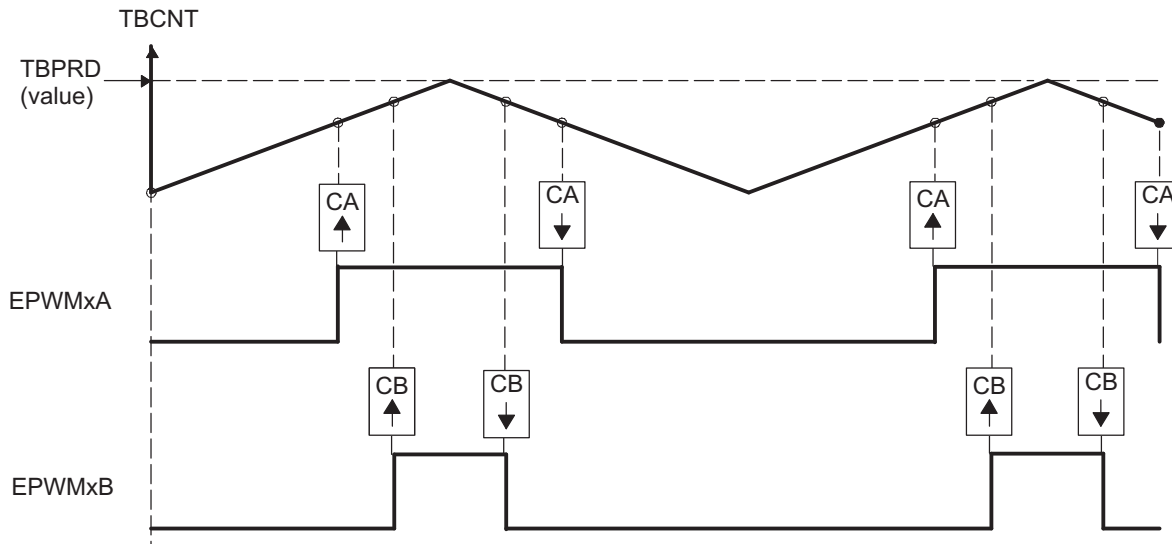
Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	200 (C8h)	Compare A = 200 TBCLK counts
CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CBU	AQ_CLEAR	
AQCTLB	ZRO	AQ_TOGGLE	

**Table 14-18. EPWMx Run Time Changes for Figure 14-24**

Register	Bit	Value	Comments
CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
CMPB	CMPB	EdgePosB	

Table 14-19 and Table 14-20 contains initialization and runtime register configurations for the waveforms in Figure 14-25. Use the code in Example 14-1 to define the headers.

**Figure 14-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low**



- (1)  $\text{PWM period} = 2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- (3) Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- (4) Outputs EPWMxA and EPWMxB can drive independent power switches

**Table 14-19. EPWMx Initialization for Figure 14-25**

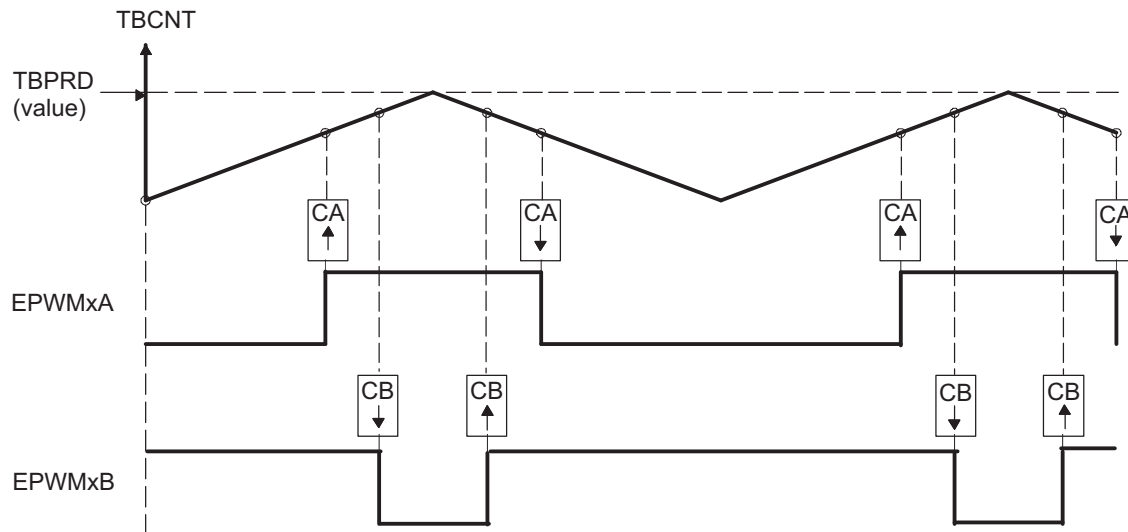
Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	400 (190h)	Compare A = 400 TBCLK counts
CMPB	CMPB	500 (1F4h)	Compare B = 500 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_SET	
	CBD	AQ_CLEAR	

**Table 14-20. EPWMx Run Time Changes for Figure 14-25**

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 14-21 and Table 14-22 contains initialization and runtime register configurations for the waveforms in Figure 14-26. Use the code in Example 14-1 to define the headers.

**Figure 14-26. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary**



- (1)  $PWM\ period = 2 \times TBPRD \times T_{TBCLK}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active low, i.e., low time duty proportional to CMPA
- (3) Duty modulation for EPWMxB is set by CMPB and is active high, i.e., high time duty proportional to CMPB
- (4) Outputs EPWMx can drive upper/lower (complementary) power switches
- (5) Dead-band =  $CMPB - CMPA$  (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

**Table 14-21. EPWMx Initialization for Figure 14-26**

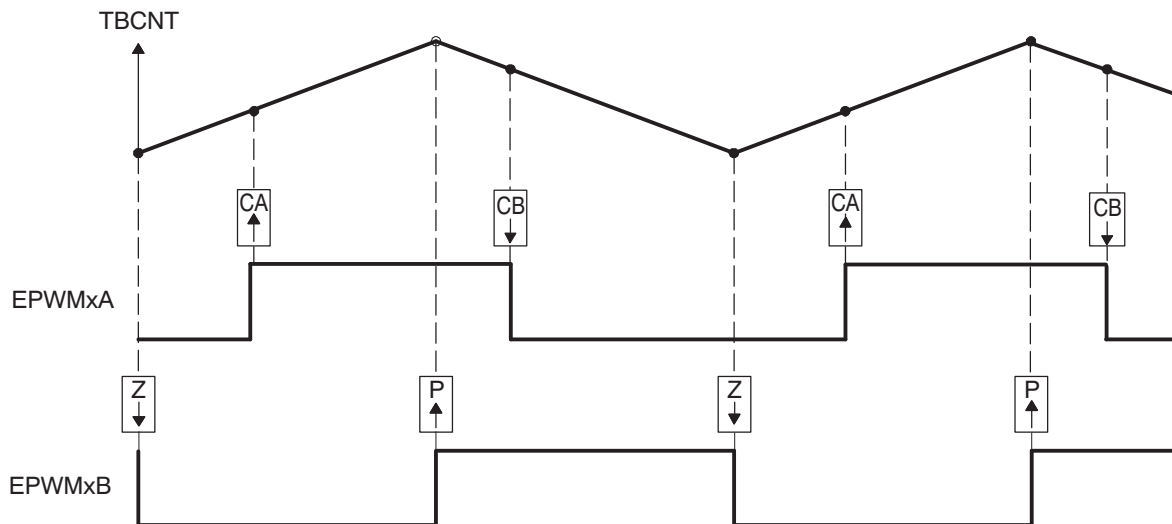
Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_CLEAR	
	CBD	AQ_SET	

**Table 14-22. EPWMx Run Time Changes for Figure 14-26**

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Table 14-23 and Table 14-24 contains initialization and runtime register configurations for the waveforms in Figure 14-27. Use the code in Example 14-1 to define the headers.

**Figure 14-27. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low**



- (1) PWM period =  $2 \times \text{TBPRD} \times \text{TBCLK}$
- (2) Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
- (3) Duty modulation for EPWMxA is set by CMPA and CMPB.
- (4) Low time duty for EPWMxA is proportional to  $(\text{CMPA} + \text{CMPB})$ .
- (5) To change this example to active high, CMPA and CMPB actions need to be inverted (i.e., Set ! Clear and Clear Set).
- (6) Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB)

**Table 14-23. EPWMx Initialization for Figure 14-27**

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	250 (FAh)	Compare A = 250 TBCLK counts
CMPB	CMPB	450 (1C2h)	Compare B = 450 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CBD	AQ_CLEAR	
AQCTLB	ZRO	AQ_CLEAR	
	PRD	AQ_SET	

**Table 14-24. EPWMx Run Time Changes for Figure 14-27**

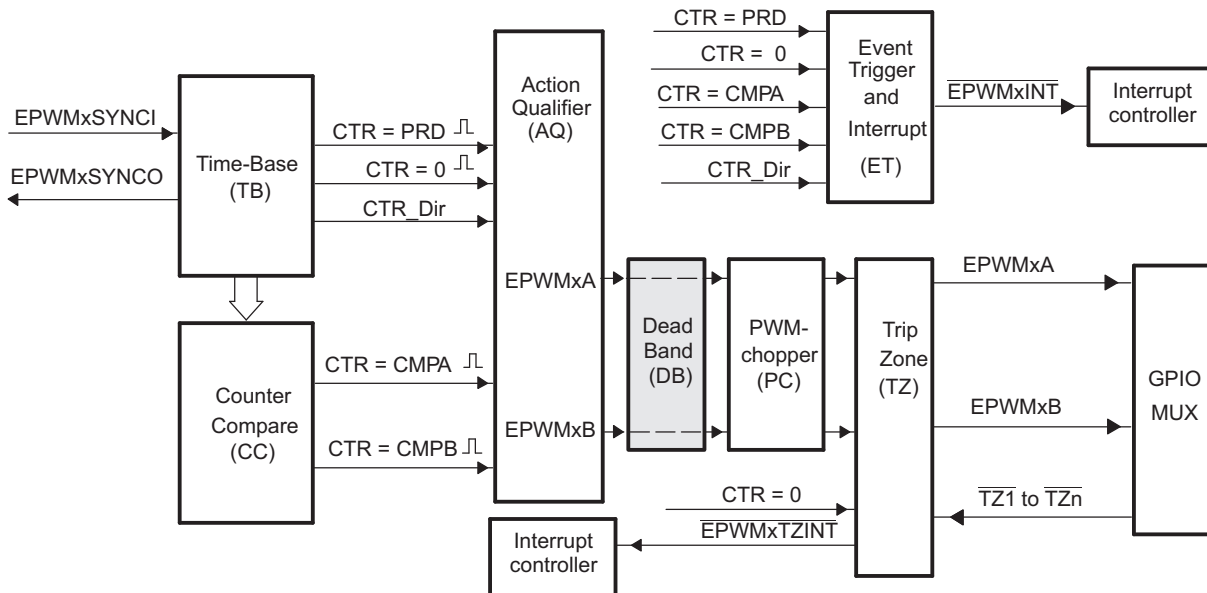
Register	Bit	Value	Comments
CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
CMPB	CMPB	EdgePosB	



## 14.2.6 Dead-Band Generator (DB) Submodule

Figure 14-28 illustrates the dead-band generator submodule within the ePWM module.

**Figure 14-28. Dead-Band Generator Submodule**



### 14.2.6.1 Purpose of the Dead-Band Submodule

The "Action-qualifier (AQ) Module" section discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band generator submodule should be used.

The key functions of the dead-band generator submodule are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in diagram)

### 14.2.6.2 Controlling and Monitoring the Dead-Band Submodule

The dead-band generator submodule operation is controlled and monitored via the following registers:

**Table 14-25. Dead-Band Generator Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
DBCTL	Dead-Band Control Register	1Eh	No
DBRED	Dead-Band Rising Edge Delay Count Register	20h	No
DBFED	Dead-Band Falling Edge Delay Count Register	22h	No

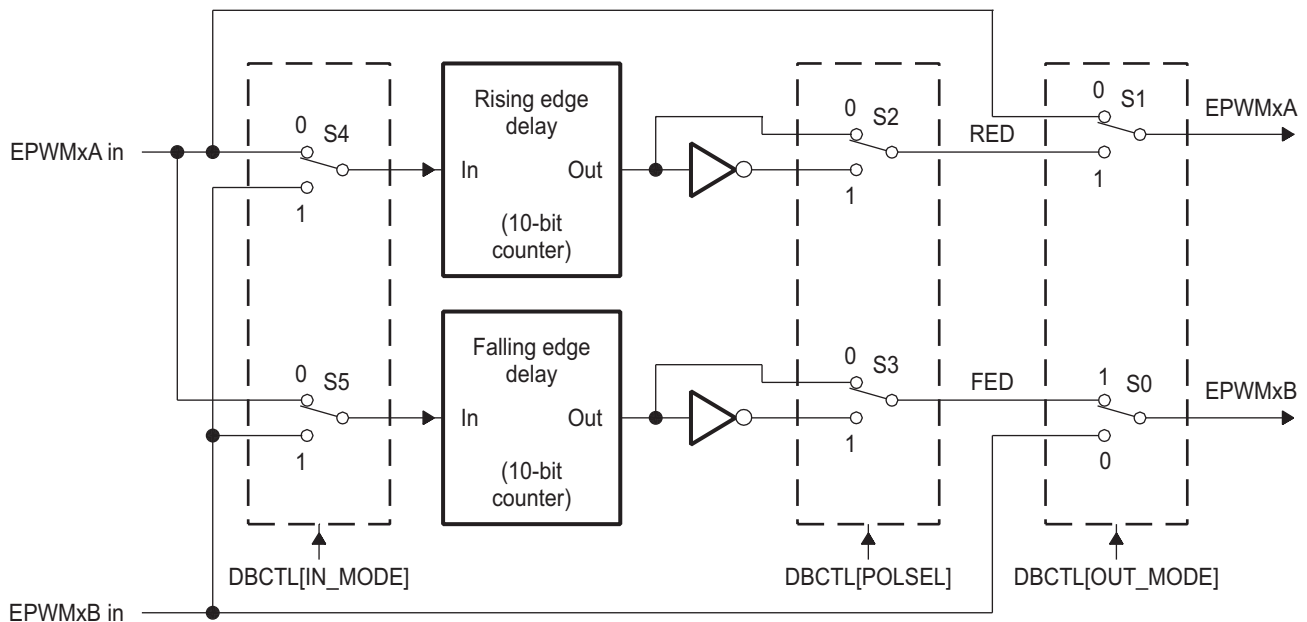
### 14.2.6.3 Operational Highlights for the Dead-Band Generator Submodule

The following sections provide the operational highlights.

The dead-band submodule has two groups of independent selection options as shown in Figure 14-29.

- Input Source Selection:** The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the DBCTL[IN\_MODE] control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:
  - EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.
  - EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.
  - EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.
  - EPWMxB In is the source for both falling-edge and rising-edge delay.
- Output Mode Control:** The output mode is configured by way of the DBCTL[OUT\_MODE] bits. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.
- Polarity Control:** The polarity control (DBCTL[POLSEL]) allows you to specify whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.

Figure 14-29. Configuration Options for the Dead-Band Generator Submodule



Although all combinations are supported, not all are typical usage modes. [Table 14-26](#) lists some classical dead-band configurations. These modes assume that the DBCTL[IN\_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in [Table 14-26](#) fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)** Allows you to fully disable the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings** These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in [Figure 14-30](#). Note that to generate equivalent waveforms to [Figure 14-30](#), configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay** Finally the last two entries in [Table 14-26](#) show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

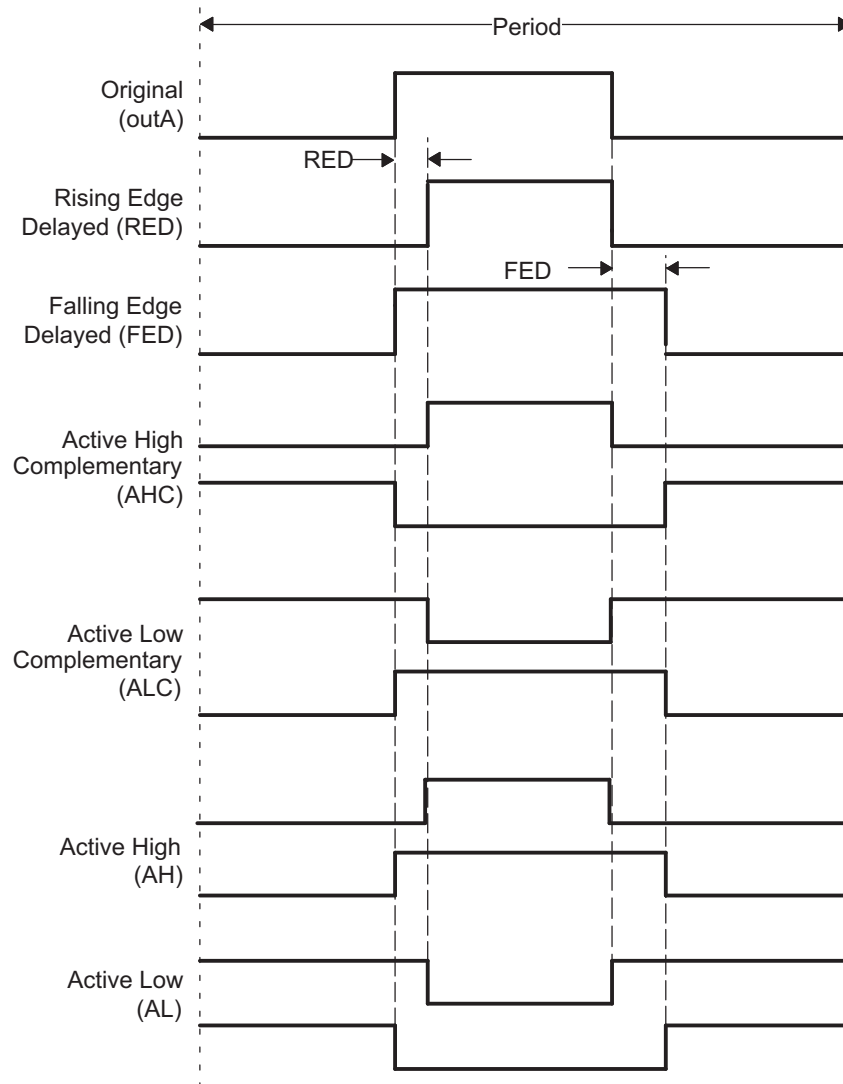
**Table 14-26. Classical Dead-Band Operating Modes**

Mode	Mode Description <sup>(1)</sup>	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	x	x	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB Out = EPWMxA In with Falling Edge Delay	0 or 1	0 or 1	0	1
7	EPWMxA Out = EPWMxA In with Rising Edge Delay EPWMxB Out = EPWMxB In with No Delay	0 or 1	0 or 1	1	0

<sup>(1)</sup> These are classical dead-band modes and assume that DBCTL[IN\_MODE] = 0,0. That is, EPWMxA in is the source for both the falling-edge and rising-edge delays. Enhanced, non-traditional modes can be achieved by changing the IN\_MODE configuration.

Figure 14-30 shows waveforms for typical cases where 0% < duty < 100%.

**Figure 14-30. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)**



The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods a signal edge is delayed by. For example, the formula to calculate falling-edge-delay and rising-edge-delay are:

$$FED = DBFED \times T_{TBCLK}$$

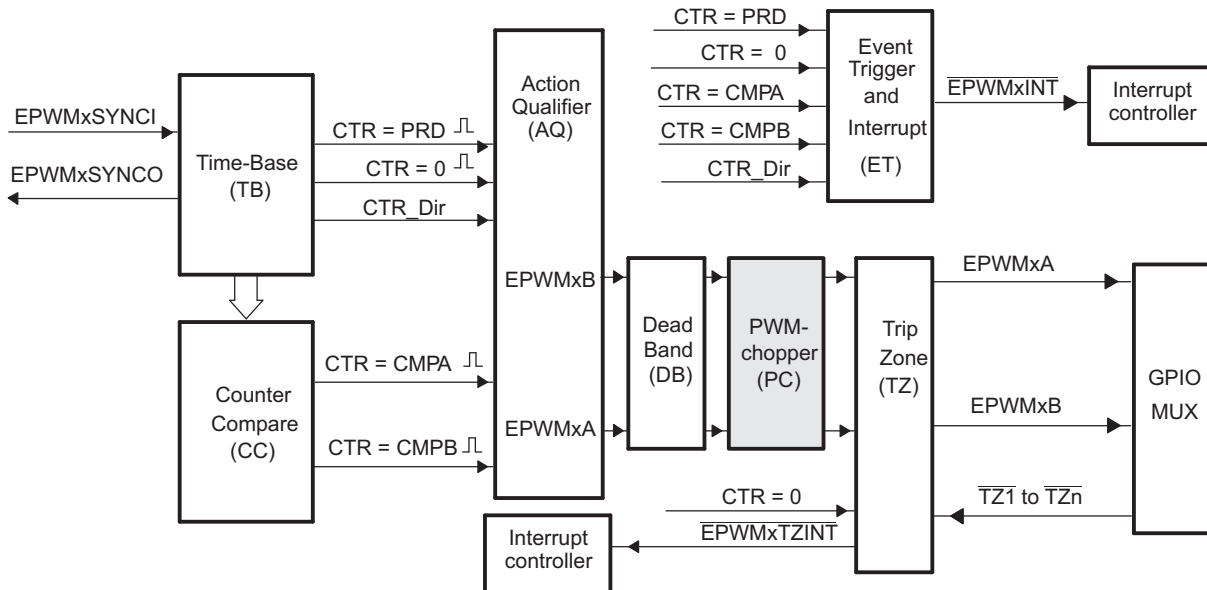
$$RED = DBRED \times T_{TBCLK}$$

Where  $T_{TBCLK}$  is the period of TBCLK, the prescaled version of SYSCLKOUT.

### 14.2.7 PWM-Chopper (PC) Submodule

Figure 14-31 illustrates the PWM-chopper (PC) submodule within the ePWM module. The PWM-chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

**Figure 14-31. PWM-Chopper Submodule**



#### 14.2.7.1 Purpose of the PWM-Chopper Submodule

The key functions of the PWM-chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

#### 14.2.7.2 Controlling the PWM-Chopper Submodule

The PWM-chopper submodule operation is controlled via the register in [Table 14-27](#).

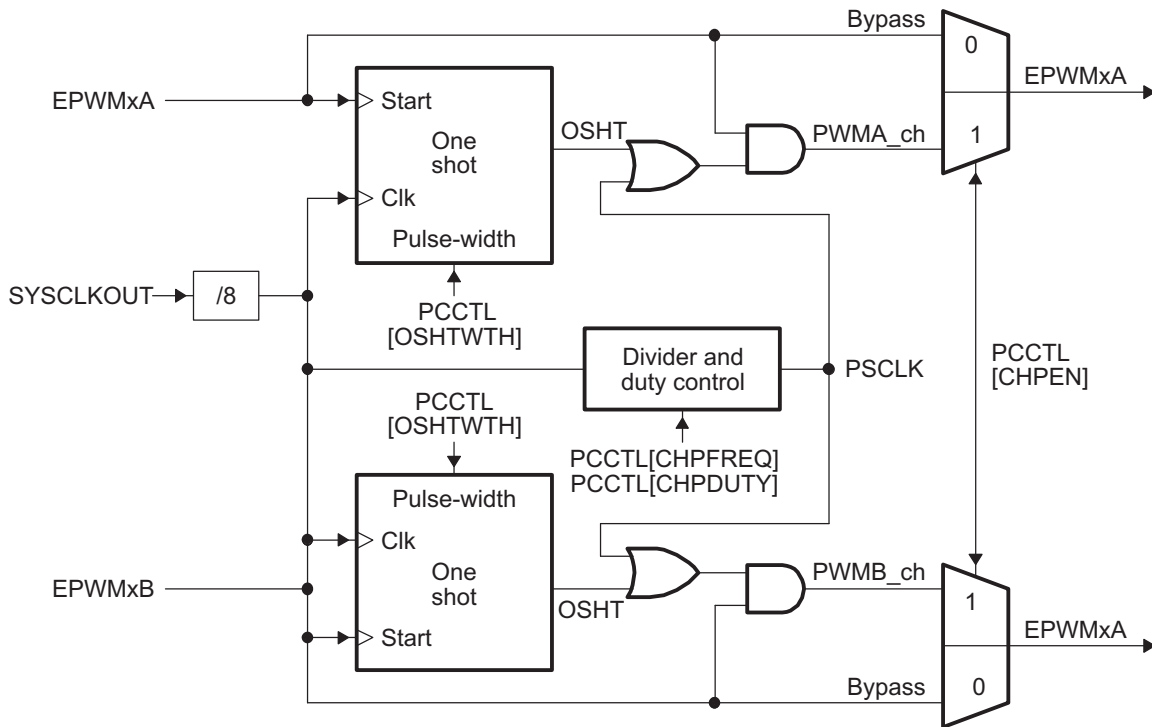
**Table 14-27. PWM-Chopper Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
PCCTL	PWM-chopper Control Register	3Ch	No

### 14.2.7.3 Operational Highlights for the PWM-Chopper Submodule

Figure 14-32 shows the operational details of the PWM-chopper submodule. The carrier clock is derived from SYSCLKOUT. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM-chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

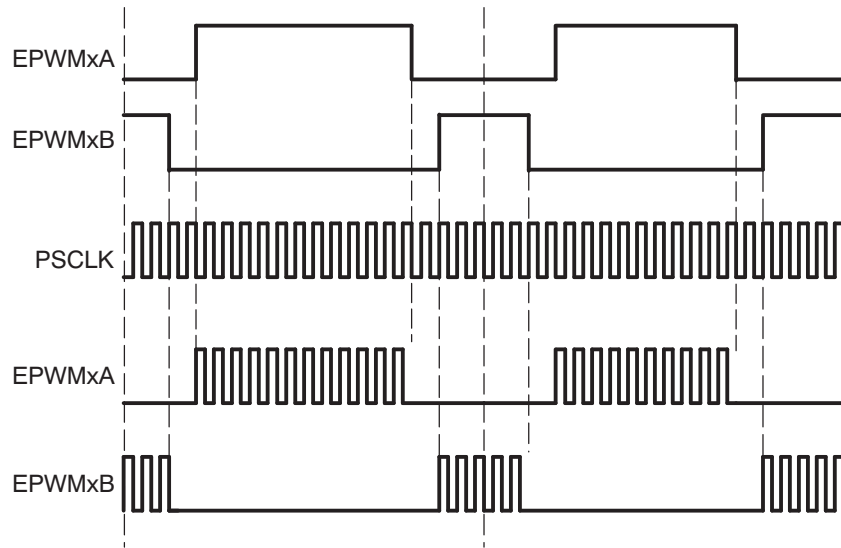
Figure 14-32. PWM-Chopper Submodule Signals and Registers



### 14.2.7.4 Waveforms

Figure 14-33 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

**Figure 14-33. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only**



#### 14.2.7.4.1 One-Shot Pulse

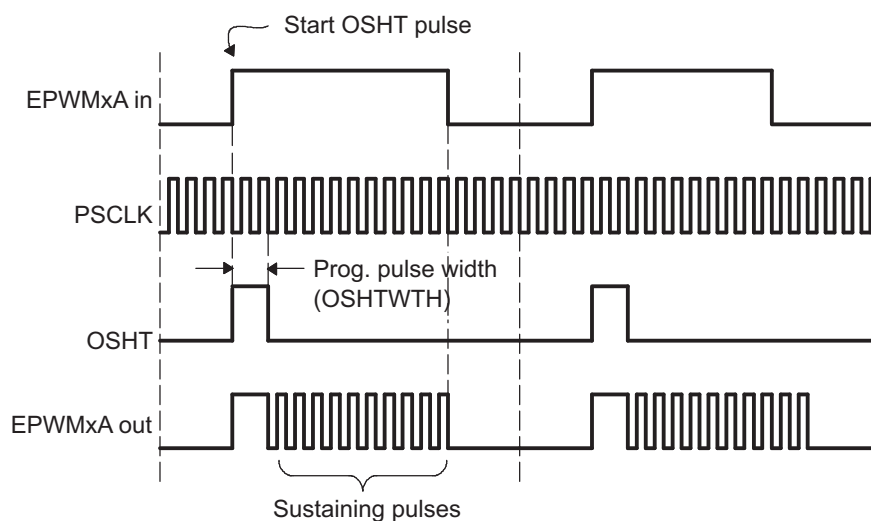
The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1stpulse} = T_{SYSCLKOUT} \times 8 \times OSHTWTH$$

Where  $T_{SYSCLKOUT}$  is the period of the system clock (SYSCLKOUT) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 14-34 shows the first and subsequent sustaining pulses.

**Figure 14-34. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**

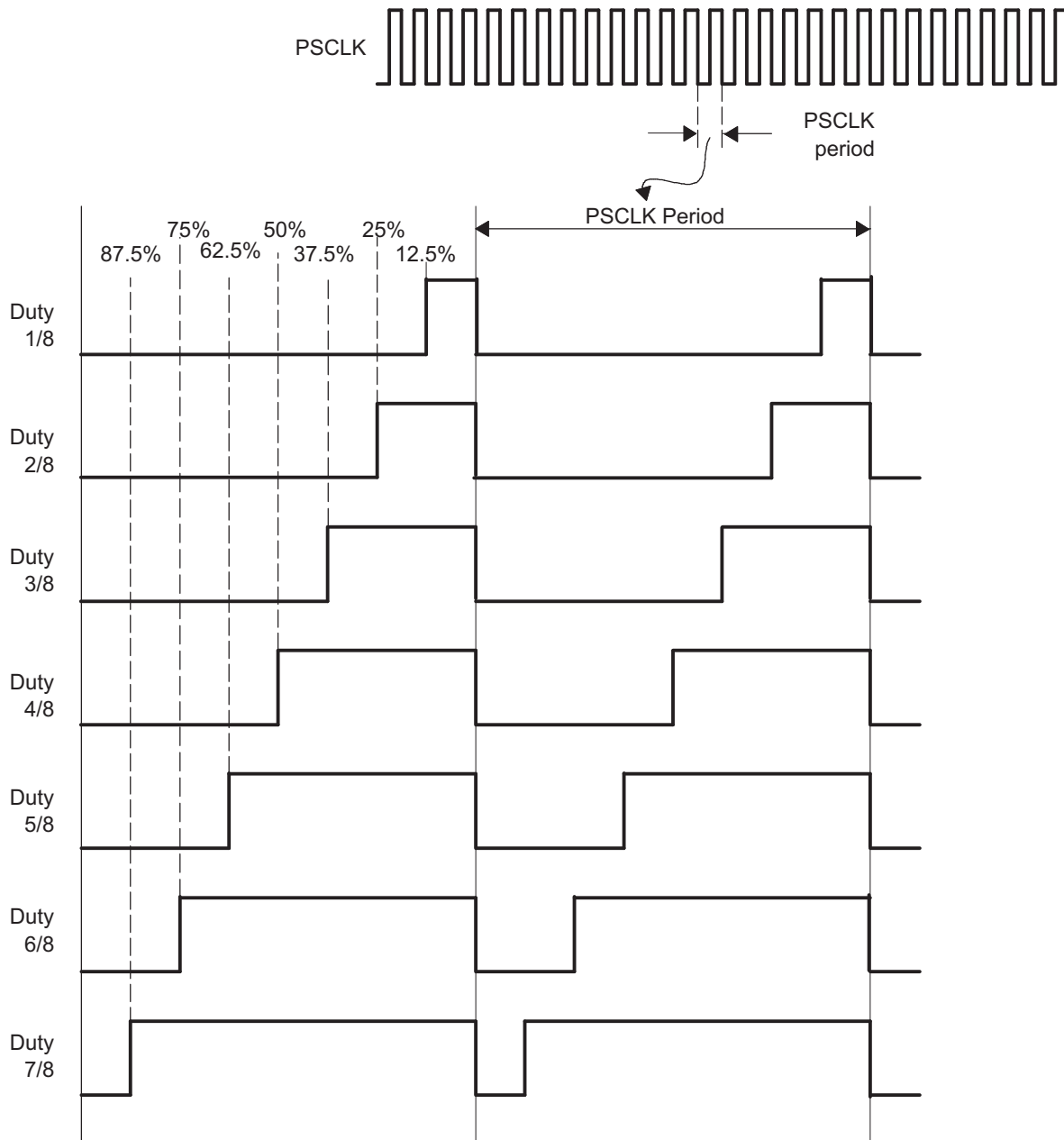


**14.2.7.4.2 Duty Cycle Control**

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 14-35 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5% to 87.5%.

**Figure 14-35. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses**

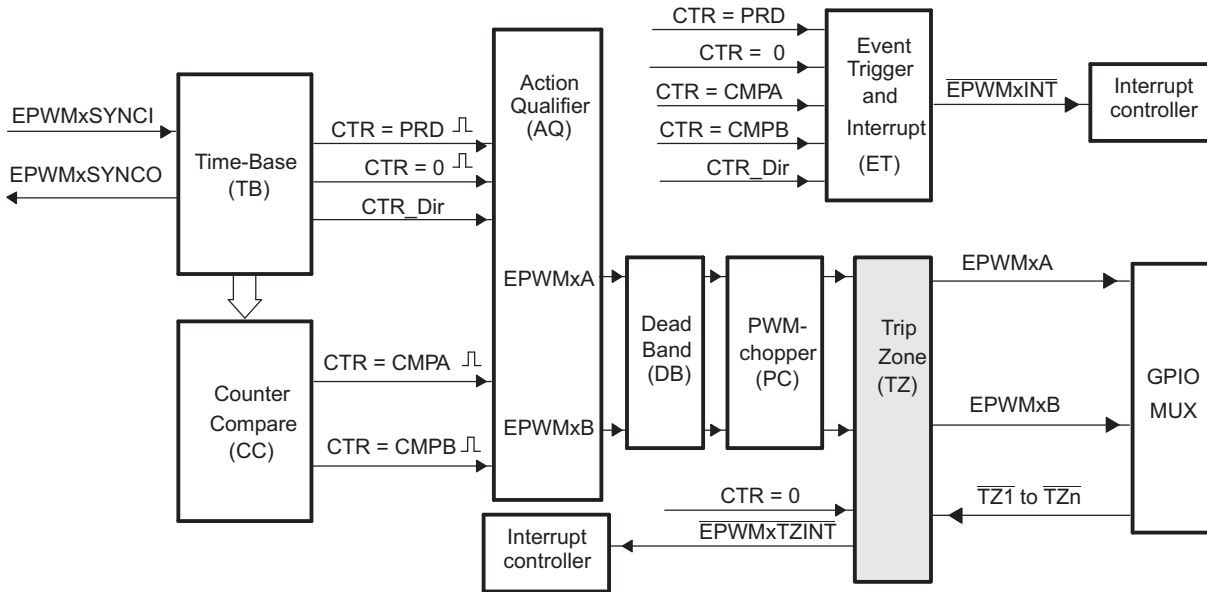




## 14.2.8 Trip-Zone (TZ) Submodule

Figure 14-36 shows how the trip-zone (TZ) submodule fits within the ePWM module. Each ePWM module is connected to every  $\overline{TZ}$  signal that are sourced from the GPIO MUX. These signals indicates external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur. See your device-specific data manual to determine the number of trip-zone pins available for the device.

**Figure 14-36. Trip-Zone Submodule**



### 14.2.8.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs  $\overline{TZ1}$  to  $\overline{TZn}$  can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs  $EPWMxA$  and  $EPWMxB$  can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone pin.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

### 14.2.8.2 Controlling and Monitoring the Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

**Table 14-28. Trip-Zone Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
TZSEL	Trip-Zone Select Register	24h	No
TZCTL	Trip-Zone Control Register	28h	No
TZEINT	Trip-Zone Enable Interrupt Register	2Ah	No
TZFLG	Trip-Zone Flag Register	2Ch	No
TZCLR	Trip-Zone Clear Register	2Eh	No
TZFRC	Trip-Zone Force Register	30h	No

### 14.2.8.3 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals at pin  $\overline{TZ1}$  to  $\overline{TZn}$  is an active-low input signal. When the pin goes low, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone pins. Which trip-zone pins are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signal may or may not be synchronized to the system clock (SYSCLKOUT). A minimum of 1 SYSCLKOUT low pulse on the  $\overline{TZn}$  inputs is sufficient to trigger a fault condition in the ePWM module. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on the  $\overline{TZn}$  inputs.

The  $\overline{TZn}$  input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for a ePWM module. The configuration is determined by the TZSEL[CBCn] and TZSEL[OSHTn] bits (where n corresponds to the trip pin) respectively.

- Cycle-by-Cycle (CBC):** When a cycle-by-cycle trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. [Table 14-29](#) lists the possible actions. In addition, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMxTZINT interrupt is generated if it is enabled in the TZEINT register.
 

The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCNT = 0000h) if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] flag bit will remain set until it is manually cleared by writing to the TZCLR[CBC] bit. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] bit is cleared, then it will again be immediately set.
- One-Shot (OSHT):** When a one-shot trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. [Table 14-29](#) lists the possible actions. In addition, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMxTZINT interrupt is generated if it is enabled in the TZEINT register. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL[TZA] and TZCTL[TZB] register bits. One of four possible actions, shown in [Table 14-29](#), can be taken on a trip event.

**Table 14-29. Possible Actions On a Trip Event**

TZCTL[TZA] and/or TZCTL[TZB]	EPWMxA and/or EPWMxB	Comment
0	High-Impedance	Tripped
1h	Force to High State	Tripped
2h	Force to Low State	Tripped
3h	No Change	Do Nothing. No change is made to the output.

**Example 14-2. Trip-Zone Configurations**
**Scenario A:**

A one-shot trip event on  $\overline{TZ1}$  pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{TZ}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{TZ}$  as a one-shot event source for ePWM2
  - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

**Scenario B:**

A cycle-by-cycle event on  $\overline{TZ5}$  pulls both EPWM1A, EPWM1B low.

A one-shot event on  $\overline{TZ1}$  or  $\overline{TZ6}$  puts EPWM2A into a high impedance state.

- Configure the ePWM1 registers as follows:
  - TZSEL[CBC5] = 1: enables  $\overline{TZ5}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{TZ1}$  as a one-shot event source for ePWM2
  - TZSEL[OSHT6] = 1: enables  $\overline{TZ6}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 0: EPWM1A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM1B will ignore the trip event.

**14.2.8.4 Generating Trip Event Interrupts**

Figure 14-37 and Figure 14-38 illustrate the trip-zone submodule control and interrupt logic, respectively.

Figure 14-37. Trip-Zone Submodule Mode Control Logic

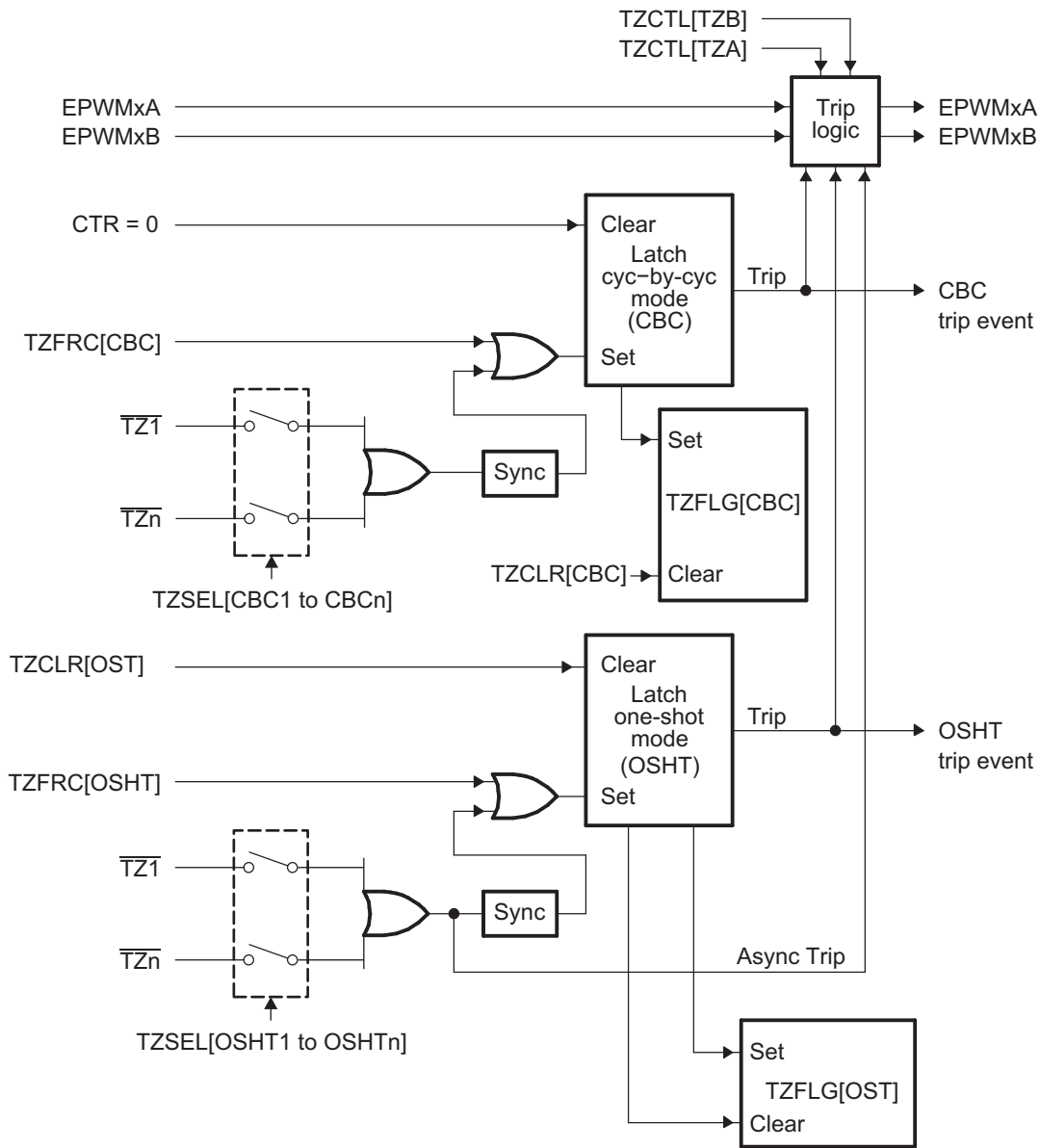
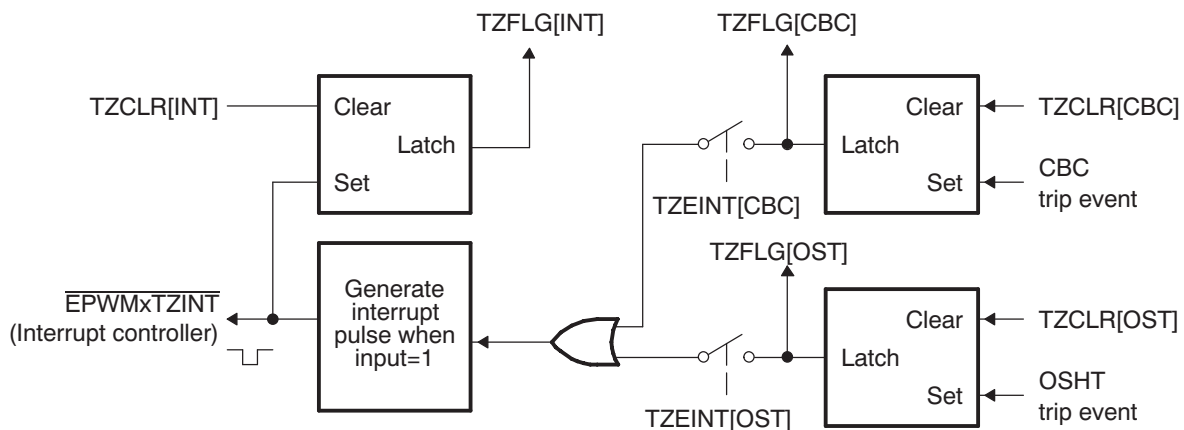


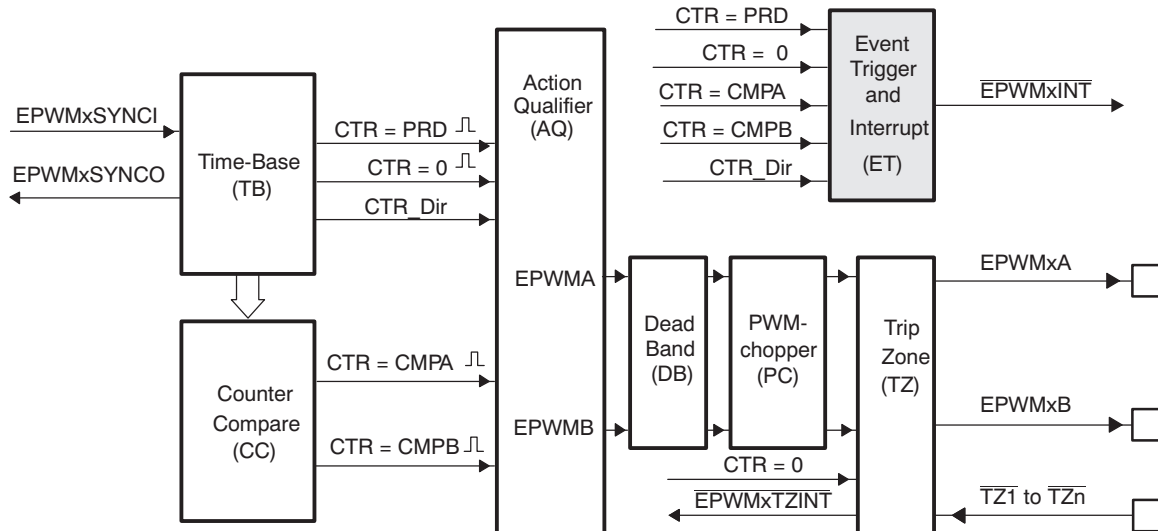
Figure 14-38. Trip-Zone Submodule Interrupt Logic



### 14.2.9 Event-Trigger (ET) Submodule

Figure 14-39 shows the event-trigger (ET) submodule in the ePWM system. The event-trigger submodule manages the events generated by the time-base submodule and the counter-compare submodule to generate an interrupt to the CPU.

Figure 14-39. Event-Trigger Submodule



#### 14.2.9.1 Purpose of the Event-Trigger Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base and counter-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags

#### 14.2.9.2 Controlling and Monitoring the Event-Trigger Submodule

The key registers used to configure the event-trigger submodule are shown in [Table 14-30](#):

Table 14-30. Event-Trigger Submodule Registers

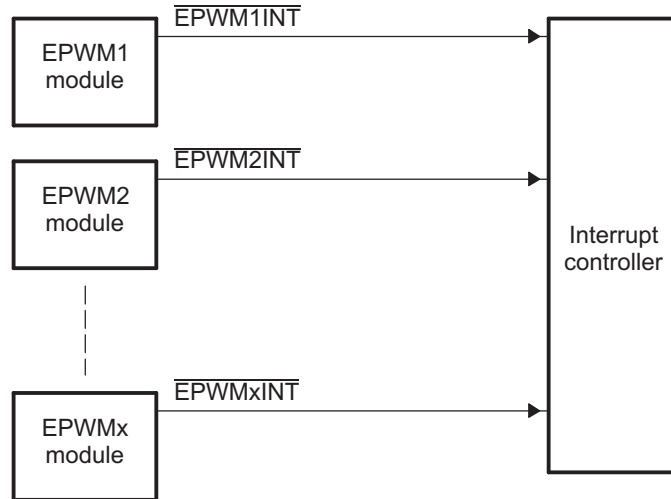
Acronym	Register Description	Address Offset	Shadowed
ETSEL	Event-Trigger Selection Register	32h	No
ETPS	Event-Trigger Prescale Register	34h	No
ETFLG	Event-Trigger Flag Register	36h	No
ETCLR	Event-Trigger Clear Register	38h	No
ETFRC	Event-Trigger Force Register	3Ah	No

### 14.2.9.3 Operational Overview of the Event-Trigger Submodule

The following sections describe the event-trigger submodule's operational highlights.

Each ePWM module has one interrupt request line connected to the interrupt controller as shown in Figure 14-40.

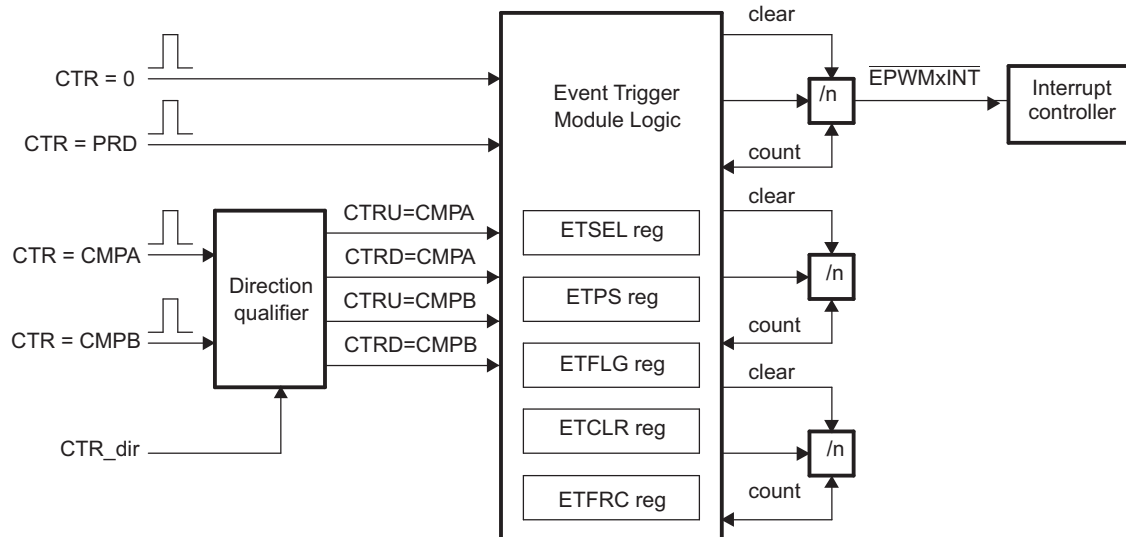
**Figure 14-40. Event-Trigger Submodule Inter-Connectivity to Interrupt Controller**



The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in Figure 14-41) and can be configured to prescale these events before issuing an Interrupt request. The event-trigger prescaling logic can issue Interrupt requests at:

- Every event
- Every second event
- Every third event

**Figure 14-41. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**



- **ETSEL**—This selects which of the possible events will trigger an interrupt.
- **ETPS**—This programs the event prescaling options previously mentioned.
- **ETFLG**—These are flag bits indicating status of the selected and prescaled events.
- **ETCLR**—These bits allow you to clear the flag bits in the ETFLG register via software.
- **ETFRC**—These bits allow software forcing of an event. Useful for debugging or software intervention.

A more detailed look at how the various register bits interact with the Interrupt is shown in [Figure 14-42](#).

[Figure 14-42](#) shows the event-trigger's interrupt generation logic. The interrupt-period (ETPS[INTPRD]) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

An interrupt cannot be generated on every fourth or more events.

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) bits. The event can be one of the following:

- Time-base counter equal to zero (TBCNT = 0000h).
- Time-base counter equal to period (TBCNT = TBPRD).
- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.

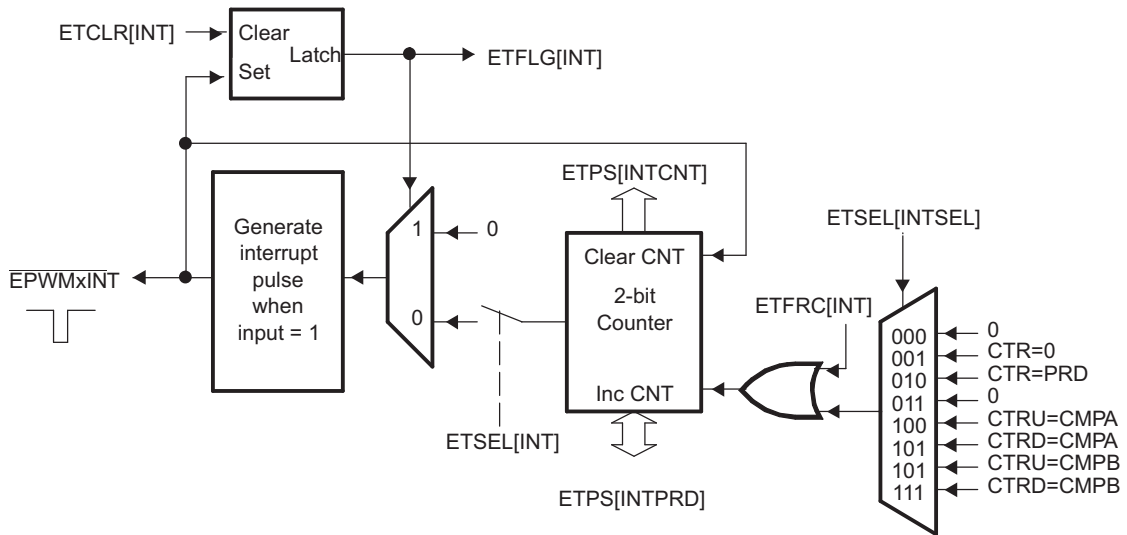
The number of events that have occurred can be read from the interrupt event counter (ETPS[INTCNT]) register bits. That is, when the specified event occurs the ETPS[INTCNT] bits are incremented until they reach the value specified by ETPS[INTPRD]. When ETPS[INTCNT] = ETPS[INTPRD] the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the interrupt controller.

When ETPS[INTCNT] reaches ETPS[INTPRD], one of the following behaviors will occur:

- If interrupts are enabled, ETSEL[INTEN] = 1 and the interrupt flag is clear, ETFLG[INT] = 0, then an interrupt pulse is generated and the interrupt flag is set, ETFLG[INT] = 1, and the event counter is cleared ETPS[INTCNT] = 0. The counter will begin counting events again.
- If interrupts are disabled, ETSEL[INTEN] = 0, or the interrupt flag is set, ETFLG[INT] = 1, the counter stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1 to the ETFRC[INT] bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events will be detected and the ETFRC[INT] bit is also ignored.

**Figure 14-42. Event-Trigger Interrupt Generator**

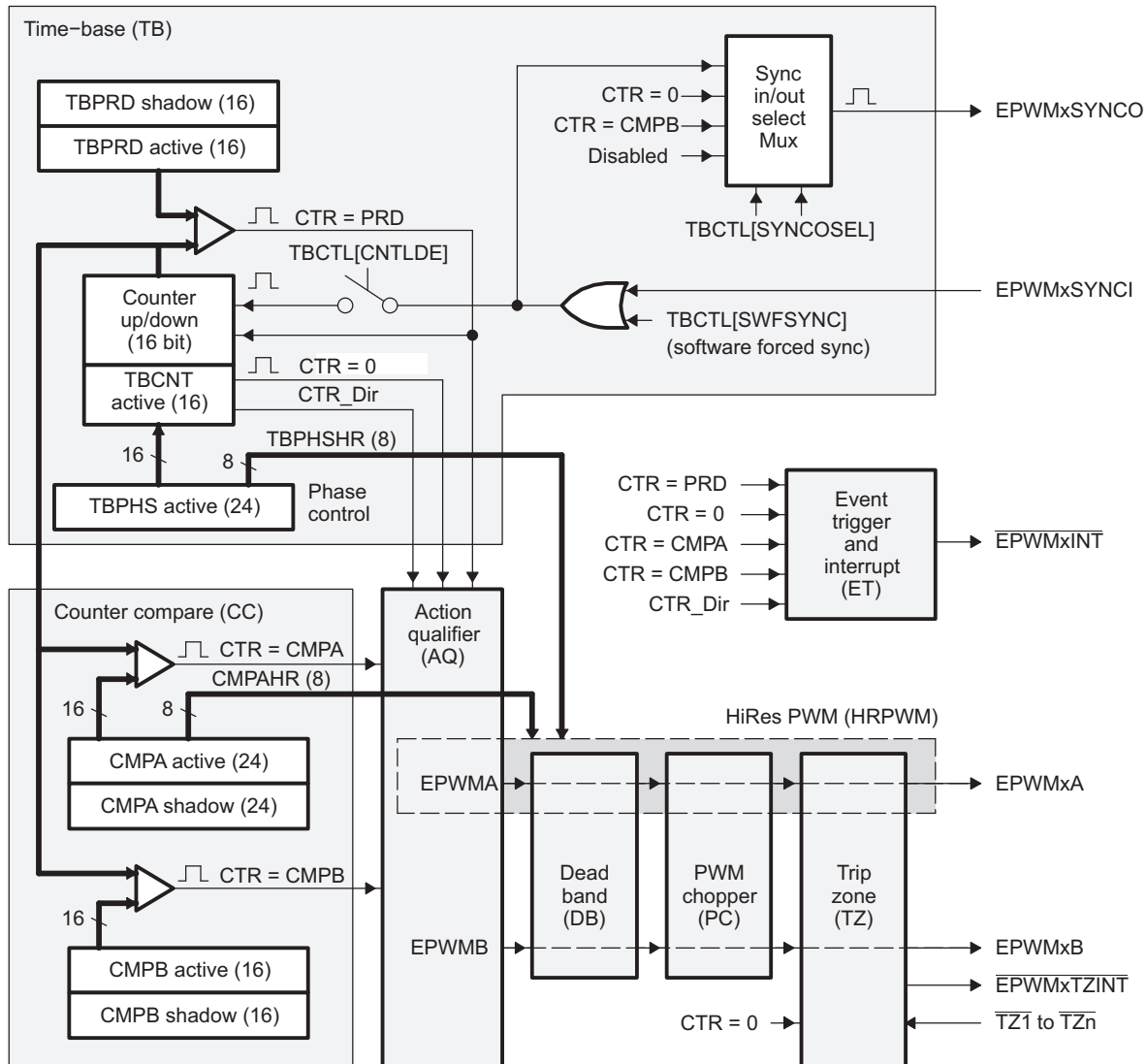




### 14.2.10 High-Resolution PWM (HRPWM) Submodule

Figure 14-43 shows the high-resolution PWM (HRPWM) submodule in the ePWM system. Some devices include the high-resolution PWM submodule, see your device-specific data manual to determine which ePWM instances include this feature.

Figure 14-43. HRPWM System Interface



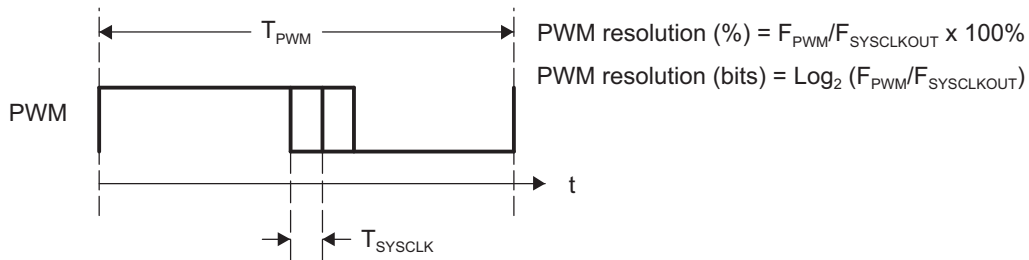
### 14.2.10.1 Purpose of the High-Resolution PWM Submodule

The enhanced high-resolution pulse-width modulator (eHRPWM) extends the time resolution capabilities of the conventionally derived digital pulse-width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~9-10 bits. The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A and Phase registers
- Implemented using the A signal path of PWM, that is, on the EPWMxA output. EPWMxB output has conventional PWM capabilities

The ePWM peripheral is used to perform a function that is mathematically equivalent to a digital-to-analog converter (DAC). As shown in Figure 14-44, the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

Figure 14-44. Resolution Calculations for Conventionally Generated PWM



If the required PWM operating frequency does not offer sufficient resolution in PWM mode, you may want to consider HRPWM. As an example of improved performance offered by HRPWM, Table 14-31 shows resolution in bits for various PWM frequencies. Table 14-31 values assume a MEP step size of 180 ps. See your device-specific data manual for typical and maximum performance specifications for the MEP.

Table 14-31. Resolution for PWM and HRPWM

PWM Frequency (kHz)	Regular Resolution (PWM)		High Resolution (HRPWM)	
	Bits	%	Bits	%
20	12.3	0.0	18.1	0.000
50	11.0	0.0	16.8	0.001
100	10.0	0.1	15.8	0.002
150	9.4	0.2	15.2	0.003
200	9.0	0.2	14.8	0.004
250	8.6	0.3	14.4	0.005
500	7.6	0.5	13.8	0.007
1000	6.6	1.0	12.4	0.018
1500	6.1	1.5	11.9	0.027
2000	5.6	2.0	11.4	0.036

Although each application may differ, typical low-frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high-frequency PWM requirements of power conversion topologies such as:

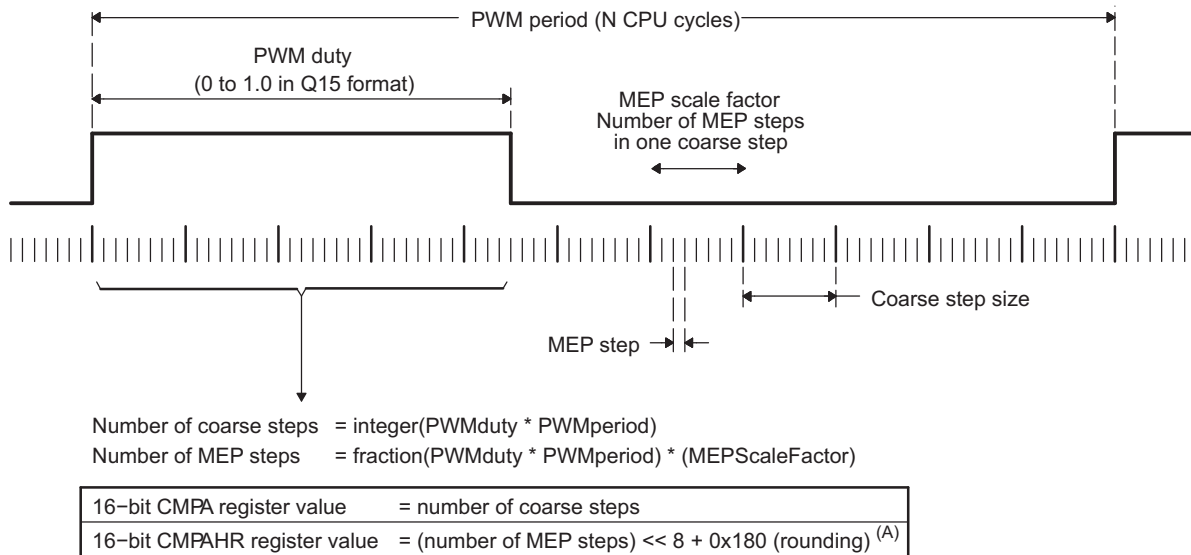
- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

### 14.2.10.2 Architecture of the High-Resolution PWM Submodule

The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions.

Figure 14-45 shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (CMPAHR).

**Figure 14-45. Operating Logic Using MEP**



A For MEP range and rounding adjustment.

To generate an HRPWM waveform, configure the TBM, CCM, and AQM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the TBM, CCM, and AQM registers to extend edge resolution, and should be configured accordingly. Although many programming combinations are possible, only a few are needed and practical.

### 14.2.10.3 Controlling and Monitoring the High-Resolution PWM Submodule

The MEP of the HRPWM is controlled by two extension registers, each 8-bits wide. These two HRPWM registers are concatenated with the 16-bit TBPHS and CMPA registers used to control PWM operation.

- TBPHSHR - Time-Base Phase High-Resolution Register
- CMPAHR - Counter-Compare A High-Resolution Register

Table 14-32 lists the registers used to control and monitor the high-resolution PWM submodule.

**Table 14-32. HRPWM Submodule Registers**

Acronym	Register Description	Address Offset	Shadowed
TBPHSHR	Extension Register for HRPWM Phase	4h	No
CMPAHR	Extension Register for HRPWM Duty	10h	Yes
HRCNFG	HRPWM Configuration Register	1040h	No

#### 14.2.10.4 Configuring the High-Resolution PWM Submodule

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register located at offset address 1040h. This register provides configuration options for the following key operating modes:

- **Edge Mode:** The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE), or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control, while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge.
- **Control Mode:** The MEP is programmed to be controlled either from the CMPAHR register (duty cycle control) or the TBPHSHR register (phase control). RE or FE control mode should be used with CMPAHR register. BE control mode should be used with TBPHSHR register.
- **Shadow Mode:** This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the CMPAHR register and should be chosen to be the same as the regular load option for the CMPA register. If TBPHSHR is used, then this option has no effect.

#### 14.2.10.5 Operational Highlights for the High-Resolution PWM Submodule

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps, each of which has a time resolution on the order of 150 ps. The MEP works with the TBM and CCM registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. [Table 14-33](#) shows the typical range of operating frequencies supported by the HRPWM.

**Table 14-33. Relationship Between MEP Steps, PWM Frequency and Resolution**

System (MHz)	MEP Steps Per SYSCLKOUT <sup>(1) (2) (3)</sup>	PWM Minimum (Hz) <sup>(4)</sup>	PWM Maximum (MHz)	Resolution at Maximum (Bits) <sup>(5)</sup>
50.0	111	763	2.50	11.1
60.0	93	916	3.00	10.9
70.0	79	1068	3.50	10.6
80.0	69	1221	4.00	10.4
90.0	62	1373	4.50	10.3
100.0	56	1526	5.00	10.1

<sup>(1)</sup> System frequency = SYSCLKOUT, that is, CPU clock. TBCLK = SYSCLKOUT

<sup>(2)</sup> Table data based on a MEP time resolution of 180 ps (this is an example value)

<sup>(3)</sup> MEP steps applied =  $T_{\text{SYSCLKOUT}}/180 \text{ ps}$  in this example.

<sup>(4)</sup> PWM minimum frequency is based on a maximum period value, TBPRD = 65 535. PWM mode is asymmetrical up-count.

<sup>(5)</sup> Resolution in bits is given for the maximum PWM frequency stated.

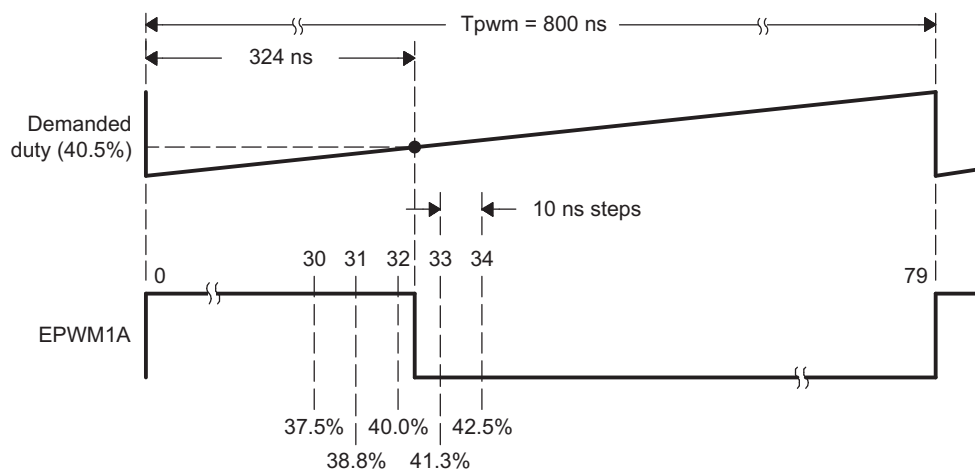
### 14.2.10.5.1 Edge Positioning

In a typical power control loop (switch modes, digital motor control (DMC), uninterruptible power supply (UPS)), a digital controller (PID, 2pole/2zero, lag/lead, etc.) issues a duty command, usually expressed in a per unit or percentage terms.

In the following example, assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on-time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 100 MHz, the duty cycle choices are in the vicinity of 40.5%. In [Figure 14-46](#), a compare value of 32 counts (duty = 40%) is the closest to 40.5% that you can attain. This is equivalent to an edge position of 320 ns instead of the desired 324 ns. This data is shown in [Table 14-34](#).

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns. [Table 14-34](#) shows that in addition to the CMPA value, 22 steps of the MEP (CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ns.

**Figure 14-46. Required PWM Waveform for a Requested Duty = 40.5%**



**Table 14-34. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)**

CMPA (count) <sup>(1)</sup> <sup>(2)</sup> <sup>(3)</sup>	DUTY (%)	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	35.0	280	32	18	40.405	323.24
29	36.3	290	32	19	40.428	323.42
30	37.5	300	32	20	40.450	323.60
31	38.8	310	32	21	40.473	323.78
32	40.0	320	32	22	40.495	323.96
33	41.3	330	32	23	40.518	324.14
34	42.5	340	32	24	40.540	324.32
			32	25	40.563	324.50
Required			32	26	40.585	324.68
32.40	40.5	324	32	27	40.608	324.86

<sup>(1)</sup> System clock, SYSCLKOUT and TBCLK = 100 MHz, 10 ns

<sup>(2)</sup> For a PWM Period register value of 80 counts, PWM Period = 80 × 10 ns = 800 ns, PWM frequency = 1/800 ns = 1.25 MHz

<sup>(3)</sup> Assumed MEP step size for the above example = 180 ps

### 14.2.10.5.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard (CMPA) and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.

Assumptions for this example:

System clock, SYSCLKOUT	= 10 ns (100 MHz)
PWM frequency	= 1.25 MHz (1/800 ns)
Required PWM duty cycle, <b>PWMDuty</b>	= 0.405 (40.5%)
PWM period in terms of coarse steps, <b>PWMperiod</b> (800 ns/10 ns)	= 80
Number of MEP steps per coarse step at 180 ps (10 ns/180 ps), <b>MEP_SF</b>	= 55
Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value)	= 180h

#### Step 1: Percentage Integer Duty value conversion for CMPA register

CMPA register value	= $\text{int}(\text{PWMDuty} \times \text{PWMperiod})$ ; int means integer part
	= $\text{int}(0.405 \times 80)$
	= $\text{int}(32.4)$
CMPA register value	= 32 (20h)

#### Step 2: Fractional value conversion for CMPAHR register

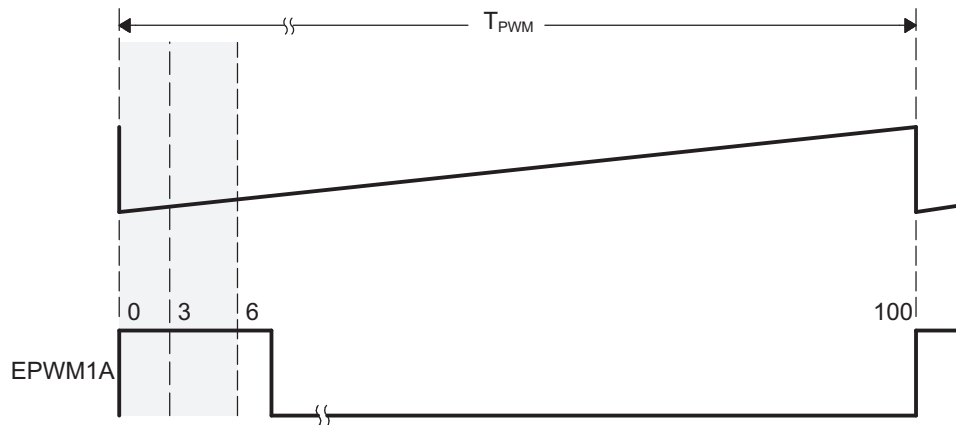
CMPAHR register value	= $(\text{frac}(\text{PWMDuty} \times \text{PWMperiod}) \times \text{MEP\_SF}) \ll 8) + 180\text{h}$ ; frac means fractional part
	= $(\text{frac}(32.4) \times 55 \ll 8) + 180\text{h}$ ; Shift is to move the value as CMPAHR high byte
	= $((0.4 \times 55) \ll 8) + 180\text{h}$
	= $(22 \ll 8) + 180\text{h}$
	= $22 \times 256 + 180\text{h}$ ; Shifting left by 8 is the same multiplying by 256.
	= $5632 + 180\text{h}$
	= $1600\text{h} + 180\text{h}$
CMPAHR value	= 1780h; CMPAHR value = 1700h, lower 8 bits will be ignored by hardware.

### 14.2.10.5.3 Duty Cycle Range Limitation

In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational 3 SYCLK cycles after the period starts.

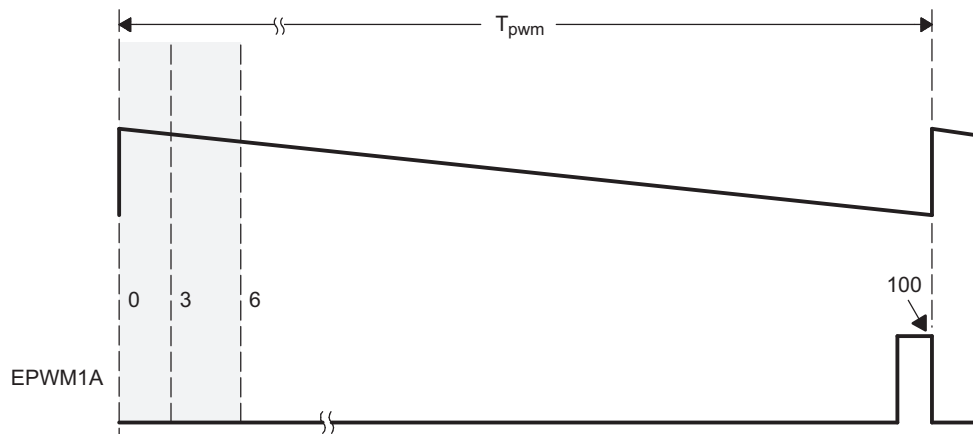
Duty cycle range limitations are illustrated in [Figure 14-47](#). This limitation imposes a lower duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. Although for the first 3 or 6 cycles, the HRPWM capabilities are not available, regular PWM duty control is still fully operational down to 0% duty. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle.

**Figure 14-47. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz**



If the application demands HRPWM operation in the low percent duty cycle region, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP. This is illustrated in [Figure 14-48](#). In this case low percent duty limitation is no longer an issue.

**Figure 14-48. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz**



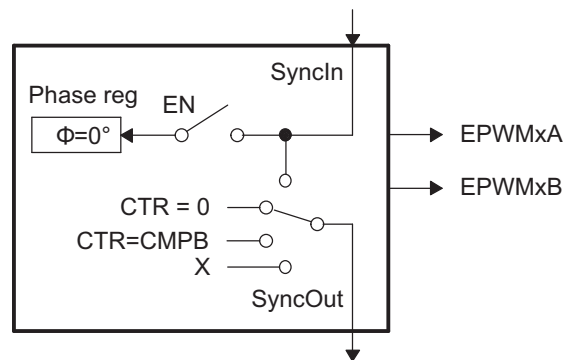
### 14.3 Applications to Power Topologies

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

#### 14.3.1 Overview of Multiple Modules

Previously in this user's guide, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in [Figure 14-49](#). This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

**Figure 14-49. Simplified ePWM Module**





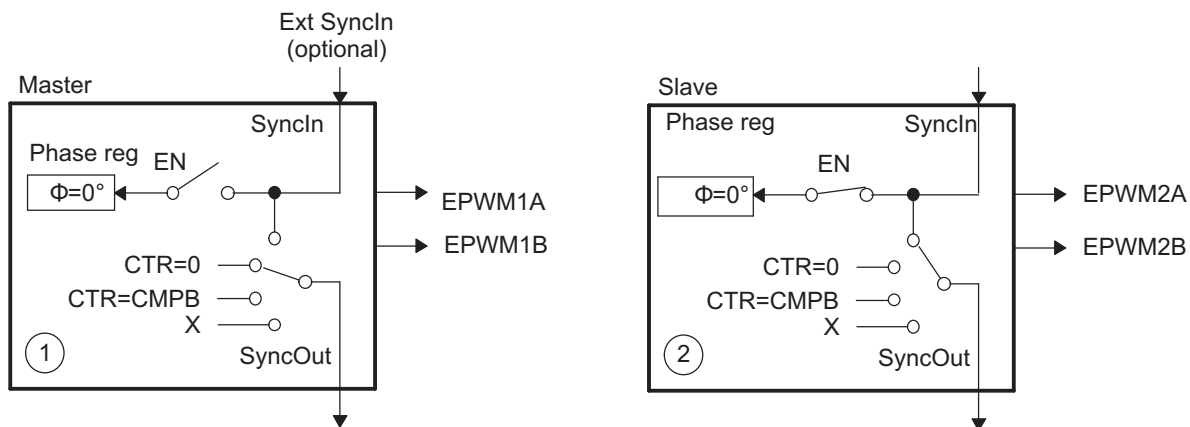
### 14.3.2 Key Configuration Capabilities

The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)

For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it, i.e., via the enable switch. Although various combinations are possible, the two most common—master module and slave module modes—are shown in [Figure 14-50](#).

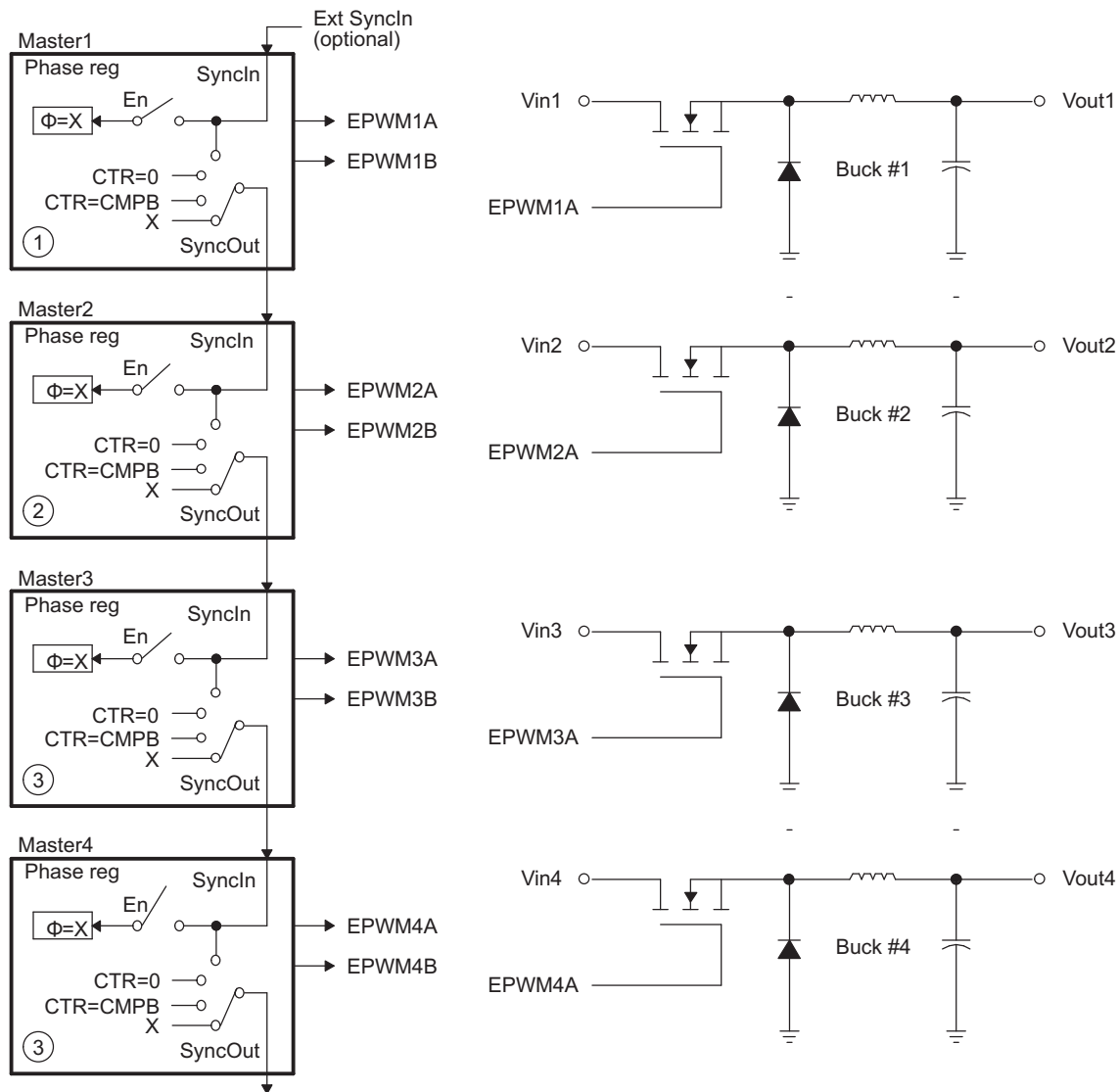
**Figure 14-50. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave**



### 14.3.3 Controlling Multiple Buck Converters With Independent Frequencies

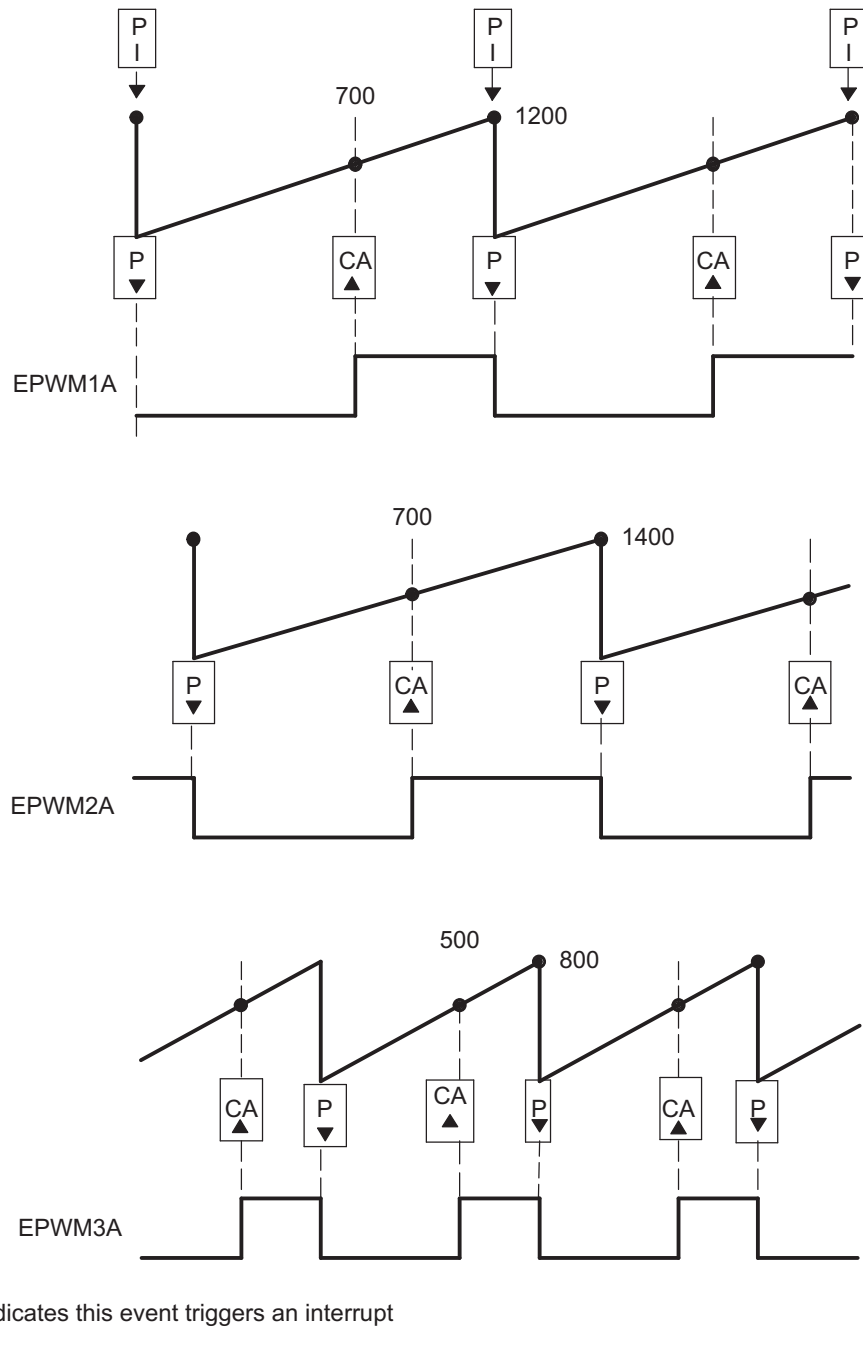
One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 14-51 shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. Figure 14-52 shows the waveforms generated by the setup shown in Figure 14-51; note that only three waveforms are shown, although there are four stages.

Figure 14-51. Control of Four Buck Stages. (Note:  $F_{P_{WM1}} \neq F_{P_{WM2}} \neq F_{P_{WM3}} \neq F_{P_{WM4}}$ )



NOTE:  $\Phi = X$  indicates value in phase register is a "don't care"

Figure 14-52. Buck Waveforms for Figure 14-51 (Note: Only three bucks shown here)



**Table 14-35. EPWM1 Initialization for Figure 14-52**

Register	Bit	Value	Comments
TBPRD	TBPRD	1200 (4B0h)	Period = 1201 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	

**Table 14-36. EPWM2 Initialization for Figure 14-52**

Register	Bit	Value	Comments
TBPRD	TBPRD	1400 (578h)	Period = 1401 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	

**Table 14-37. EPWM3 Initialization for Figure 14-52**

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 801 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	

**Example 14-3. Configuration for Example in Figure 14-52**

```

// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 700;           // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 700;           // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 500;           // adjust duty for output EPWM3A
    
```

**14.3.4 Controlling Multiple Buck Converters With Same Frequencies**

If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. Figure 14-53 shows such a configuration; Figure 14-54 shows the waveforms generated by the configuration.

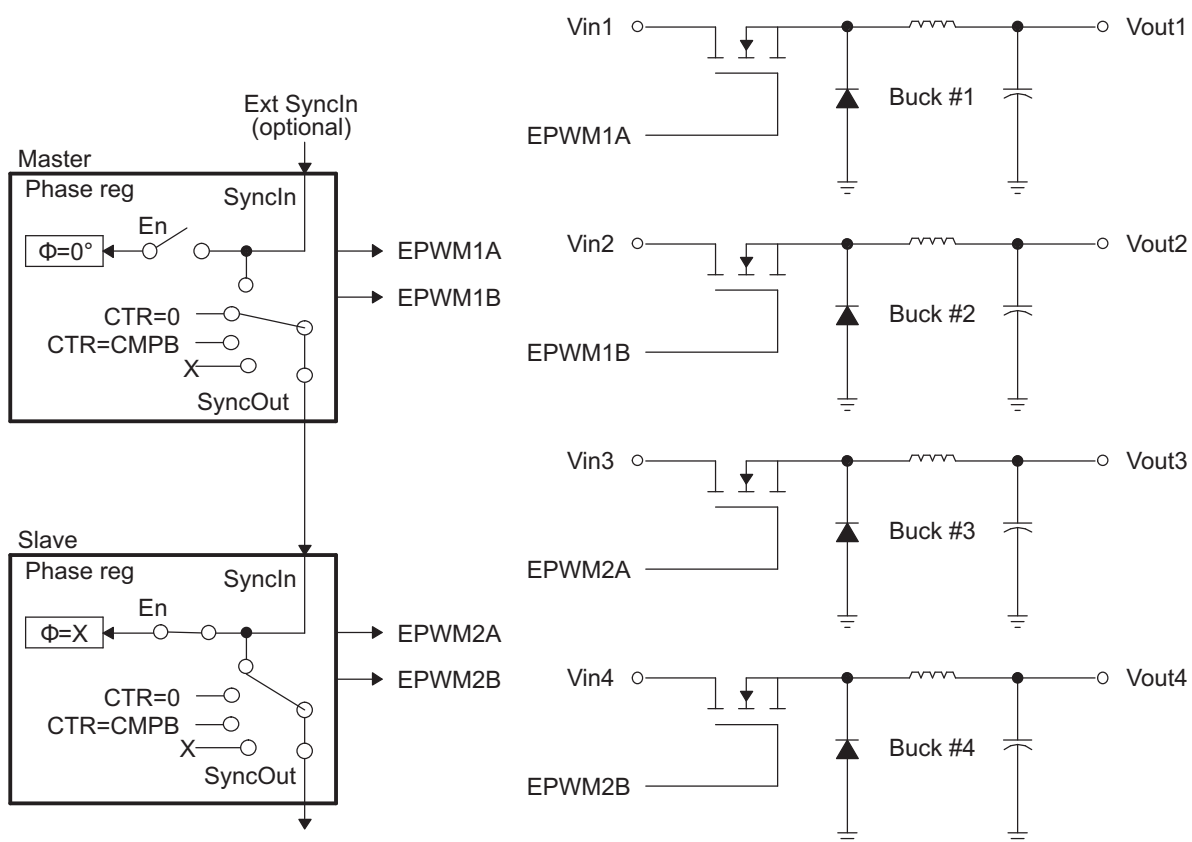
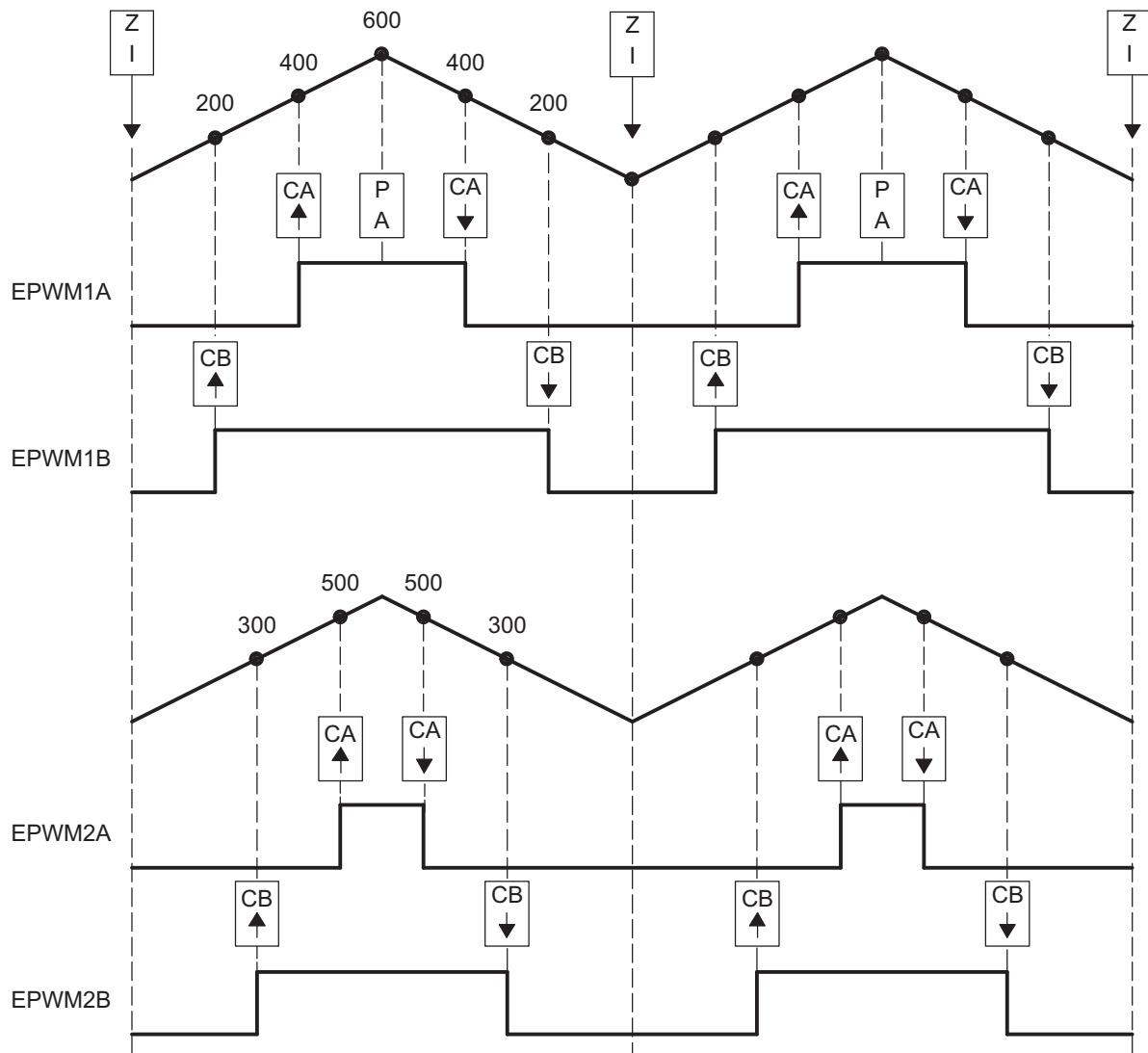
**Figure 14-53. Control of Four Buck Stages. (Note:  $F_{PWM2} = N \times F_{PWM1}$ )**


Figure 14-54. Buck Waveforms for Figure 14-53 (Note:  $F_{P_{WM2}} = F_{P_{WM1}}$ )



**Table 14-38. EPWM1 Initialization for Figure 14-53**

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM1A
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_SET	Set actions for EPWM1B
	CBD	AQ_CLEAR	

**Table 14-39. EPWM2 Initialization for Figure 14-53**

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Phase loading enabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM2A
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_SET	Set actions for EPWM2B
	CBD	AQ_CLEAR	

**Example 14-4. Code Snippet for Configuration in Figure 14-53**

```

// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 400;      // adjust duty for output EPWM1A
EPwm1Regs.CMPB = 200;                // adjust duty for output EPWM1B
EPwm2Regs.CMPA.half.CMPA = 500;      // adjust duty for output EPWM2A
EPwm2Regs.CMPB = 300;                // adjust duty for output EPWM2B
    
```

### 14.3.5 Controlling Multiple Half H-Bridge (HHB) Converters

Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 14-55 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 14-56 shows the waveforms generated by the configuration shown in Figure 14-55.

Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

Figure 14-55. Control of Two Half-H Bridge Stages ( $F_{P\text{WM}2} = N \times F_{P\text{WM}1}$ )

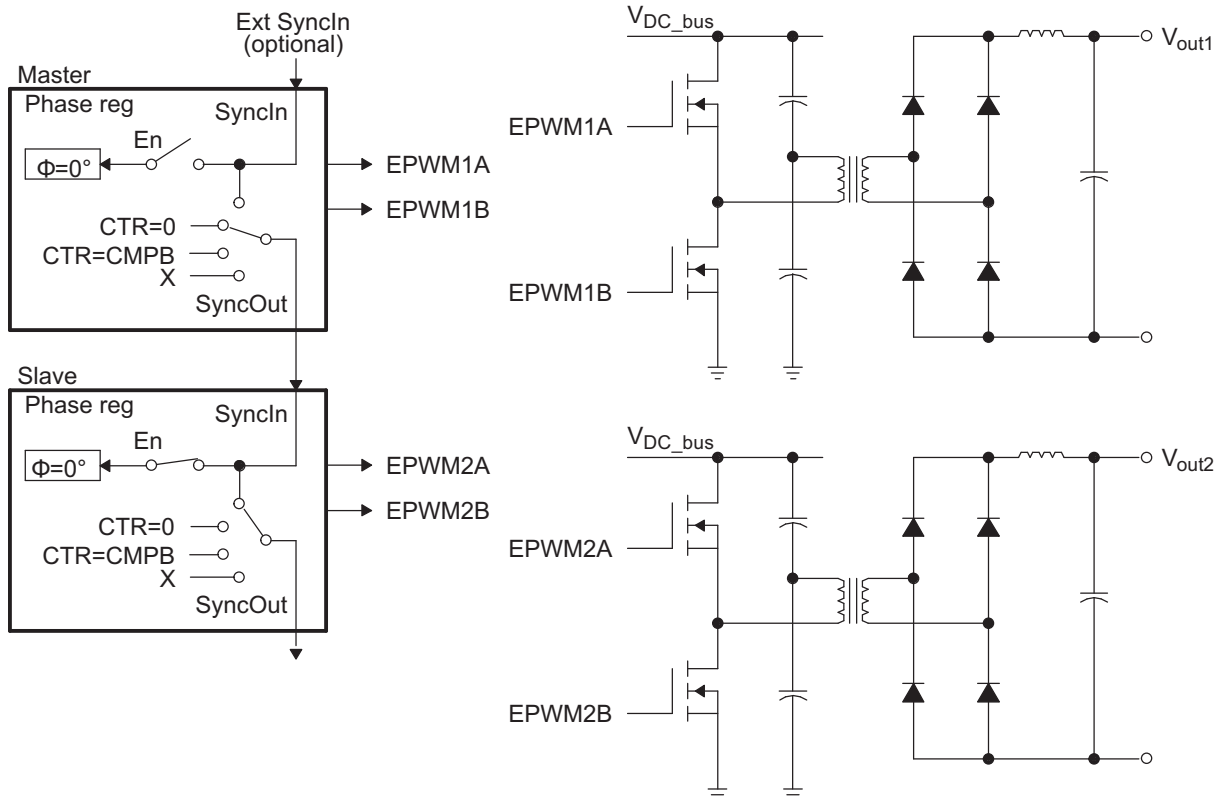
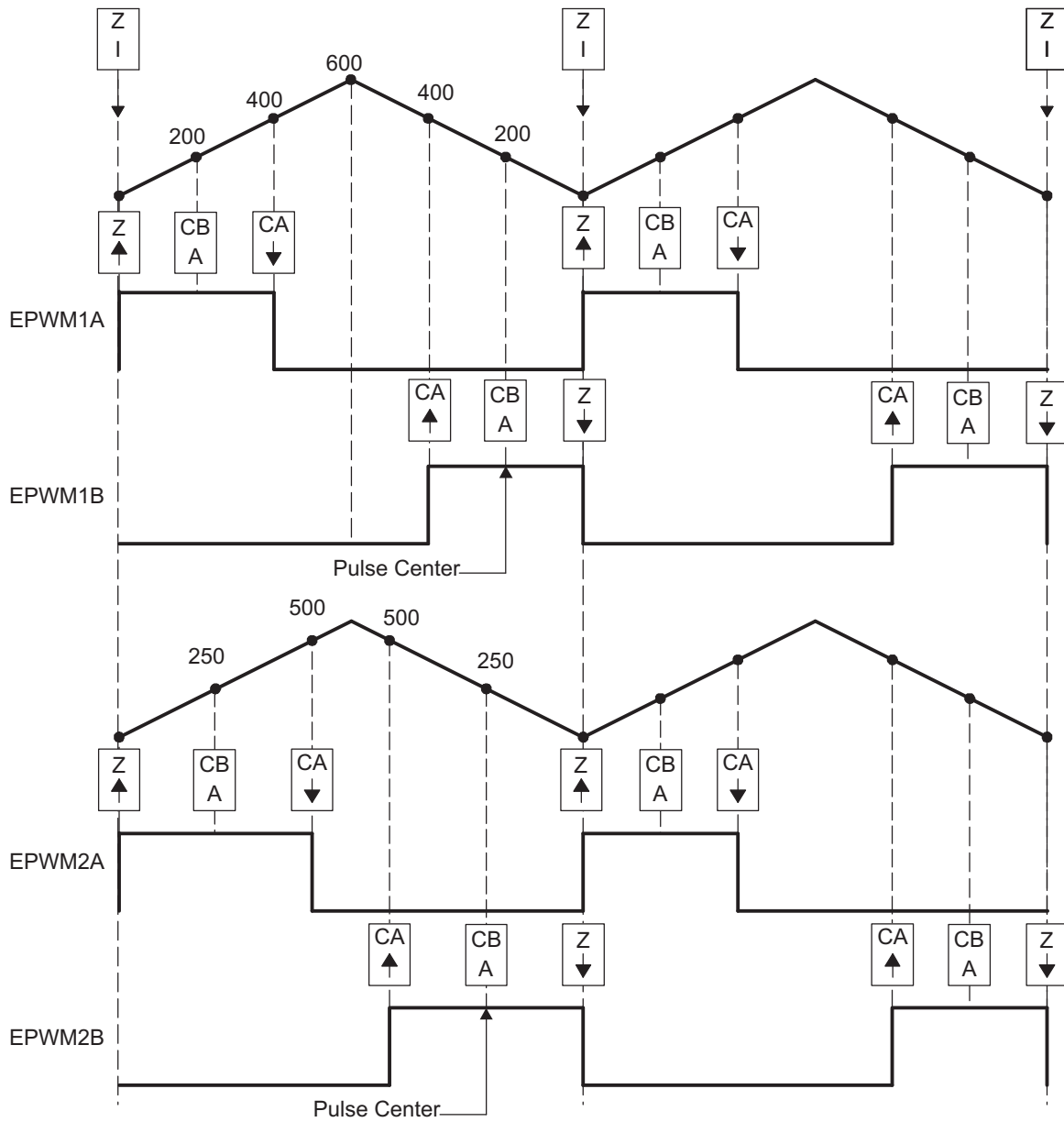




Figure 14-56. Half-H Bridge Waveforms for Figure 14-55 (Note:  $F_{PVM2} = F_{PVM1}$ )



**Table 14-40. EPWM1 Initialization for Figure 14-55**

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM1A
	CAU	AQ_CLEAR	
AQCTLB	ZRO	AQ_CLEAR	Set actions for EPWM1B
	CAD	AQ_SET	

**Table 14-41. EPWM2 Initialization for Figure 14-55**

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Phase loading enabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM2A
	CAU	AQ_CLEAR	
AQCTLB	ZRO	AQ_CLEAR	Set actions for EPWM2B
	CAD	AQ_SET	

**Example 14-5. Code Snippet for Configuration in Figure 14-55**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 400; // adjust duty for output EPWM1A
EPwm1Regs.CMPB = 200;          // adjust duty for output EPWM1B
EPwm2Regs.CMPA.half.CMPA = 500; // adjust duty for output EPWM2A
EPwm2Regs.CMPB = 250;          // adjust duty for output EPWM2B
```

### 14.3.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

The idea of multiple modules controlling a single power stage can be extended to the 3-phase Inverter case. In such a case, six switching elements can be controlled using three PWM modules, one for each leg of the inverter. Each leg must switch at the same frequency and all legs must be synchronized. A master + two slaves configuration can easily address this requirement. Figure 14-57 shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in Figure 14-57), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

Figure 14-57. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control

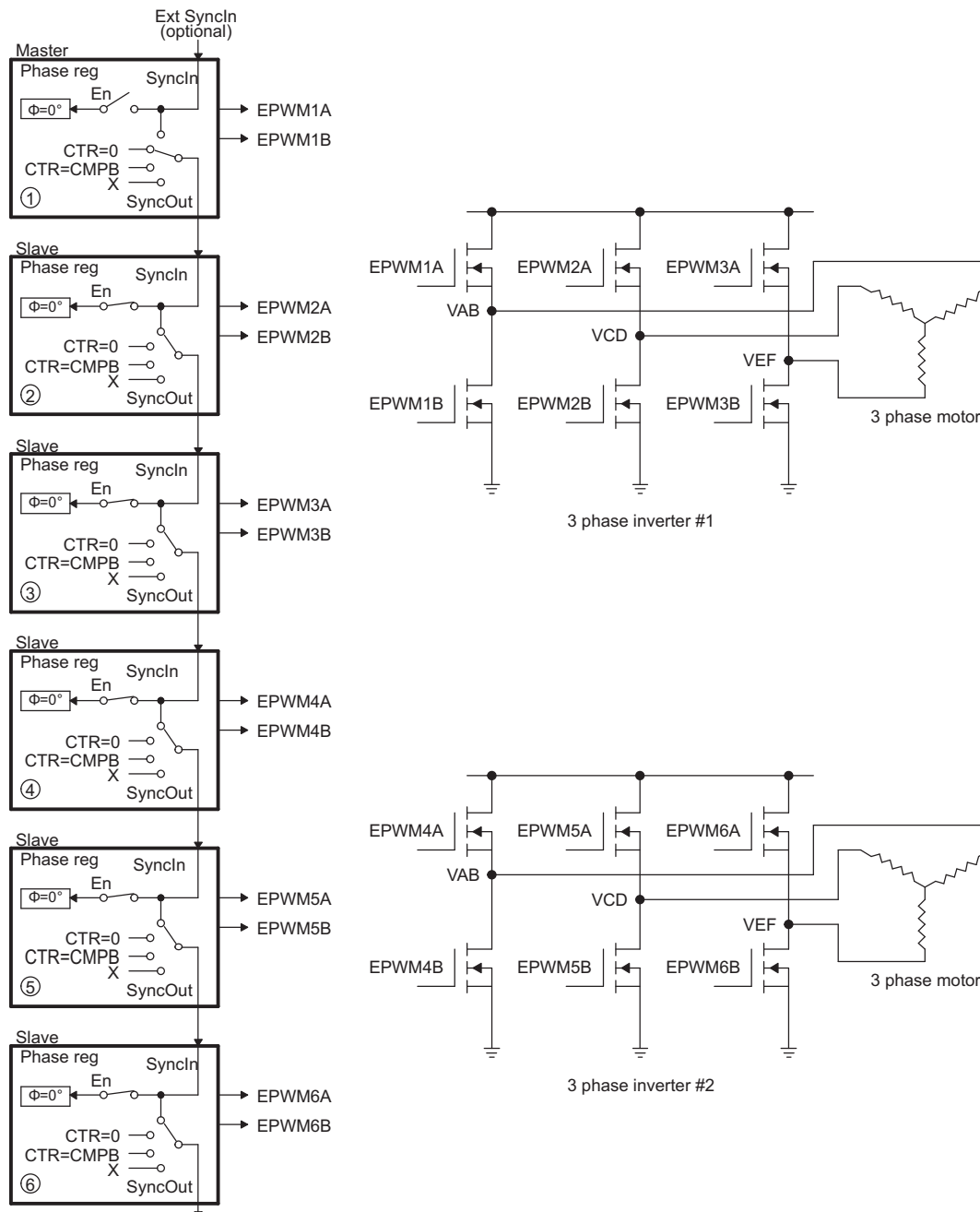
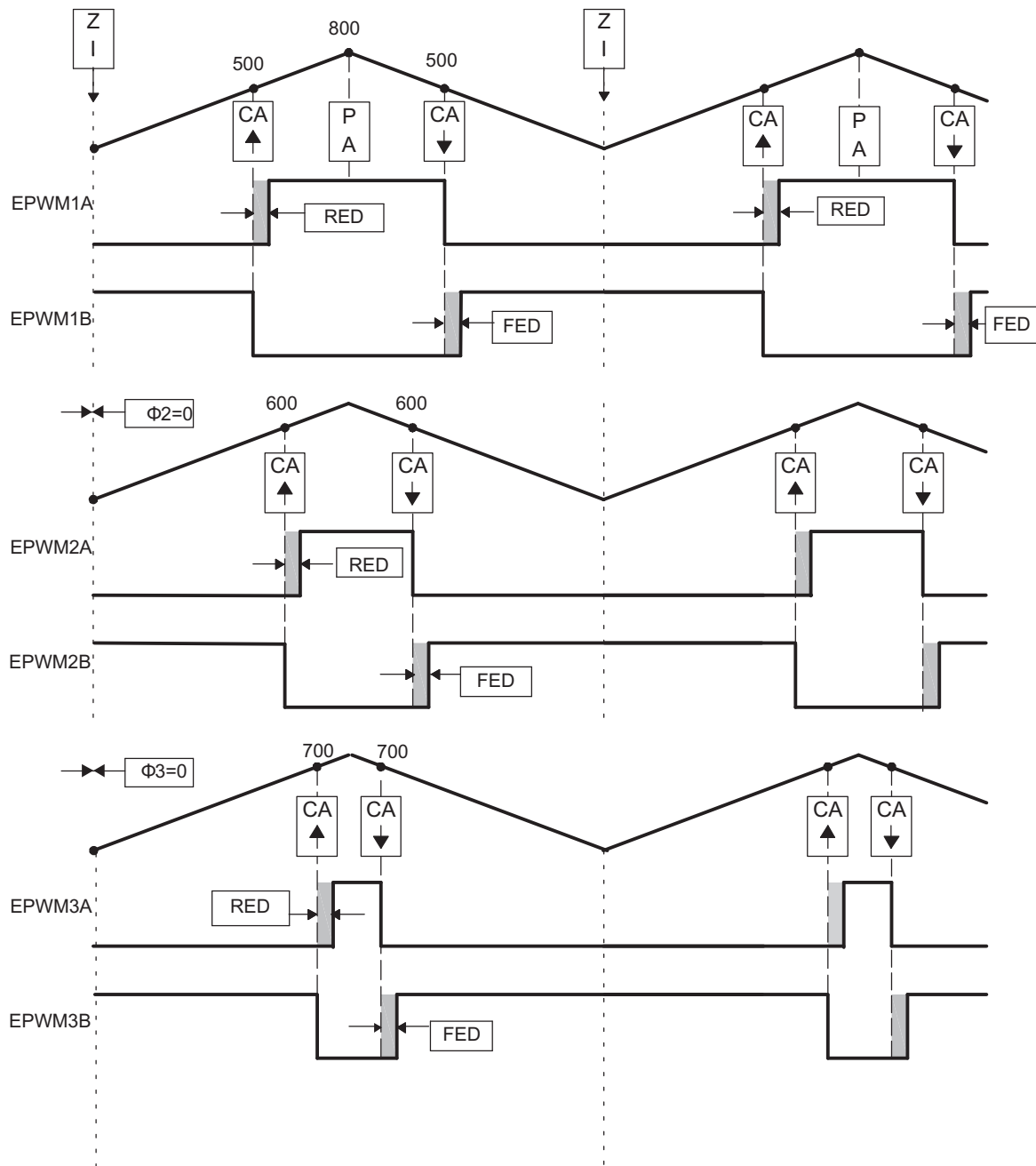


Figure 14-58. 3-Phase Inverter Waveforms for Figure 14-57 (Only One Inverter Shown)



**Table 14-42. EPWM1 Initialization for Figure 14-57**

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 1600 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM1A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	50	RED = 50 TBCLKs

**Table 14-43. EPWM2 Initialization for Figure 14-57**

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 1600 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM2A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	50	RED = 50 TBCLKs

**Table 14-44. EPWM3 Initialization for Figure 14-57**

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 1600 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM3A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	50	RED = 50 TBCLKs

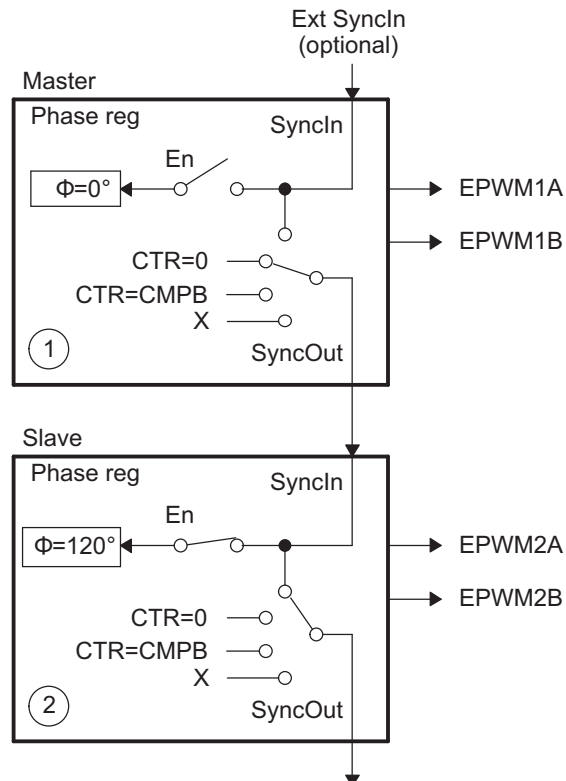
**Example 14-6. Code Snippet for Configuration in Figure 14-57**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 500; // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 600; // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 700; // adjust duty for output EPWM3A
```

### 14.3.7 Practical Applications Using Phase Control Between PWM Modules

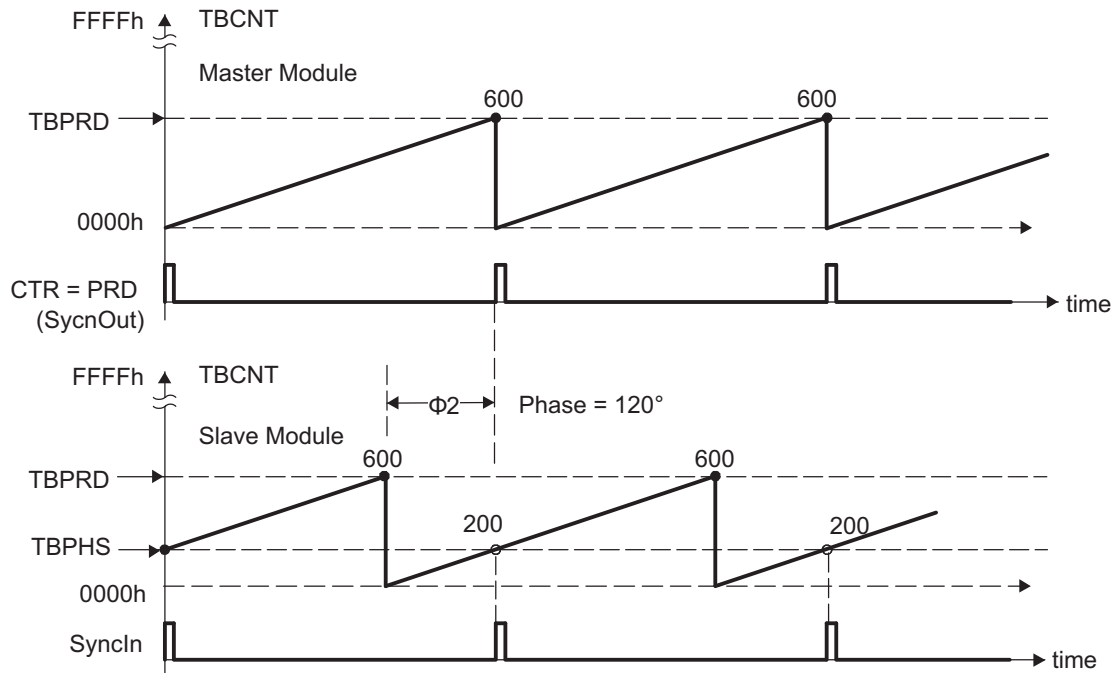
So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or its value has been a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the TB module section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCNT register. To illustrate this concept, [Figure 14-59](#) shows a master and slave module with a phase relationship of 120°, that is, the slave leads the master.

**Figure 14-59. Configuring Two PWM Modules for Phase Control**



[Figure 14-60](#) shows the associated timing waveforms for this configuration. Here,  $TBPRD = 600$  for both master and slave. For the slave,  $TBPHS = 200$  ( $200/600 \times 360^\circ = 120^\circ$ ). Whenever the master generates a SyncIn pulse ( $CTR = PRD$ ), the value of  $TBPHS = 200$  is loaded into the slave TBCNT register so the slave time-base is always leading the master's time-base by 120°.

**Figure 14-60. Timing Waveforms Associated With Phase Control Between 2 Modules**



### 14.3.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in [Figure 14-61](#). This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be  $F = 120^\circ$ . This is achieved by setting the slave TBPHS registers 2 and 3 with values of 1/3 and 2/3 of the period value, respectively. For example, if the period register is loaded with a value of 600 counts, then TBPHS (slave 2) = 200 and TBPHS (slave 3) = 400. Both slave modules are synchronized to the master 1 module.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

$$TBPHS(N,M) = (TBPRD/N) \times (M - 1)$$

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N = 3), TBPRD = 600,

$$TBPHS(3,2) = (600/3) \times (2 - 1) = 200 \times 1 = 200 \text{ (Phase value for Slave module 2)}$$

$$TBPHS(3,3) = (600/3) \times (3 - 1) = 200 \times 2 = 400 \text{ (Phase value for Slave module 3)}$$

[Figure 14-62](#) shows the waveforms for the configuration in [Figure 14-61](#).



Figure 14-61. Control of a 3-Phase Interleaved DC/DC Converter

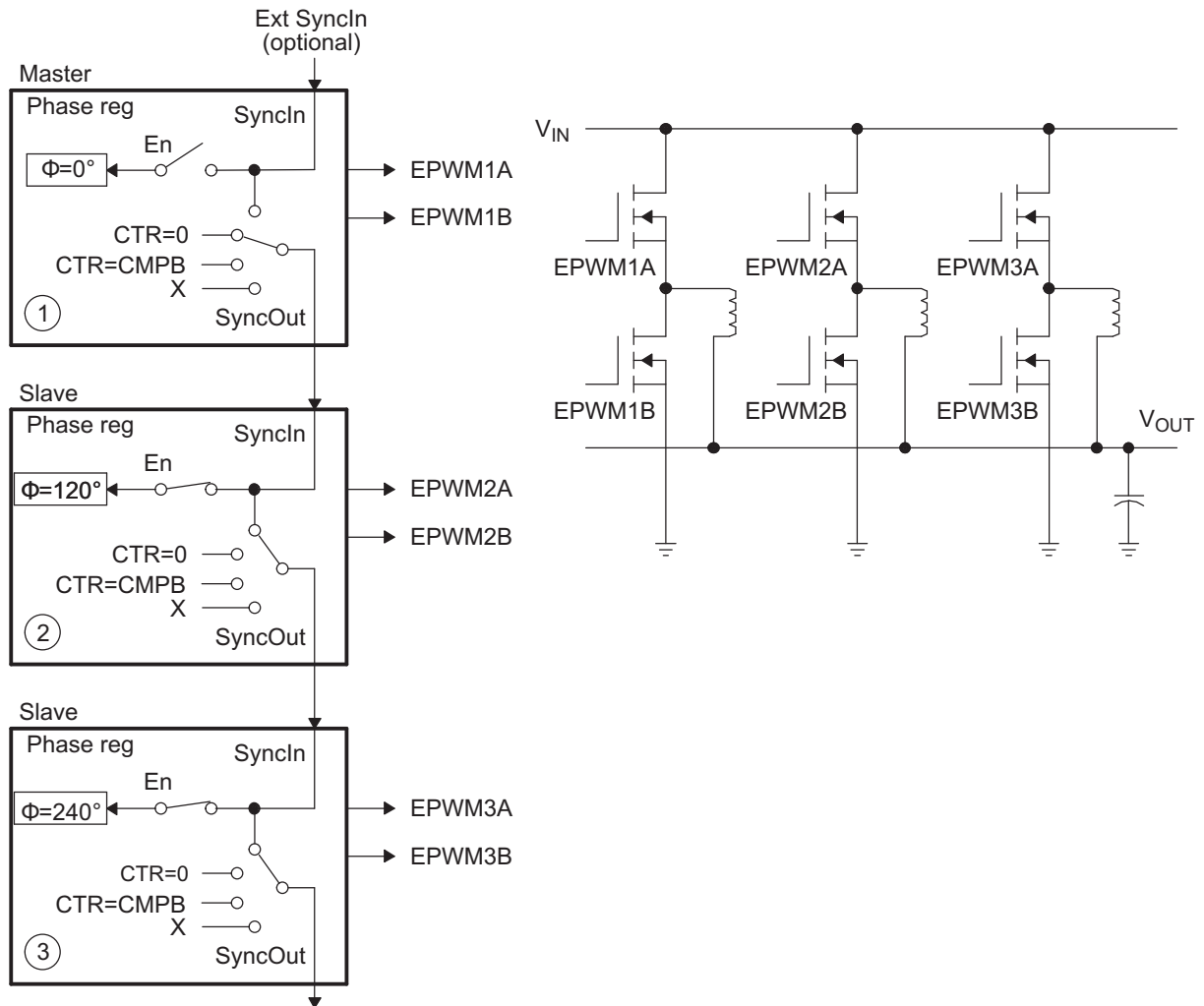
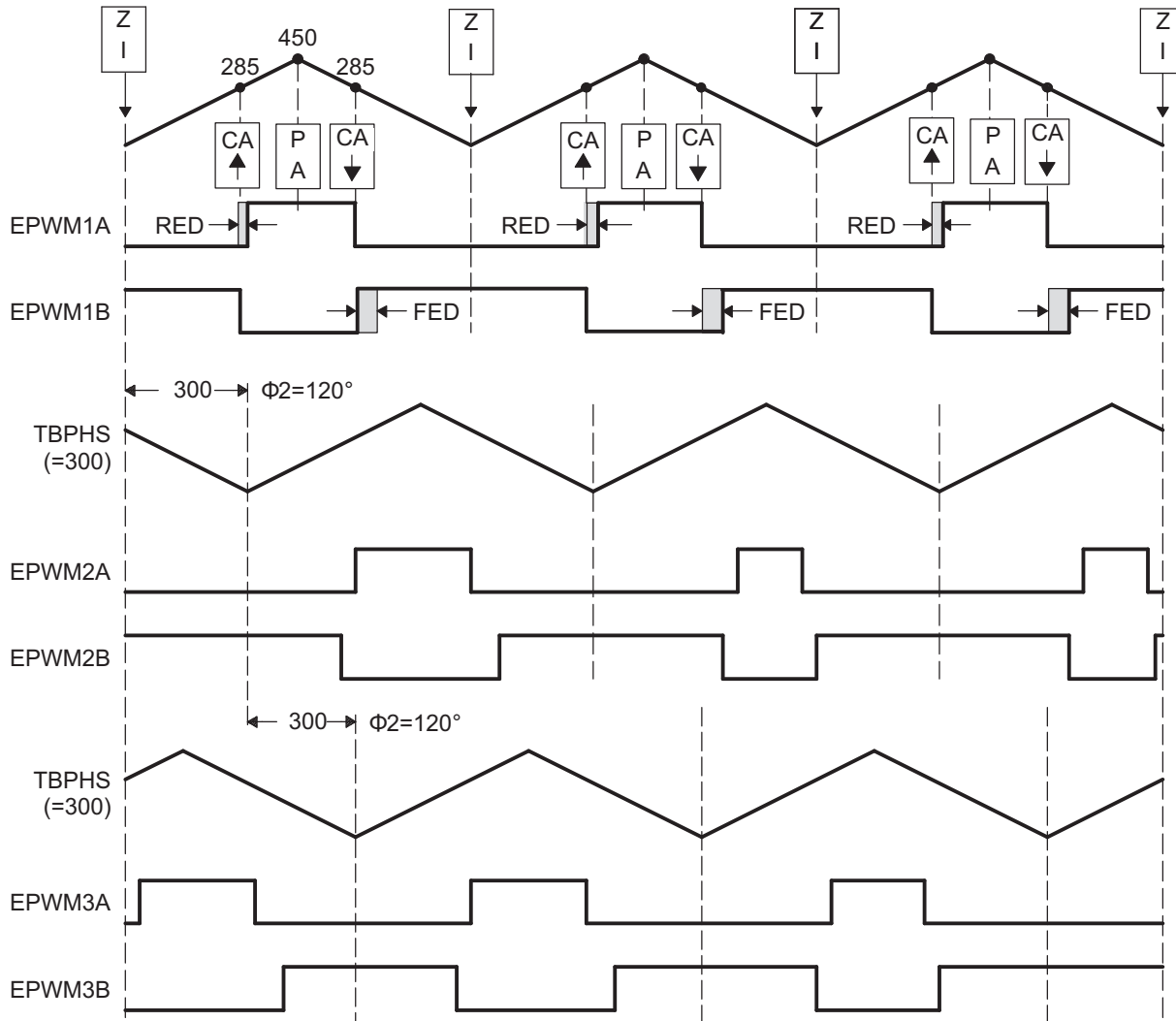


Figure 14-62. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 14-61



**Table 14-45. EPWM1 Initialization for Figure 14-61**

Register	Bit	Value	Comments
TBPRD	TBPRD	450 (1C2h)	Period = 900 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM1A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	20	FED = 20 TBCLKs
	DBRED	20	RED = 20 TBCLKs

**Table 14-46. EPWM2 Initialization for Figure 14-61**

Register	Bit	Value	Comments
TBPRD	TBPRD	450 (1C2h)	Period = 900 TBCLK counts
TBPHS	TBPHS	300	Phase = $(300/900) \times 360 = 120^\circ$
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
	PHSDIR	TB_DOWN	Count DOWN on sync
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM2A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	20	FED = 20 TBCLKs
	DBRED	20	RED = 20 TBCLKs

**Table 14-47. EPWM3 Initialization for Figure 14-61**

Register	Bit	Value	Comments
TBPRD	TBPRD	450 (1C2h)	Period = 900 TBCLK counts
TBPHS	TBPHS	300	Phase = $(300/900) \times 360 = 120^\circ$
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
	PHSDIR	TB_UP	Count UP on sync
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM3A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	20	FED = 20 TBCLKs
	DBRED	20	RED = 20 TBCLKs

**Example 14-7. Code Snippet for Configuration in Figure 14-61**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 285;           // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 285;           // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 285;           // adjust duty for output EPWM3A
```

### 14.3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in Figure 14-63 assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class of power topologies known as *phase-shifted full bridge*, or *zero voltage switched full bridge*. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is the phase relationship between legs. Such a system can be implemented by allocating the resources of two PWM modules to control a single power stage, which in turn requires control of four switching elements. Figure 14-64 shows a master/slave module combination synchronized together to control a full H-bridge. In this case, both master and slave modules are required to switch at the same PWM frequency. The phase is controlled by using the slave's phase register (TBPHS). The master's phase register is not used and therefore can be initialized to zero.

**Figure 14-63. Controlling a Full-H Bridge Stage ( $F_{PWM2} = F_{PWM1}$ )**

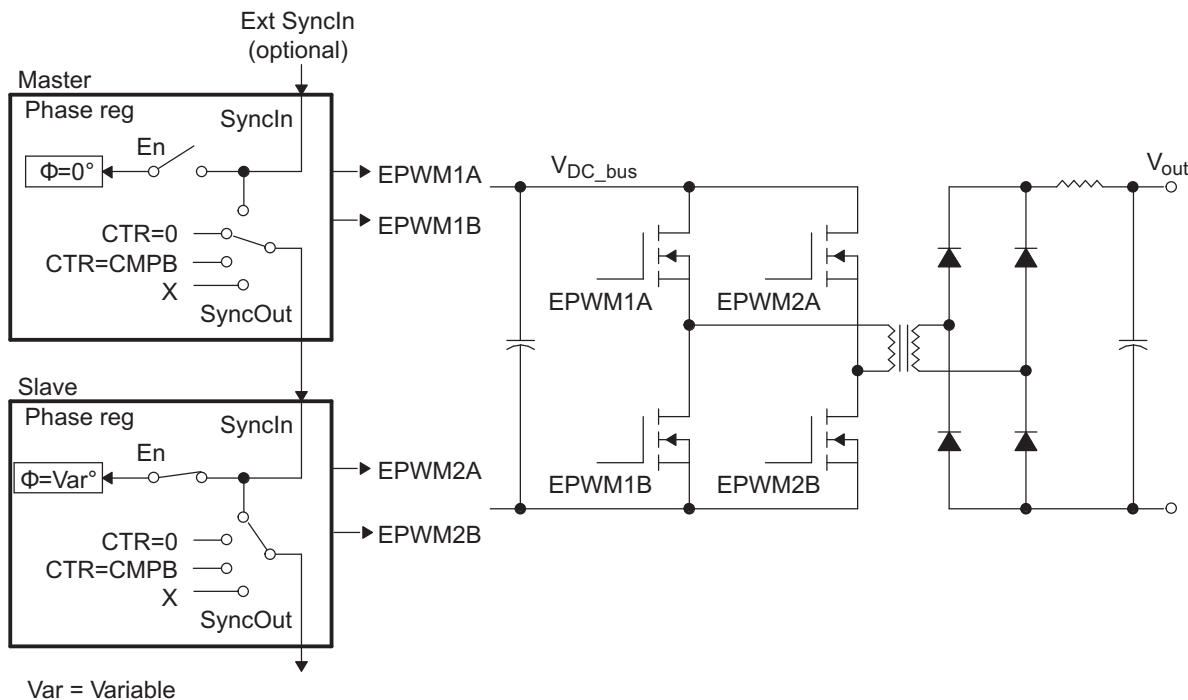
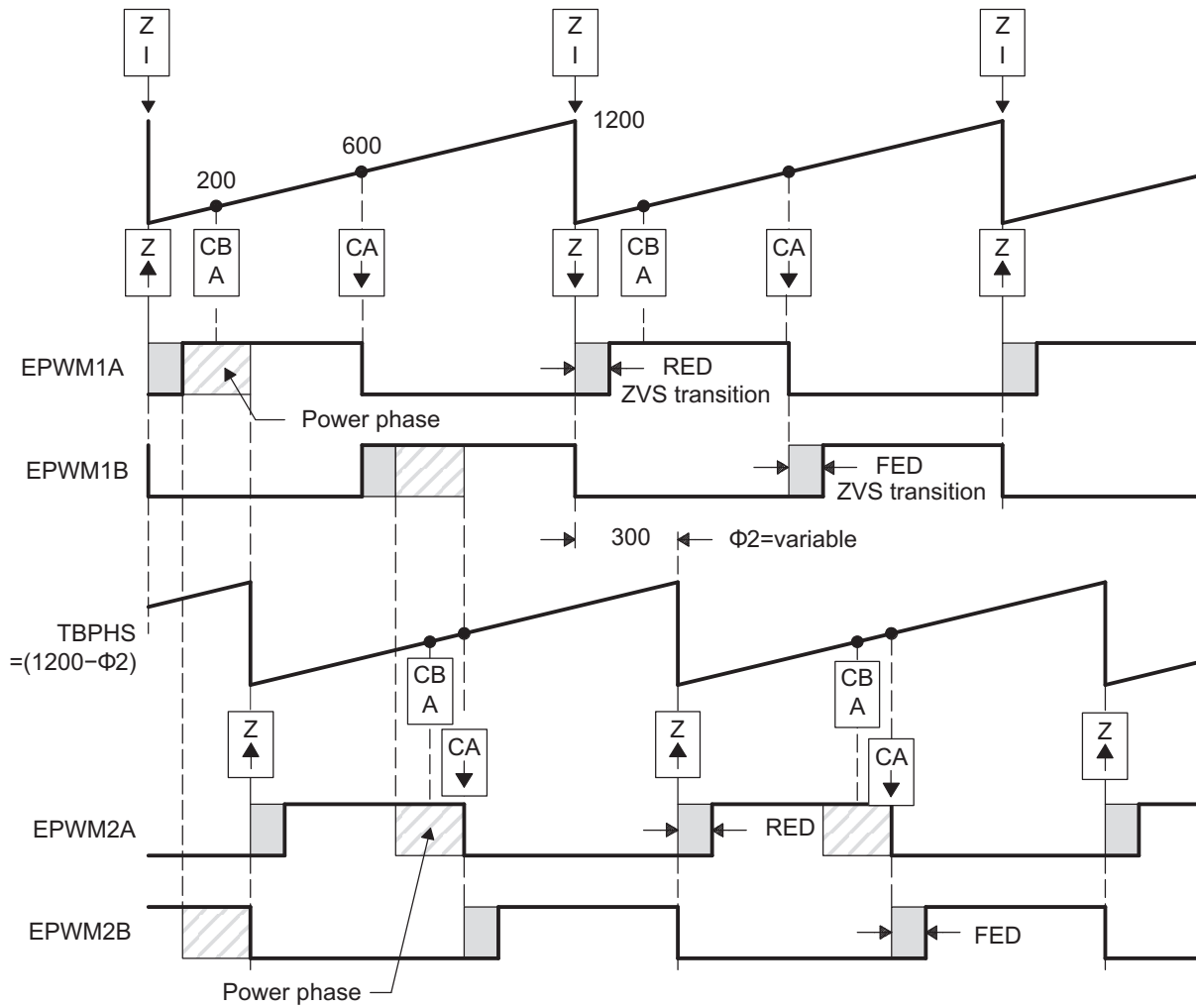


Figure 14-64. ZVS Full-H Bridge Waveforms



**Table 14-48. EPWM1 Initialization for Figure 14-63**

Register	Bit	Value	Comments
TBPRD	TBPRD	1200 (4B0h)	Period = 1201 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPA	CMPA	600 (258h)	Set 50% duty for EPWM1A
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM1A
	CAU	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	70	RED = 70 TBCLKs

**Table 14-49. EPWM2 Initialization for Figure 14-63**

Register	Bit	Value	Comments
TBPRD	TBPRD	1200 (4B0h)	Period = 1201 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_ENABLE	Slave module
	PRDL	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPA	CMPA	600 (258h)	Set 50% duty for EPWM2A
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM2A
	CAU	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	30	FED = 30 TBCLKs
	DBRED	40	RED = 40 TBCLKs

**Example 14-8. Code Snippet for Configuration in Figure 14-63**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm2Regs.TBPHS = 1200-300; // Set Phase reg to 300/1200 * 360 = 90 deg
EPwm1Regs.DBFED = FED1_NewValue; // Update ZVS transition interval
EPwm1Regs.DBRED = RED1_NewValue; // Update ZVS transition interval
EPwm2Regs.DBFED = FED2_NewValue; // Update ZVS transition interval
EPwm2Regs.DBRED = RED2_NewValue; // Update ZVS transition interval
```

## 14.4 Registers

This section includes the registers for the submodules.

**Table 14-50. Submodule Registers**

Submodule	Section
Time-Base Submodule Registers	<a href="#">Section 14.4.1</a>
Counter-Compare Submodule Registers	<a href="#">Section 14.4.2</a>
Action-Qualifier Submodule Registers	<a href="#">Section 14.4.3</a>
Dead-Band Generator Submodule Registers	<a href="#">Section 14.4.4</a>
PWM-Chopper Submodule Registers	<a href="#">Section 14.4.5</a>
Trip-Zone Submodule Registers	<a href="#">Section 14.4.6</a>
Event-Trigger Submodule Registers	<a href="#">Section 14.4.7</a>
High-Resolution PWM Registers	<a href="#">Section 14.4.8</a>

### 14.4.1 Time-Base Submodule Registers

[Table 14-51](#) lists the memory-mapped registers for the time-base submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in [Table 14-51](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-51. Time-Base Submodule Registers**

Offset	Acronym	Register Description	Section
0h	TBCTL	Time-Base Control Register	<a href="#">Section 14.4.1.1</a>
2h	TBSTS	Time-Base Status Register	<a href="#">Section 14.4.1.2</a>
4h	TBPHSHR	Time-Base Phase High-Resolution Register <sup>(1)</sup>	<a href="#">Section 14.4.8.1</a>
6h	TBPHS	Time-Base Phase Register	<a href="#">Section 14.4.1.3</a>
8h	TBCNT	Time-Base Counter Register	<a href="#">Section 14.4.1.4</a>
Ah	TBPRD	Time-Base Period Register	<a href="#">Section 14.4.1.5</a>

<sup>(1)</sup> This register is only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, this location is reserved. See your device-specific data manual to determine which instances include the HRPWM.

#### 14.4.1.1 Time-Base Control Register (TBCTL)

The time-base control register (TBCTL) is shown in [Figure 14-65](#) and described in [Table 14-52](#).

**Figure 14-65. Time-Base Control Register (TBCTL)**

15	14	13	12	10	9	8
FREE, SOFT		PHSDIR		CLKDIV		HSPCLKDIV
R/W-0		R/W-0		R/W-0		R/W-0
7	6	5	4	3	2	1
HSPCLKDIV	SWFSYNC	SYNCOSEL	PRDLD	PHSEN		CTRMODE
R/W-1	R/W-0	R/W-0	R/W-0	R/W-0		R/W-3h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset



**Table 14-52. Time-Base Control Register (TBCTL) Field Descriptions**

Bit	Field	Value	Description
15-14	FREE, SOFT	0-3h 0 1h 2h-3h	<p>Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events:</p> <ul style="list-style-type: none"> <li>0 Stop after the next time-base counter increment or decrement</li> <li>1h Stop when counter completes a whole cycle: <ul style="list-style-type: none"> <li>• Up-count mode: stop when the time-base counter = period (TBCNT = TBPRD)</li> <li>• Down-count mode: stop when the time-base counter = 0000 (TBCNT = 0000h)</li> <li>• Up-down-count mode: stop when the time-base counter = 0000 (TBCNT = 0000h)</li> </ul> </li> <li>2h-3h Free run</li> </ul>
13	PHSDIR	0 1	<p>Phase Direction Bit. This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCNT) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event..</p> <p>In the up-count and down-count modes this bit is ignored.</p> <ul style="list-style-type: none"> <li>0 Count down after the synchronization event.</li> <li>1 Count up after the synchronization event.</li> </ul>
12:10	CLKDIV	0-7h 0 1h 2h 3h 4h 5h 6h 7h	<p>Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value. <math>TBCLK = SYSCLKOUT / (HSPCLKDIV \times CLKDIV)</math></p> <ul style="list-style-type: none"> <li>0 /1 (default on reset)</li> <li>1h /2</li> <li>2h /4</li> <li>3h /8</li> <li>4h /16</li> <li>5h /32</li> <li>6h /64</li> <li>7h /128</li> </ul>
9-7	HSPCLKDIV	0-7h 0 1h 2h 3h 4h 5h 6h 7h	<p>High-Speed Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value. <math>TBCLK = SYSCLKOUT / (HSPCLKDIV \times CLKDIV)</math></p> <p>This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.</p> <ul style="list-style-type: none"> <li>0 /1</li> <li>1h /2 (default on reset)</li> <li>2h /4</li> <li>3h /6</li> <li>4h /8</li> <li>5h /10</li> <li>6h /12</li> <li>7h /14</li> </ul>
6	SWFSYNC	0 1	<p>Software Forced Synchronization Pulse</p> <ul style="list-style-type: none"> <li>0 Writing a 0 has no effect and reads always return a 0.</li> <li>1 Writing a 1 forces a one-time synchronization pulse to be generated.</li> </ul> <p>This event is ORed with the EPWMxSYNCl input of the ePWM module.</p> <p>SWFSYNC is valid (operates) only when EPWMxSYNCl is selected by SYNCOSSEL = 00.</p>
5-4	SYNCOSSEL	0-3h 0 1h 2h 3h	<p>Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.</p> <ul style="list-style-type: none"> <li>0 EPWMxSYNCO:</li> <li>1h CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)</li> <li>2h CTR = CMPB : Time-base counter equal to counter-compare B (TBCNT = CMPB)</li> <li>3h Disable EPWMxSYNCO signal</li> </ul>

**Table 14-52. Time-Base Control Register (TBCTL) Field Descriptions (continued)**

Bit	Field	Value	Description
3	PRDL	0 1	Active Period Register Load From Shadow Register Select The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCNT, is equal to zero. A write or read to the TBPRD register accesses the shadow register. Load the TBPRD register immediately without using a shadow register. A write or read to the TBPRD register directly accesses the active register.
2	PHSEN	0 1	Counter Register Load From Phase Register Enable Do not load the time-base counter (TBCNT) from the time-base phase register (TBPHS) Load the time-base counter with the phase register when an EPWMxSYNCl input signal occurs or when a software synchronization is forced by the SWFSYNC bit.
1-0	CTRM	0-3h 0 1h 2h 3h	Counter Mode. The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change. These bits set the time-base counter mode of operation as follows: Up-count mode Down-count mode Up-down-count mode Stop-freeze counter operation (default on reset)

#### 14.4.1.2 Time-Base Status Register (TBSTS)

The time-base status register (TBSTS) is shown in [Figure 14-66](#) and described in [Table 14-53](#).

**Figure 14-66. Time-Base Status Register (TBSTS)**

15	3	2	1	0
Reserved	CTRMAX	SYNCl	CTRDIR	
R-0	R/W1C-0	R/W1C-0	R-1	

LEGEND: R/W = Read/Write; R/W1C = Read/Write 1 to clear; -n = value after reset

**Table 14-53. Time-Base Status Register (TBSTS) Field Descriptions**

Bit	Field	Value	Description
15-3	Reserved	0	Reserved
2	CTRMAX	0 1	Time-Base Counter Max Latched Status Bit Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect. Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit will clear the latched event.
1	SYNCl	0 1	Input Synchronization Latched Status Bit Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred. Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCl). Writing a 1 to this bit will clear the latched event.
0	CTRDIR	0 1	Time-Base Counter Direction Status Bit. At reset, the counter is frozen; therefore, this bit has no meaning. To make this bit meaningful, you must first set the appropriate mode via TBCTL[CTRM]. Time-Base Counter is currently counting down. Time-Base Counter is currently counting up.

#### 14.4.1.3 Time-Base Phase Register (TBPHS)

The time-base phase register (TBPHS) is shown in [Figure 14-67](#) and described in [Table 14-54](#).

**Figure 14-67. Time-Base Phase Register (TBPHS)**

15	0
TBPHS	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

**Table 14-54. Time-Base Phase Register (TBPHS) Field Descriptions**

Bits	Name	Value	Description
15-0	TBPHS	0-FFFFh	<p>These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal.</p> <ul style="list-style-type: none"> <li>If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase.</li> <li>If TBCTL[PHSEN] = 1, then the time-base counter (TBCNT) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCI) or by a software forced synchronization.</li> </ul>

#### 14.4.1.4 Time-Base Counter Register (TBCNT)

The time-base counter register (TBCNT) is shown in [Figure 14-68](#) and described in [Table 14-55](#).

**Figure 14-68. Time-Base Counter Register (TBCNT)**

15	0
TBCNT	
R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

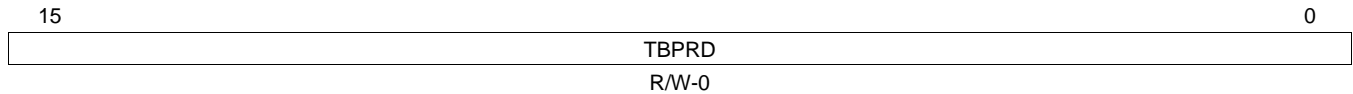
**Table 14-55. Time-Base Counter Register (TBCNT) Field Descriptions**

Bits	Name	Value	Description
15-0	TBCNT	0-FFFFh	<p>Reading these bits gives the current time-base counter value.</p> <p>Writing to these bits sets the current time-base counter value. The update happens as soon as the write occurs; the write is NOT synchronized to the time-base clock (TBCLK) and the register is not shadowed.</p>

### 14.4.1.5 Time-Base Period Register (TBPRD)

The time-base period register (TBPRD) is shown in [Figure 14-69](#) and described in [Table 14-56](#).

**Figure 14-69. Time-Base Period Register (TBPRD)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 14-56. Time-Base Period Register (TBPRD) Field Descriptions**

Bits	Name	Value	Description
15-0	TBPRD	0-FFFFh	<p>These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals zero.</li> <li>If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>The active and shadow registers share the same memory map address.</li> </ul>

### 14.4.2 Counter-Compare Submodule Registers

[Table 14-57](#) lists the memory-mapped registers for the counter-compare submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in [Table 14-57](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-57. Counter-Compare Submodule Registers**

Offset	Acronym	Register Description	Section
Eh	CMPCTL	Counter-Compare Control Register	<a href="#">Section 14.4.2.1</a>
10h	CMPAHR	Counter-Compare A High-Resolution Register <sup>(1)</sup>	<a href="#">Section 14.4.8.2</a>
12h	CMPA	Counter-Compare A Register	<a href="#">Section 14.4.2.2</a>
14h	CMPB	Counter-Compare B Register	<a href="#">Section 14.4.2.3</a>

<sup>(1)</sup> This register is only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, this location is reserved. See your device-specific data manual to determine which instances include the HRPWM.

### 14.4.2.1 Counter-Compare Control Register (CMPCTL)

The counter-compare control register (CMPCTL) is shown in [Figure 14-70](#) and described in [Table 14-58](#).

**Figure 14-70. Counter-Compare Control Register (CMPCTL)**

15 Reserved				10		9		8							
						SHDWBFULL	SHDWAFULL								
R-0						R-0		R-0							
7		6		5		4		3		2		1		0	
Reserved		SHDWBMODE		Reserved		SHDWAMODE		LOADBMODE				LOADAMODE			
R-0		R/W-0		R-0		R/W-0		R/W-0				R/W-0			

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

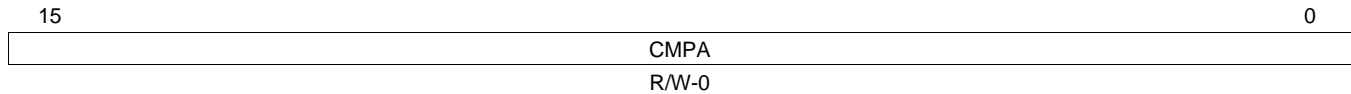
**Table 14-58. Counter-Compare Control Register (CMPCTL) Field Descriptions**

Bits	Name	Value	Description
15-10	Reserved	0	Reserved
9	SHDWBFULL		Counter-compare B (CMPB) Shadow Register Full Status Flag. This bit self clears once a load-strobe occurs.
		0	CMPB shadow FIFO not full yet
		1	Indicates the CMPB shadow FIFO is full; a CPU write will overwrite current shadow value.
8	SHDWAFULL		Counter-compare A (CMPA) Shadow Register Full Status Flag. The flag bit is set when a 32-bit write to CMPA:CMPAHR register or a 16-bit write to CMPA register is made. A 16-bit write to CMPAHR register will not affect the flag. This bit self clears once a load-strobe occurs.
		0	CMPA shadow FIFO not full yet
		1	Indicates the CMPA shadow FIFO is full, a CPU write will overwrite the current shadow value.
7	Reserved	0	Reserved
6	SHDWBMODE		Counter-compare B (CMPB) Register Operating Mode
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action.
5	Reserved		Reserved
4	SHDWAMODE		Counter-compare A (CMPA) Register Operating Mode
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action
3-2	LOADBMODE	0-3h	Active Counter-Compare B (CMPB) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).
		0	Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
		1h	Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)
		2h	Load on either CTR = 0 or CTR = PRD
		3h	Freeze (no loads possible)
1-0	LOADAMODE	0-3h	Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).
		0	Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
		1h	Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)
		2h	Load on either CTR = 0 or CTR = PRD
		3h	Freeze (no loads possible)

### 14.4.2.2 Counter-Compare A Register (CMPA)

The counter-compare A register (CMPA) is shown in [Figure 14-71](#) and described in [Table 14-59](#).

**Figure 14-71. Counter-Compare A Register (CMPA)**



LEGEND: R/W = Read/Write; -n = value after reset

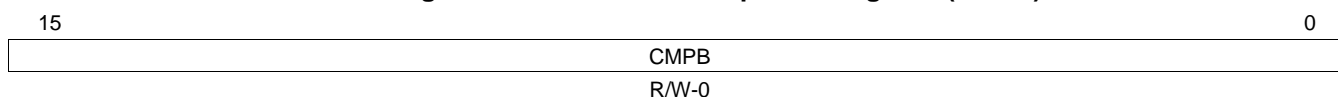
**Table 14-59. Counter-Compare A Register (CMPA) Field Descriptions**

Bits	Name	Value	Description
15-0	CMPA	0-FFFFh	<p>The value in the active CMPA register is continuously compared to the time-base counter (TBCNT). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>• Do nothing; the event is ignored.</li> <li>• Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>• Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>• Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>• If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register.</li> <li>• Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full.</li> <li>• If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>• In either mode, the active and shadow registers share the same memory map address.</li> </ul>

### 14.4.2.3 Counter-Compare B Register (CMPB)

The counter-compare B register (CMPB) is shown in [Figure 14-72](#) and described in [Table 14-60](#).

**Figure 14-72. Counter-Compare B Register (CMPB)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 14-60. Counter-Compare B Register (CMPB) Field Descriptions**

Bits	Name	Value	Description
15-0	CMPB	0-FFFFh	<p>The value in the active CMPB register is continuously compared to the time-base counter (TBCNT). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare B" event. This event is sent to the action-qualifier where it is qualified and converted into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>• Do nothing. event is ignored.</li> <li>• Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>• Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>• Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWBMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>• If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register:</li> <li>• Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full.</li> <li>• If CMPCTL[SHDWBMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>• In either mode, the active and shadow registers share the same memory map address.</li> </ul>

### 14.4.3 Action-Qualifier Submodule Registers

[Table 14-61](#) lists the memory-mapped registers for the action-qualifier submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in [Table 14-61](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-61. Action-Qualifier Submodule Registers**

Offset	Acronym	Register Description	Section
16h	AQCTLA	Action-Qualifier Output A Control Register	<a href="#">Section 14.4.3.1</a>
18h	AQCTLB	Action-Qualifier Output B Control Register	<a href="#">Section 14.4.3.2</a>
1Ah	AQSFR	Action-Qualifier Software Force Register	<a href="#">Section 14.4.3.3</a>
1Ch	AQCSFR	Action-Qualifier Continuous Software Force Register	<a href="#">Section 14.4.3.4</a>

### 14.4.3.1 Action-Qualifier Output A Control Register (AQCTLA)

The action-qualifier output A control register (AQCTLA) is shown in [Figure 14-73](#) and described in [Table 14-62](#).

**Figure 14-73. Action-Qualifier Output A Control Register (AQCTLA)**

15	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved				CBD		CBU		CAD		CAU		PRD		ZRO
R-0				R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-62. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions**

Bits	Name	Value	Description
15-12	Reserved	0	Reserved
11-10	CBD	0-3h	Action when the time-base counter equals the active CMPB register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.		
9-8	CBU	0-3h	Action when the counter equals the active CMPB register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.		
7-6	CAD	0-3h	Action when the counter equals the active CMPA register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.		
5-4	CAU	0-3h	Action when the counter equals the active CMPA register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.		
3-2	PRD	0-3h	Action when the counter equals the period. Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.		
1-0	ZRO	0-3h	Action when counter equals zero. Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.		



### 14.4.3.2 Action-Qualifier Output B Control Register (AQCTLB)

The action-qualifier output B control register (AQCTLB) is shown in [Figure 14-74](#) and described in [Table 14-63](#).

**Figure 14-74. Action-Qualifier Output B Control Register (AQCTLB)**

15	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved				CBD		CBU		CAD		CAU		PRD		ZRO
R-0				R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-63. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions**

Bits	Name	Value	Description
15-12	Reserved	0	Reserved
11-10	CBD	0-3h	Action when the counter equals the active CMPB register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
9-8	CBU	0-3h	Action when the counter equals the active CMPB register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
7-6	CAD	0-3h	Action when the counter equals the active CMPA register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
5-4	CAU	0-3h	Action when the counter equals the active CMPA register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
3-2	PRD	0-3h	Action when the counter equals the period. Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
1-0	ZRO	0-3h	Action when counter equals zero. Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.

### 14.4.3.3 Action-Qualifier Software Force Register (AQSFRC)

The action-qualifier software force register (AQSFRC) is shown in [Figure 14-75](#) and described in [Table 14-64](#).

**Figure 14-75. Action-Qualifier Software Force Register (AQSFRC)**

15	8	7	6	5	4	3	2	1	0
Reserved		RLDCSF	OTSFB	ACTSFB	OTSFA			ACTSFA	
R-0		R/W-0	R/W-0	R/W-0	R/W-0			R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-64. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions**

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7-6	RLDCSF	0-3h	AQSFRC Active Register Reload From Shadow Options
		0	Load on event counter equals zero
		1h	Load on event counter equals period
		2h	Load on event counter equals zero or counter equals period
		3h	Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register).
5	OTSFB	0	One-Time Software Forced Event on Output B
		0	Writing a 0 (zero) has no effect. Always reads back a 0
			This bit is auto cleared once a write to this register is complete, that is, a forced event is initiated.)
			This is a one-shot forced event. It can be overridden by another subsequent event on output B.
		1	Initiates a single s/w forced event
4-3	ACTSFB	0-3h	Action when One-Time Software Force B Is Invoked
		0	Does nothing (action disabled)
		1h	Clear (low)
		2h	Set (high)
		3h	Toggle (Low -> High, High -> Low)
			<b>Note:</b> This action is not qualified by counter direction (CNT_dir)
2	OTSFA	0	One-Time Software Forced Event on Output A
		0	Writing a 0 (zero) has no effect. Always reads back a 0.
			This bit is auto cleared once a write to this register is complete (that is, a forced event is initiated).
		1	Initiates a single software forced event
1-0	ACTSFA	0-3h	Action When One-Time Software Force A Is Invoked
		0	Does nothing (action disabled)
		1h	Clear (low)
		2h	Set (high)
		3h	Toggle (Low → High, High → Low)
			<b>Note:</b> This action is not qualified by counter direction (CNT_dir)

#### 14.4.3.4 Action-Qualifier Continuous Software Force Register (AQCSFRC)

The action-qualifier continuous software force register (AQCSFRC) is shown in [Figure 14-76](#) and described in [Table 14-65](#).

**Figure 14-76. Action-Qualifier Continuous Software Force Register (AQCSFRC)**

15	4	3	2	1	0
Reserved				CSFB	CSFA
R-0				R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-65. Action-Qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions**

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3-2	CSFB	0-3h	Continuous Software Force on Output B In immediate mode, a continuous force takes effect on the next TBCLK edge. In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register. To configure shadow mode, use AQSFRC[RLDCSF].
		0	Forcing disabled, that is, has no effect
		1h	Forces a continuous low on output B
		2h	Forces a continuous high on output B
		3h	Software forcing is disabled and has no effect
1-0	CSFA	0-3h	Continuous Software Force on Output A In immediate mode, a continuous force takes effect on the next TBCLK edge. In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.
		0	Forcing disabled, that is, has no effect
		1h	Forces a continuous low on output A
		2h	Forces a continuous high on output A
		3h	Software forcing is disabled and has no effect

#### 14.4.4 Dead-Band Generator Submodule Registers

[Table 14-66](#) lists the memory-mapped registers for the dead-band generator submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in [Table 14-66](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-66. Dead-Band Generator Submodule Registers**

Offset	Acronym	Register Description	Section
1Eh	DBCTL	Dead-Band Generator Control Register	<a href="#">Section 14.4.4.1</a>
20h	DBRED	Dead-Band Generator Rising Edge Delay Register	<a href="#">Section 14.4.4.2</a>
22h	DBFED	Dead-Band Generator Falling Edge Delay Register	<a href="#">Section 14.4.4.3</a>

#### 14.4.4.1 Dead-Band Generator Control Register (DBCTL)

The dead-band generator control register (DBCTL) is shown in [Figure 14-77](#) and described in [Table 14-67](#).

**Figure 14-77. Dead-Band Generator Control Register (DBCTL)**

15	6	5	4	3	2	1	0
Reserved		IN_MODE	POLSEL	OUT_MODE			
R-0		R/W-0	R/W-0	R/W-0			

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

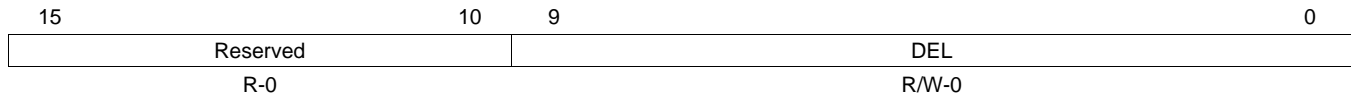
**Table 14-67. Dead-Band Generator Control Register (DBCTL) Field Descriptions**

Bits	Name	Value	Description
15-6	Reserved	0	Reserved
5-4	IN_MODE	0-3h	<p>Dead Band Input Mode Control. Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown in <a href="#">Figure 14-29</a>. This allows you to select the input source to the falling-edge and rising-edge delay. To produce classical dead-band waveforms, the default is EPWMxA In is the source for both falling and rising-edge delays.</p> <p>0 EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.</p> <p>1h EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.</p> <p>2h EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.</p> <p>3h EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.</p> <p>EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.</p> <p>3h EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.</p>
3-2	POLSEL	0-3h	<p>Polarity Select Control. Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in <a href="#">Figure 14-29</a>. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule.</p> <p>The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter.</p> <p>These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.</p> <p>0 Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).</p> <p>1h Active low complementary (ALC) mode. EPWMxA is inverted.</p> <p>2h Active high complementary (AHC). EPWMxB is inverted.</p> <p>3h Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.</p>
1-0	OUT_MODE	0-3h	<p>Dead-band Output Mode Control. Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in <a href="#">Figure 14-29</a>. This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.</p> <p>0 Dead-band generation is bypassed for both output signals. In this mode, both the EPWMxA and EPWMxB output signals from the action-qualifier are passed directly to the PWM-chopper submodule. In this mode, the POLSEL and IN_MODE bits have no effect.</p> <p>1h Disable rising-edge delay. The EPWMxA signal from the action-qualifier is passed straight through to the EPWMxA input of the PWM-chopper submodule. The falling-edge delayed signal is seen on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].</p> <p>2h Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule. The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE].</p> <p>3h Dead-band is fully enabled for both rising-edge delay on output EPWMxA and falling-edge delay on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].</p>

#### 14.4.4.2 Dead-Band Generator Rising Edge Delay Register (DBRED)

The dead-band generator rising edge delay register (DBRED) is shown in [Figure 14-78](#) and described in [Table 14-68](#).

**Figure 14-78. Dead-Band Generator Rising Edge Delay Register (DBRED)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

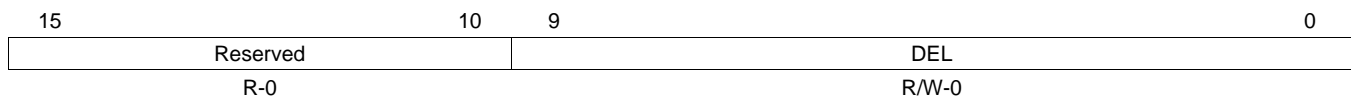
**Table 14-68. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions**

Bits	Name	Value	Description
15-10	Reserved	0	Reserved
9-0	DEL	0-3FFh	Rising Edge Delay Count. 10-bit counter.

#### 14.4.4.3 Dead-Band Generator Falling Edge Delay Register (DBFED)

The dead-band generator falling edge delay register (DBFED) is shown in [Figure 14-79](#) and described in [Table 14-69](#).

**Figure 14-79. Dead-Band Generator Falling Edge Delay Register (DBFED)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-69. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions**

Bits	Name	Value	Description
15-10	Reserved	0	Reserved
9-0	DEL	0-3FFh	Falling Edge Delay Count. 10-bit counter

### 14.4.5 PWM-Chopper Submodule Register

The PWM-chopper control register (PCCTL) is shown in [Figure 14-80](#) and described in [Table 14-70](#).

**Figure 14-80. PWM-Chopper Control Register (PCCTL)**

15	11	10	8	7	5	4	1	0
Reserved		CHPDUTY		CHPFREQ		OSHTWTH		CHPEN
R-0		R/W-0		R/W-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-70. PWM-Chopper Control Register (PCCTL) Bit Descriptions**

Bits	Name	Value	Description
15-11	Reserved	0	Reserved
10-8	CHPDUTY	0-7h	Chopping Clock Duty Cycle
		0	Duty = 1/8 (12.5%)
		1h	Duty = 2/8 (25.0%)
		2h	Duty = 3/8 (37.5%)
		3h	Duty = 4/8 (50.0%)
		4h	Duty = 5/8 (62.5%)
		5h	Duty = 6/8 (75.0%)
		6h	Duty = 7/8 (87.5%)
7h	Reserved		
7-5	CHPFREQ	0-7h	Chopping Clock Frequency
		0	Divide by 1 (no prescale)
		1h	Divide by 2
		2h	Divide by 3
3h-7h	Divide by 4 to divide by 8		
4-1	OSHTWTH	0-Fh	One-Shot Pulse Width
		0	1 × SYSCLKOUT/8 wide
		1h	2 × SYSCLKOUT/8 wide
		2h	3 × SYSCLKOUT/8 wide
3h-Fh	4 × SYSCLKOUT/8 wide to 16 × SYSCLKOUT/8 wide		
0	CHPEN		PWM-chopping Enable
		0	Disable (bypass) PWM chopping function
		1	Enable chopping function

### 14.4.6 Trip-Zone Submodule Registers

Table 14-71 lists the memory-mapped registers for the trip-zone submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 14-71 should be considered as reserved locations and the register contents should not be modified.

**Table 14-71. Trip-Zone Submodule Registers**

Offset	Acronym	Register Description	Section
24h	TZSEL	Trip-Zone Select Register	<a href="#">Section 14.4.6.1</a>
28h	TZCTL	Trip-Zone Control Register	<a href="#">Section 14.4.6.2</a>
2Ah	TZEINT	Trip-Zone Enable Interrupt Register	<a href="#">Section 14.4.6.3</a>
2Ch	TZFLG	Trip-Zone Flag Register	<a href="#">Section 14.4.6.4</a>
2Eh	TZCLR	Trip-Zone Clear Register	<a href="#">Section 14.4.6.5</a>
30h	TZFRC	Trip-Zone Force Register	<a href="#">Section 14.4.6.6</a>

#### 14.4.6.1 Trip-Zone Select Register (TZSEL)

The trip-zone select register (TZSEL) is shown in Figure 14-81 and described in Table 14-72.

**Figure 14-81. Trip-Zone Select Register (TZSEL)**

15	9	8	7	1	0
Reserved/OSHT $n$ <sup>(1)</sup>		OSHT1	Reserved/CBC $n$ <sup>(1)</sup>		CBC1
R/W-0		R/W-0	R/W-0		R/W-0

LEGEND: R/W = Read/Write; - $n$  = value after reset

<sup>(1)</sup> Number of register bits depends on how many trip-zone pins are available in the device. See your device-specific data manual.

**Table 14-72. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions**

Bits	Name	Value	Description
15-8	OSHT $n$	0 1	<p>Trip-zone <math>n</math> (<math>\overline{TZn}</math>) select. One-Shot (OSHT) trip-zone enable/disable. When any of the enabled pins go low, a one-shot trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register (<a href="#">Section 14.4.6.2</a>) is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until you clear the condition via the TZCLR register (<a href="#">Section 14.4.6.5</a>).</p> <p>0 Disable <math>\overline{TZn}</math> as a one-shot trip source for this ePWM module.</p> <p>1 Enable <math>\overline{TZn}</math> as a one-shot trip source for this ePWM module.</p>
7-0	CBC $n$	0 1	<p>Trip-zone <math>n</math> (<math>\overline{TZn}</math>) select. Cycle-by-Cycle (CBC) trip-zone enable/disable. When any of the enabled pins go low, a cycle-by-cycle trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register (<a href="#">Section 14.4.6.2</a>) is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.</p> <p>0 Disable <math>\overline{TZn}</math> as a CBC trip source for this ePWM module.</p> <p>1 Enable <math>\overline{TZn}</math> as a CBC trip source for this ePWM module.</p>

### 14.4.6.2 Trip-Zone Control Register (TZCTL)

The trip-zone control register (TZCTL) is shown in [Figure 14-82](#) and described in [Table 14-73](#).

**Figure 14-82. Trip-Zone Control Register (TZCTL)**

15	Reserved	4	3	2	1	0
	R-0			TZB	TZB	TZA
				R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-73. Trip-Zone Control Register (TZCTL) Field Descriptions**

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3-2	TZB	0-3h	When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register ( <a href="#">Section 14.4.6.1</a> ).
		0	High impedance (EPWMxB = High-impedance state)
		1h	Force EPWMxB to a high state
		2h	Force EPWMxB to a low state
		3h	Do nothing, no action is taken on EPWMxB.
1-0	TZA	0-3h	When a trip event occurs the following action is taken on output EPWMxA. Which trip-zone pins can cause an event is defined in the TZSEL register ( <a href="#">Section 14.4.6.1</a> ).
		0	High impedance (EPWMxA = High-impedance state)
		1h	Force EPWMxA to a high state
		2h	Force EPWMxA to a low state
		3h	Do nothing, no action is taken on EPWMxA.

### 14.4.6.3 Trip-Zone Enable Interrupt Register (TZEINT)

The trip-zone enable interrupt register (TZEINT) is shown in [Figure 14-83](#) and described in [Table 14-74](#).

**Figure 14-83. Trip-Zone Enable Interrupt Register (TZEINT)**

15	Reserved	3	2	1	0
	R-0		OST	CBC	Rsvd
			R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-74. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions**

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
2	OST	0-1	Trip-zone One-Shot Interrupt Enable
		0	Disable one-shot interrupt generation
		1	Enable Interrupt generation; a one-shot trip event will cause a EPWMxTZINT interrupt.
1	CBC	0-1	Trip-zone Cycle-by-Cycle Interrupt Enable
		0	Disable cycle-by-cycle interrupt generation.
		1	Enable interrupt generation; a cycle-by-cycle trip event will cause an EPWMxTZINT interrupt.
0	Reserved	0	Reserved



#### 14.4.6.4 Trip-Zone Flag Register (TZFLG)

The trip-zone flag register (TZFLG) is shown in [Figure 14-84](#) and described in [Table 14-75](#).

**Figure 14-84. Trip-Zone Flag Register (TZFLG)**

15	Reserved	3	2	1	0
	R-0		OST	CBC	INT
			R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-75. Trip-Zone Flag Register (TZFLG) Field Descriptions**

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
2	OST	0 1	Latched Status Flag for A One-Shot Trip Event. 0 No one-shot trip event has occurred. 1 Indicates a trip event has occurred on a pin selected as a one-shot trip source. This bit is cleared by writing the appropriate value to the TZCLR register ( <a href="#">Section 14.4.6.5</a> ).
1	CBC	0 1	Latched Status Flag for Cycle-By-Cycle Trip Event 0 No cycle-by-cycle trip event has occurred. 1 Indicates a trip event has occurred on a pin selected as a cycle-by-cycle trip source. The TZFLG[CBC] bit will remain set until it is manually cleared by the user. If the cycle-by-cycle trip event is still present when the CBC bit is cleared, then CBC will be immediately set again. The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCNT = 0000h) if the trip condition is no longer present. The condition on the pins is only cleared when the TBCNT = 0000h no matter where in the cycle the CBC flag is cleared. This bit is cleared by writing the appropriate value to the TZCLR register ( <a href="#">Section 14.4.6.5</a> ).
0	INT	0 1	Latched Trip Interrupt Status Flag 0 Indicates no interrupt has been generated. 1 Indicates an EPWMxTZINT interrupt was generated because of a trip condition. No further EPWMxTZINT interrupts will be generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts. This bit is cleared by writing the appropriate value to the TZCLR register ( <a href="#">Section 14.4.6.5</a> ).

#### 14.4.6.5 Trip-Zone Clear Register (TZCLR)

The trip-zone clear register (TZCLR) is shown in [Figure 14-85](#) and described in [Table 14-76](#).

**Figure 14-85. Trip-Zone Clear Register (TZCLR)**

15	Reserved	3	2	1	0
			OST	CBC	INT
	R-0		R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-76. Trip-Zone Clear Register (TZCLR) Field Descriptions**

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
2	OST	0	Clear Flag for One-Shot Trip (OST) Latch Has no effect. Always reads back a 0.
		1	Clears this Trip (set) condition.
1	CBC	0	Clear Flag for Cycle-By-Cycle (CBC) Trip Latch Has no effect. Always reads back a 0.
		1	Clears this Trip (set) condition.
0	INT	0	Global Interrupt Clear Flag Has no effect. Always reads back a 0.
		1	Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]). <b>NOTE:</b> No further EPWMxTZINT interrupts will be generated until the flag is cleared. If the TZFLG[INT] bit is cleared and any of the other flag bits are set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.

#### 14.4.6.6 Trip-Zone Force Register (TZFRC)

The trip-zone force register (TZFRC) is shown in [Figure 14-86](#) and described in [Table 14-77](#).

**Figure 14-86. Trip-Zone Force Register (TZFRC)**

15	Reserved	3	2	1	0
			OST	CBC	Rsvd
	R-0		R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-77. Trip-Zone Force Register (TZFRC) Field Descriptions**

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
2	OST	0	Force a One-Shot Trip Event via Software Writing of 0 is ignored. Always reads back a 0.
		1	Forces a one-shot trip event and sets the TZFLG[OST] bit.
1	CBC	0	Force a Cycle-by-Cycle Trip Event via Software Writing of 0 is ignored. Always reads back a 0.
		1	Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit.
0	Reserved	0	Reserved

### 14.4.7 Event-Trigger Submodule Registers

Table 14-78 lists the memory-mapped registers for the event-trigger submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 14-78 should be considered as reserved locations and the register contents should not be modified.

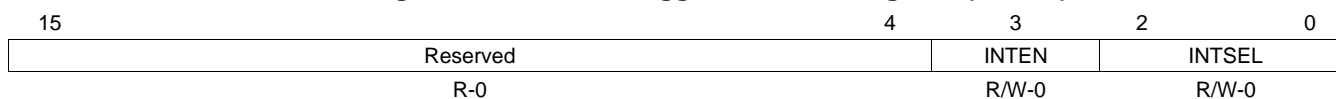
**Table 14-78. Event-Trigger Submodule Registers**

Offset	Acronym	Register Description	Section
32h	ETSEL	Event-Trigger Selection Register	<a href="#">Section 14.4.7.1</a>
34h	ETPS	Event-Trigger Prescale Register	<a href="#">Section 14.4.7.2</a>
36h	ETFLG	Event-Trigger Flag Register	<a href="#">Section 14.4.7.3</a>
38h	ETCLR	Event-Trigger Clear Register	<a href="#">Section 14.4.7.4</a>
3Ah	ETFRC	Event-Trigger Force Register	<a href="#">Section 14.4.7.5</a>

#### 14.4.7.1 Event-Trigger Selection Register (ETSEL)

The event-trigger selection register (ETSEL) is shown in [Figure 14-87](#) and described in [Table 14-79](#).

**Figure 14-87. Event-Trigger Selection Register (ETSEL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-79. Event-Trigger Selection Register (ETSEL) Field Descriptions**

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3	INTEN	0	Enable ePWM Interrupt (EPWMx_INT) Generation
		0	Disable EPWMx_INT generation
		1	Enable EPWMx_INT generation
2-0	INTSEL	0-7h	ePWM Interrupt (EPWMx_INT) Selection Options
		0	Reserved
		1h	Enable event time-base counter equal to zero. (TBCNT = 0000h)
		2h	Enable event time-base counter equal to period (TBCNT = TBPRD)
		3h	Reserved
		4h	Enable event time-base counter equal to CMPA when the timer is incrementing.
		5h	Enable event time-base counter equal to CMPA when the timer is decrementing.
		6h	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		7h	Enable event: time-base counter equal to CMPB when the timer is decrementing.

### 14.4.7.2 Event-Trigger Prescale Register (ETPS)

The event-trigger prescale register (ETPS) is shown in [Figure 14-88](#) and described in [Table 14-80](#).

**Figure 14-88. Event-Trigger Prescale Register (ETPS)**

15	4	3	2	1	0
Reserved			INTCNT	INTPRD	
R-0			R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-80. Event-Trigger Prescale Register (ETPS) Field Descriptions**

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3-2	INTCNT	0-3h	ePWM Interrupt Event (EPWMx_INT) Counter Register. These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
		0	No events have occurred.
		1h	1 event has occurred.
		2h	2 events have occurred.
		3h	3 events have occurred.
1-0	INTPRD	0-3h	ePWM Interrupt (EPWMx_INT) Period Select. These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared.  Writing a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear.  Writing a INTPRD value that is less than the current counter value will result in an undefined state.  If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented.
		0	Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.
		1h	Generate an interrupt on the first event INTCNT = 01 (first event)
		2h	Generate interrupt on ETPS[INTCNT] = 1,0 (second event)
		3h	Generate interrupt on ETPS[INTCNT] = 1,1 (third event)

### 14.4.7.3 Event-Trigger Flag Register (ETFLG)

The event-trigger flag register (ETFLG) is shown in [Figure 14-89](#) and described in [Table 14-81](#).

**Figure 14-89. Event-Trigger Flag Register (ETFLG)**

15	Reserved	1	0
R-0		INT R-0	

LEGEND: R = Read only; -n = value after reset

**Table 14-81. Event-Trigger Flag Register (ETFLG) Field Descriptions**

Bits	Name	Value	Description
15-1	Reserved	0	Reserved
0	INT	0	Latched ePWM Interrupt (EPWMx_INT) Status Flag Indicates no event occurred
		1	Indicates that an ePWMx interrupt (EPWMx_INT) was generated. No further interrupts will be generated until the flag bit is cleared. Up to one interrupt can be pending while the ETFLG[INT] bit is still set. If an interrupt is pending, it will not be generated until after the ETFLG[INT] bit is cleared. Refer to <a href="#">Figure 14-42</a> .

### 14.4.7.4 Event-Trigger Clear Register (ETCLR)

The event-trigger clear register (ETCLR) is shown in [Figure 14-90](#) and described in [Table 14-82](#).

**Figure 14-90. Event-Trigger Clear Register (ETCLR)**

15	Reserved	1	0
R-0		INT R-0	

LEGEND: R = Read only; -n = value after reset

**Table 14-82. Event-Trigger Clear Register (ETCLR) Field Descriptions**

Bits	Name	Value	Description
15-1	Reserved	0	Reserved
0	INT	0	ePWM Interrupt (EPWMx_INT) Flag Clear Bit Writing a 0 has no effect. Always reads back a 0.
		1	Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated.

### 14.4.7.5 Event-Trigger Force Register (ETFRC)

The event-trigger force register (ETFRC) is shown in [Figure 14-91](#) and described in [Table 14-83](#).

**Figure 14-91. Event-Trigger Force Register (ETFRC)**

15	Reserved	1	0
R-0		INT R-0	

LEGEND: R = Read only; -n = value after reset

**Table 14-83. Event-Trigger Force Register (ETFRC) Field Descriptions**

Bits	Name	Value	Description
15-1	Reserved	0	Reserved
0	INT	0	INT Force Bit. The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless.
		0	Writing 0 to this bit will be ignored. Always reads back a 0.
		1	Generates an interrupt on $\overline{\text{EPWMxINT}}$ and set the INT flag bit. This bit is used for test purposes.

### 14.4.8 High-Resolution PWM Submodule Registers

[Table 14-84](#) lists the memory-mapped registers for the high-resolution PWM submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in [Table 14-84](#) should be considered as reserved locations and the register contents should not be modified.

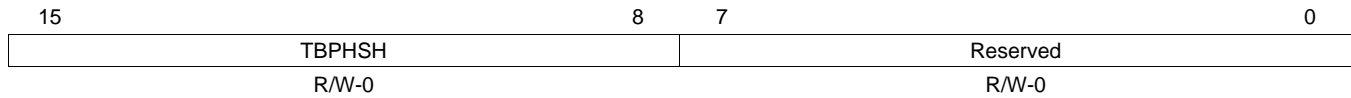
**Table 14-84. High-Resolution PWM Submodule Registers**

Offset	Acronym	Register Description	Section
4h	TBPHSHR	Time-Base Phase High-Resolution Register	<a href="#">Section 14.4.8.1</a>
10h	CMPAHR	Counter-Compare A High-Resolution Register	<a href="#">Section 14.4.8.2</a>
1040h	HRCNFG	HRPWM Configuration Register	<a href="#">Section 14.4.8.3</a>

#### 14.4.8.1 Time-Base Phase High-Resolution Register (TBPHSHR)

The time-base phase high-resolution register (TBPHSHR) is shown in [Figure 14-92](#) and described in [Table 14-85](#).

**Figure 14-92. Time-Base Phase High-Resolution Register (TBPHSHR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

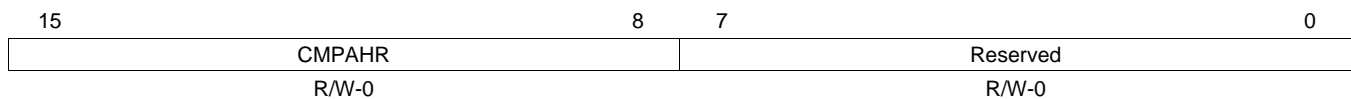
**Table 14-85. Time-Base Phase High-Resolution Register (TBPHSHR) Field Descriptions**

Bit	Field	Value	Description
15-8	TBPHSH	0-FFh	Time-base phase high-resolution bits
7-0	Reserved	0	Reserved

#### 14.4.8.2 Counter-Compare A High-Resolution Register (CMPAHR)

The counter-compare A high-resolution register (CMPAHR) is shown in [Figure 14-93](#) and described in [Table 14-86](#).

**Figure 14-93. Counter-Compare A High-Resolution Register (CMPAHR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-86. Counter-Compare A High-Resolution Register (CMPAHR) Field Descriptions**

Bit	Field	Value	Description
15-8	CMPAHR	1-FFh	Compare A High-Resolution register bits for MEP step control. A minimum value of 1h is needed to enable HRPWM capabilities. Valid MEP range of operation 1-255h.
7-0	Reserved	0	Reserved

### 14.4.8.3 HRPWM Configuration Register (HRCNFG)

The HRPWM configuration register (HRCNFG) is shown in [Figure 14-94](#) and described in [Table 14-87](#).

**Figure 14-94. HRPWM Configuration Register (HRCNFG)**

15	4	3	2	1	0
Reserved		HRLOAD	CTLMODE	EDGMODE	
R-0		R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-87. HRPWM Configuration Register (HRCNFG) Field Descriptions**

Bit	Field	Value	Description
15-4	Reserved	0	Reserved
3	HRLOAD	0 1	Shadow mode bit: Selects the time event that loads the CMPAHR shadow value into the active register: CTR = 0 (counter equals zero) CTR = PRD (counter equal period) Note: Load mode selection is valid only if CTLMODE = 0 has been selected. You should select this event to match the selection of the CMPA load mode (CMPCTL[LOADMODE] bits) in the EPWM module as follows: 0 Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h) 1h Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD) 2h Load on either CTR = 0 or CTR = PRD (should not be used with HRPWM) 3h Freeze (no loads possible – should not be used with HRPWM)
2	CTLMODE	0 1	Control Mode Bits: Selects the register (CMP or TBPHS) that controls the MEP: 0 CMPAHR(8) Register controls the edge position (this is duty control mode). (default on reset) 1 TBPHSHR(8) Register controls the edge position (this is phase control mode).
1-0	EDGMODE	0-3h 0 1h 2h 3h	Edge Mode Bits: Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic: 0 HRPWM capability is disabled (default on reset) 1h MEP control of rising edge 2h MEP control of falling edge 3h MEP control of both edges



## ***Enhanced Direct Memory Access (EDMA3) Controller***

---



---

The enhanced direct memory access (EDMA3) controller is a high-performance, multichannel, multithreaded DMA controller that allows you to program a wide variety of transfer geometries and transfer sequences. This chapter describes the features and operations of the EDMA3 controller.

[Section 15.1](#) provides a brief overview, features, and terminology. [Section 15.2](#) provides the architecture details and common operations of the EDMA3 channel controllers (EDMA3\_m\_CC0) and the EDMA3 transfer controllers (EDMA3\_m\_TCn). [Section 15.3](#) contains examples and common usage scenarios. [Section 15.4](#) describes the memory-mapped registers associated with the EDMA3 controller.

Topic	Page
<b>15.1 Introduction .....</b>	<b><a href="#">447</a></b>
<b>15.2 Architecture .....</b>	<b><a href="#">452</a></b>
<b>15.3 Transfer Examples .....</b>	<b><a href="#">495</a></b>
<b>15.4 Registers .....</b>	<b><a href="#">512</a></b>
<b>15.5 Tips .....</b>	<b><a href="#">579</a></b>
<b>15.6 Setting Up a Transfer .....</b>	<b><a href="#">581</a></b>

## 15.1 Introduction

### 15.1.1 Overview

The enhanced direct memory access (EDMA3) controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the device. Typical usage includes, but is not limited to:

- Servicing software driven paging transfers (for example, from external memory to internal device memory)
- Servicing event driven peripherals, such as a serial port
- Performing sorting or subframe extraction of various data structures
- Offloading data transfers from the main device CPU(s) or DSP(s) (See your device-specific data manual for specific peripherals that are accessible via EDMA3. See the section on SCR connectivity in your device-specific data manual for EDMA3 connectivity.)

The EDMA3 controller consists of two principal blocks:

- EDMA3 channel controller: (EDMA3\_m\_CC0)
- EDMA3 transfer controller: (EDMA3\_m\_TCn)

The EDMA3 channel controller serves as the user interface for the EDMA3 controller. The EDMA3CC includes parameter RAM (PaRAM), channel control registers, and interrupt control registers. The EDMA3CC serves to prioritize incoming software requests or events from peripherals, and submits transfer requests (TR) to the EDMA3 transfer controller.

The EDMA3 transfer controllers are responsible for data movement. The transfer request packets (TRP) submitted by the EDMA3CC contains the transfer context, based on which the transfer controller issues read/write commands to the source and destination addresses programmed for a given transfer.

### 15.1.2 Features

The EDMA3 channel controller (EDMA3CC) has the following features:

- Fully orthogonal transfer description
  - 3 transfer dimensions
  - A-synchronized transfers: 1 dimension serviced per event
  - AB-synchronized transfers: 2 dimensions serviced per event
  - Independent indexes on source and destination
  - Chaining feature allows 3-D transfer based on single event
- Flexible transfer definition
  - Increment or constant addressing modes
  - Linking mechanism allows automatic PaRAM set update. Useful for ping-pong type transfers, auto-reload transfers.
  - Chaining allows multiple transfers to execute with a single event
- Interrupt generation for:
  - Transfer completion
  - Error conditions (illegal addresses, illegal modes, exceeding queue threshold)
- Debug visibility
  - Queue watermarking
  - Error and status recording to facilitate debug
  - Missed event detection

- EDMA3\_0\_CC0:
  - 32 DMA channels
  - 8 QDMA channels
  - 128 parameter RAM (PaRAM) entries
  - 2 event queues
  - 4 shadow regions
  - 2 transfer controllers (EDMA3\_0\_TC0 and EDMA3\_0\_TC1)
  - 5 interrupts:
    - EDMA3\_0\_CC0\_INT0
    - EDMA3\_0\_CC0\_INT1
    - EDMA3\_0\_CC0\_INT2
    - EDMA3\_0\_CC0\_INT3
    - EDMA3\_0\_CC0\_ERRINT
- EDMA3\_1\_CC0:
  - 32 DMA channels
  - 8 QDMA channels
  - 128 parameter RAM (PaRAM) entries
  - 1 event queue
  - 4 shadow regions
  - 1 transfer controller (EDMA3\_1\_TC0)
  - 5 interrupts:
    - EDMA3\_1\_CC0\_INT0
    - EDMA3\_1\_CC0\_INT1
    - EDMA3\_1\_CC0\_INT2
    - EDMA3\_1\_CC0\_INT3
    - EDMA3\_1\_CC0\_ERRINT

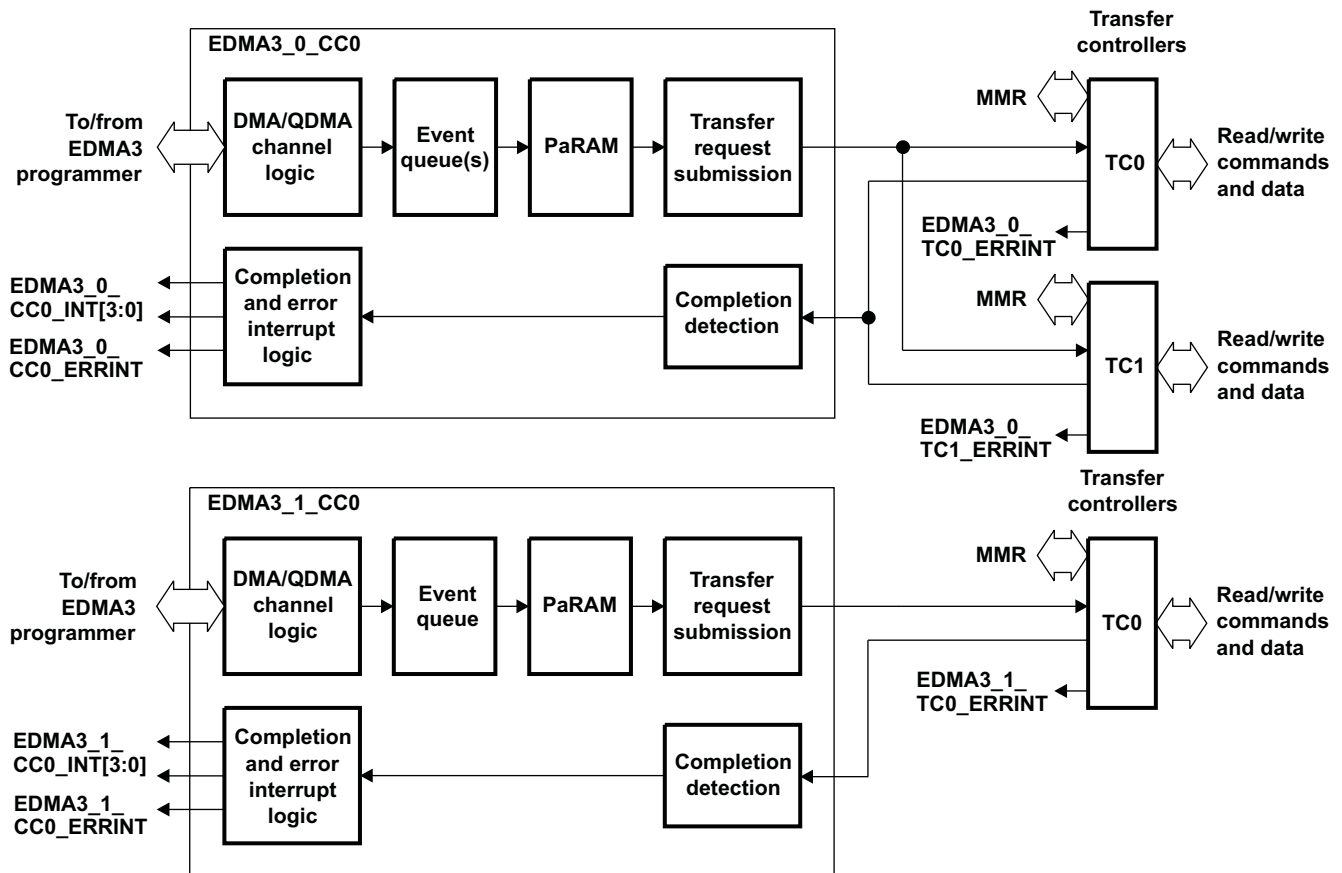
The EDMA3 transfer controller (EDMA3TC) has the following features:

- Supports 2-dimensional transfers with independent indexes on source and destination (EDMA3CC manages the 3rd dimension)
- More than one transfer controller allows concurrent transfers
- Programmable priority level for each transfer controller relative to each other and other masters in the system.
- Support for increment or constant addressing mode transfers
- Error conditions with interrupt support
- Supports more than one in-flight transfer requests
- Debug/status visibility
- 64-bit wide read and write ports
- Little-endian mode
- EDMA3\_0\_TC0:
  - FIFIOSIZE = 128 bytes
  - BUSWIDTH (Read/Write Controllers) = 8 byte (64 bits)
  - DSTREGDEPTH = 4
  - DBS (default) = 16 bytes. The default burst size (DBS) is programmable, and can be configured for 16-, 32-, or 64-bytes burst size. See the Chip Configuration 0 Register (CFGCHIP0) in the *System Configuration (SYSCFG) Module* chapter for details to change the default burst size value.
  - Error interrupt: EDMA3\_0\_TC0\_ERRINT
  - EDMA3 channel controller used: EDMA3\_0\_CC0
- EDMA3\_0\_TC1:
  - FIFIOSIZE = 128 bytes
  - BUSWIDTH (Read/Write Controllers) = 8 byte (64 bits)
  - DSTREGDEPTH = 4
  - DBS (default) = 16 bytes. The default burst size (DBS) is programmable, and can be configured for 16-, 32-, or 64-bytes burst size. See the Chip Configuration 0 Register (CFGCHIP0) in the *System Configuration (SYSCFG) Module* chapter for details to change the default burst size value.
  - Error interrupt: EDMA3\_0\_TC1\_ERRINT
  - EDMA3 channel controller used: EDMA3\_0\_CC0
- EDMA3\_1\_TC0:
  - FIFIOSIZE = 256 bytes
  - BUSWIDTH (Read/Write Controllers) = 8 byte (64 bits)
  - DSTREGDEPTH = 4
  - DBS (default) = 16 bytes. The default burst size (DBS) is programmable, and can be configured for 16-, 32-, or 64-bytes burst size. See the Chip Configuration 1 Register (CFGCHIP1) in the *System Configuration (SYSCFG) Module* chapter for details to change the default burst size value.
  - Error interrupt: EDMA3\_1\_TC0\_ERRINT
  - EDMA3 channel controller used: EDMA3\_1\_CC0

### 15.1.3 Functional Block Diagram

Figure 15-1 shows a block diagram of the EDMA3 controller.

Figure 15-1. EDMA3 Controller Block Diagram



#### 15.1.4 Terminology Used in This Document

The following are some terms used in this chapter.

Term	Meaning
A-synchronized transfer	A transfer type where 1 dimension is serviced per synchronization event.
AB-synchronized transfer	A transfer type where 2 dimensions are serviced per synchronization event.
Chaining	A trigger mechanism in which a transfer can be initiated at the completion of another transfer or subtransfer.
CPU(s)	The main processing engine or engines on a device. Typically a DSP or general-purpose processor. (See your device-specific data manual to learn more about the CPU on your system.)
DMA channel	A channel that can be triggered by external, manual, and chained events. All DMA channels exist in the EDMA3CC.
Dummy set or Dummy PaRAM set	A PaRAM set for which at least one of the count fields is equal to 0 and at least one of the count fields is nonzero. A null PaRAM set has all the count set fields cleared.

Term	Meaning
Dummy transfer	A dummy set results in the EDMA3CC performing a dummy transfer. This is not an error condition. A null set results in an error condition.
EDMA3 channel controller (EDMA3CC)	The user-programmable portion of the EDMA3. The EDMA3CC contains the parameter RAM (PaRAM) , event processing logic, DMA/QDMA channels, event queues, etc. The EDMA3CC services events (external, manual, chained, QDMA) and is responsible for submitting transfer requests to the transfer controllers (EDMA3TC), which perform the actual transfer.
EDMA3 programmer	Any entity on the chip that has read/write access to the EDMA3 registers and can program an EDMA3 transfer.
EDMA3 transfer controller(s) (EDMA3TC)	Transfer controllers are the transfer engine for the EDMA3. Performs the read/writes as dictated by the transfer requests submitted by the EDMA3CC.
Enhanced direct memory access (EDMA3) controller	Consists of the EDMA3 channel controller (EDMA3CC) and EDMA3 transfer controller(s) (EDMA3TC). Is referred to as EDMA3 in this document.
Link parameter set	A PaRAM set that is used for linking.
Linking	The mechanism of reloading a PaRAM set with new transfer characteristics on completion of the current transfer.
Memory-mapped slave	All on-chip memories, off-chip memories, and slave peripherals. These typically rely on the EDMA3 (or other master peripheral) to perform transfers to and from them.
Master peripherals	All peripherals that are capable of initiating read and write transfers to the peripherals system and may not solely rely on the EDMA3 for their data transfers.
Null set or Null PaRAM set	A PaRAM set that has all count fields cleared (except for the link field). A dummy PaRAM set has at least one of the count fields nonzero.
Null transfer	A trigger event for a null PaRAM set results in the EDMA3CC performing a null transfer. This is an error condition. A dummy transfer is not an error condition.
QDMA channel	One of the 8 channels that can be triggered when writing to the trigger word (TRWORD) of a PaRAM set. All QDMA channels exist in the EDMA3CC.
Parameter RAM (PaRAM)	Programmable RAM that stores PaRAM sets used by DMA channels, QDMA channels, and linking.
Parameter RAM (PaRAM) set	A 32-byte EDMA3 channel transfer definition. Each parameter set consists of 8 words (4-bytes each), which store the context for a DMA/QDMA/link transfer. A PaRAM set includes source address, destination address, counts, indexes, options, etc.
Parameter RAM (PaRAM) set entry	One of the 4-byte components of the parameter set.
Slave end points	All on-chip memories, off-chip memories, and slave peripherals. These rely on the EDMA3 to perform transfers to and from them.
Transfer request (TR)	A command for data movement that is issued from the EDMA3CC to the EDMA3TC. A TR includes source and destination addresses, counts, indexes, options, etc.
Trigger event	Action that causes the EDMA3CC to service the PaRAM set and submit a transfer request to the EDMA3TC. Trigger events for DMA channels include manual triggered (CPU triggered), external event triggered, and chain triggered. Trigger events for QDMA channels include autotriggered and link triggered.
Trigger word	For QDMA channels, the trigger word specifies the PaRAM set entry that when written results in a QDMA trigger event. The trigger word is programmed via the QDMA channel <i>n</i> mapping register (QCHMAP <i>n</i> ) and can point to any PaRAM set entry.
TR synchronization (sync) event	See Trigger event.

## 15.2 Architecture

This section discusses the architecture of the EDMA3 controller.

### 15.2.1 Functional Overview

This section provides an overview of the EDMA3 channel controller (EDMA3CC) and EDMA3 transfer controller (EDMA3TC).

#### 15.2.1.1 EDMA3 Channel Controller (EDMA3CC)

[Figure 15-2](#) shows a functional block diagram of the EDMA3 channel controller (EDMA3CC).

The main blocks of the EDMA3CC are:

- **DMA/QDMA Channel Logic:** This block consists of logic that captures external system or peripheral events that can be used to initiate event triggered transfers, it also includes registers that allow configuring the DMA/QDMA channels (queue mapping, PaRAM entry mapping). It includes all the registers for different trigger type (manual, external events, chained and auto triggered) for enabling/disabling events, and monitor event status.
- **Parameter RAM (PaRAM):** Maintains parameter set entries for channel and reload parameter sets. The PaRAM needs to be written with the transfer context for the desired channels and link parameter sets.
- **Event queues:** These form the interface between the event detection logic and the transfer request submission logic.
- **Transfer Request Submission Logic:** This logic processes PaRAM sets based on a trigger event submitted to the event queue and submits a transfer request (TR) to the transfer controller associated with the event queue.
- **Completion detection:** The completion detect block detects completion of transfers by the EDMA3 transfer controller (EDMA3TC) and/or slave peripherals. Completion of transfers can optionally be used to chain trigger new transfers or to assert interrupts. The logic includes the interrupt processing registers for enabling/disabling interrupt (to be sent to the CPU), interrupt status/clearing registers.

Additionally there are:

- **Region registers:** Region registers allow DMA resources (DMA channels and interrupts) to be assigned to unique regions, which can be owned by unique EDMA programmers (a use model for hetero/multi core devices) or by unique tasks/threads (a use model for single core devices).
- **Debug registers:** Debug registers allow debug visibility by providing registers to read the queue status, channel controller status (what logic within the CC is active), and missed event status.

The EDMA3CC includes two channel types: DMA channels and QDMA channels.

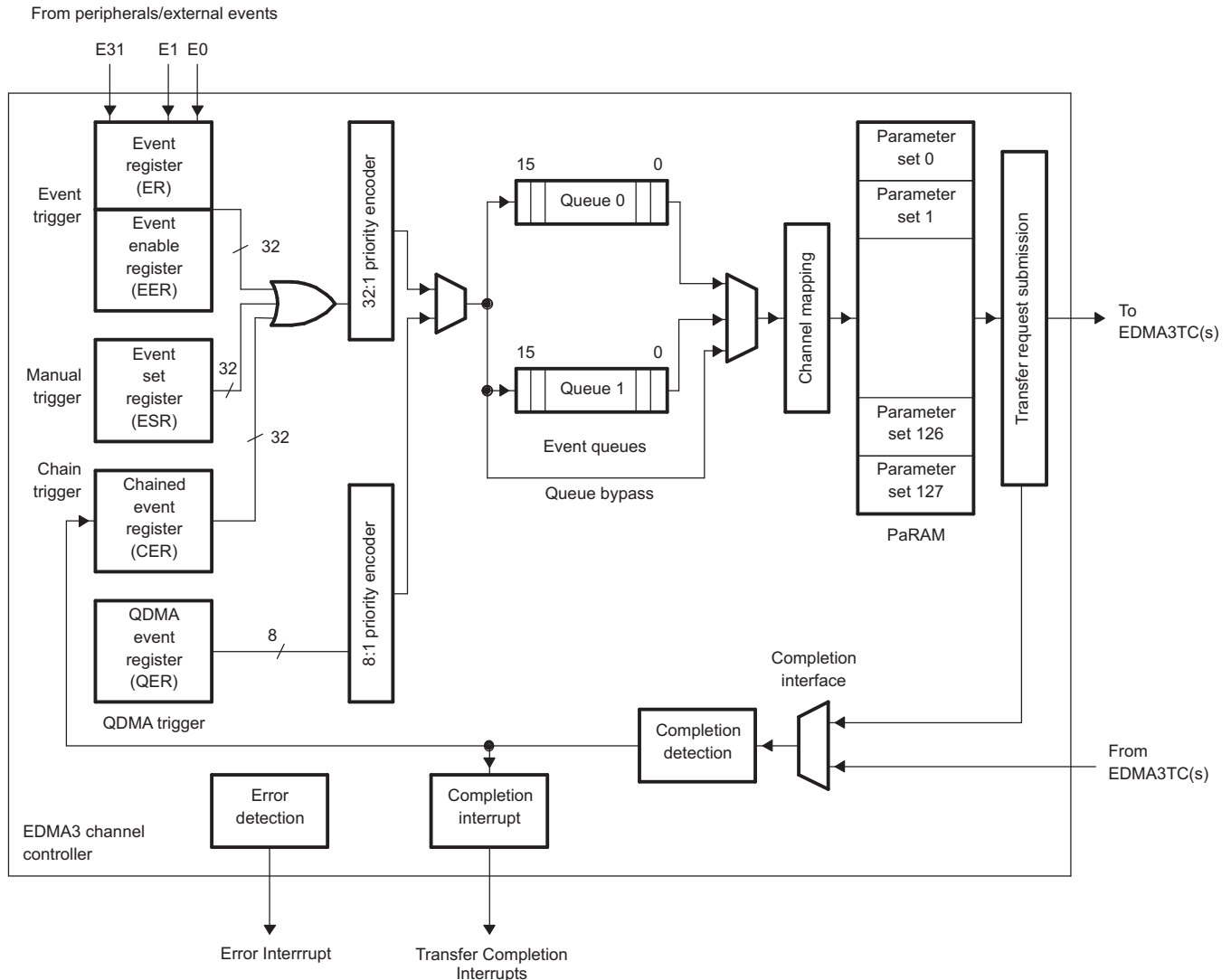
Each channel is associated with a given event queue/transfer controller and with a given PaRAM set. The main difference between a DMA channel and QDMA channel is how the transfers are triggered by the system. See [Section 15.2.4](#).

A trigger event is needed to initiate a transfer. For DMA channels, a trigger event may be due to an external event, manual write to the event set register, or chained event. QDMA channels are autotriggered when a write is performed to the user-programmed trigger word. All such trigger events are logged into appropriate registers upon recognition. See DMA channel registers ([Section 15.4.2.5](#)) and QDMA channel registers ([Section 15.4.2.7](#)).

Once a trigger event is recognized, the event type/channel is queued in the appropriate EDMA3CC event queue. The assignment of each DMA/QDMA channel to event queue is programmable. Each queue is 16 deep, so up to 16 events may be queued (on a single queue) in the EDMA3CC at an instant in time. Additional pending events mapped to a full queue are queued when event queue space becomes available. See [Section 15.2.10](#).

If events on different channels are detected simultaneously, the events are queued based on fixed priority arbitration scheme with the DMA channels being higher priority than the QDMA channels. Among the two groups of channels, the lowest-numbered channel is the highest priority.

Figure 15-2. EDMA3 Channel Controller (EDMA3CC) Block Diagram



Each event in the event queue is processed in the order it was queued. On reaching the head of the queue, the PaRAM associated with that channel is read to determine the transfer details. The TR submission logic evaluates the validity of the TR and is responsible for submitting a valid transfer request (TR) to the appropriate EDMA3TC (based on the event queue to EDMA3TC association, Q0 goes to TC0, and Q1 goes to TC1, etc.). For more details, see [Section 15.2.3](#).

The EDMA3TC receives the request and is responsible for data movement as specified in the transfer request packet (TRP) and other necessary tasks like buffering, ensuring transfers are carried out in an optimal fashion wherever possible. For more details on EDMA3TC, see [Section 15.2.1.2](#).

You may have chosen to receive an interrupt or chain to another channel on completion of the current transfer in which case the EDMA3TC signals completion to the EDMA3CC completion detection logic when the transfer is done. You can alternately choose to trigger completion when a TR leaves the EDMA3CC boundary rather than wait for all the data transfers to complete. Based on the setting of the EDMA3CC interrupt registers, the completion interrupt generation logic is responsible for generating EDMA3CC completion interrupts to the CPU. For more details, see [Section 15.2.5](#).

Additionally, the EDMA3CC also has an error detection logic, which causes error interrupt generation on various error conditions (like missed events, exceeding event queue thresholds, etc.). For more details on error interrupts, see [Section 15.2.9.4](#).



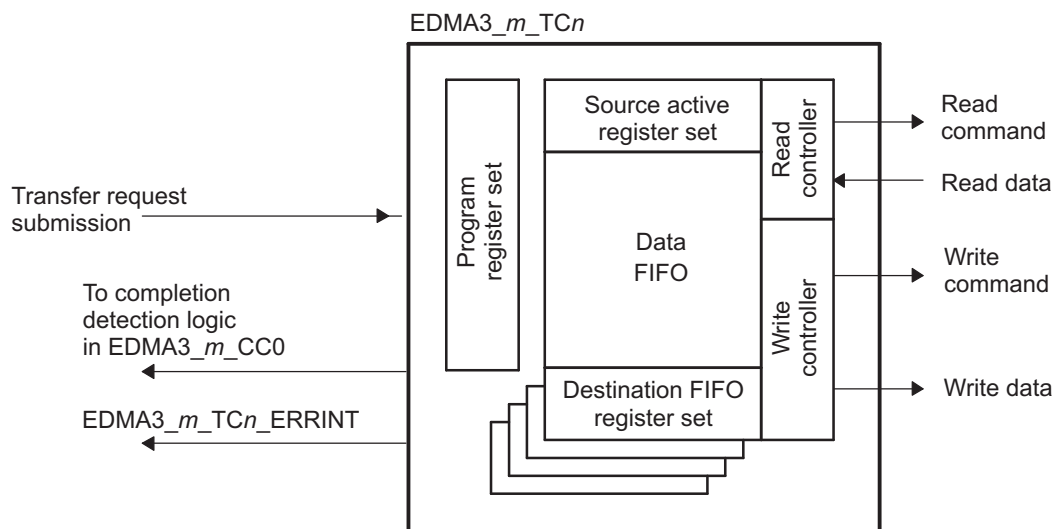
### 15.2.1.2 EDMA3 Transfer Controller (EDMA3TC)

Figure 15-3 shows a functional block diagram of the EDMA3 transfer controller (EDMA3TC).

The main blocks of the EDMA3TC are:

- DMA program register set: The DMA program register set stores the transfer requests received from the EDMA3 channel controller (EDMA3CC).
- DMA source active register set: The DMA source active register set stores the context for the DMA transfer request currently in progress in the read controller.
- Read controller: The read controller issues read commands to the source address.
- Destination FIFO register set: The destination (Dst) FIFO register set stores the context for the DMA transfer request(s) currently in progress or pending in the write controller.
- Write controller: The write controller issues write commands/write data to the destination address.
- Data FIFO: The data FIFO holds temporary in-flight data. The source peripheral's read data is stored in the data FIFO and subsequently written to the destination peripheral/end point by the write controller.
- Completion interface: The completion interface sends completion codes to the EDMA3CC when a transfer completes, and is used for generating interrupts and chained events (see Section 15.2.5 for details on transfer completion reporting).

**Figure 15-3. EDMA3 Transfer Controller (EDMA3TC) Block Diagram**



When the EDMA3TC is idle and receives its first TR, the TR is received in the DMA program register set, where it transitions to the DMA source active set and the destination FIFO register set immediately. The source active register set tracks the commands for the source side of the transfers, and the destination FIFO register set tracks commands for the destination side of the transfer. The second TR (if pending from EDMA3CC) is loaded into the DMA program set, ensuring it can start as soon as possible when the active transfer (the transfer in the source active set) is completed. As soon as the current active set is exhausted, the TR is loaded from the DMA program register set into the DMA source active register set as well as to the appropriate entry in the destination FIFO register set.

The read controller issues read commands governed by the rules of command fragmentation and optimization. These are issued only when the data FIFO has space available for the read data. The number of read commands issued depends on the TR transfer size. The TC write controller starts issuing write commands as soon as sufficient data is read in the data FIFO for the write controller to issue optimally sized write commands following the rules for command fragmentation and optimization. For details on command fragmentation and optimization, see Section 15.2.11.1.2.

The DSTREGDEPTH parameter (fixed for a given transfer controller) determines the number of entries in the Dst FIFO register set. The number of entries determines the amount of TR pipelining possible for a given TC. The write controller can manage the write context for the number of entries in the Dst FIFO register set. This allows the read controller to go ahead and issue read commands for the subsequent TRs while the Dst FIFO register set manages the write commands and data for the previous TR. In summary, if the DSTREGDEPTH is  $n$ , the read controller is able to process up to  $n$ TRs ahead of the write controller. However, the overall TR pipelining is also subject to the amount of free space in the data FIFO.

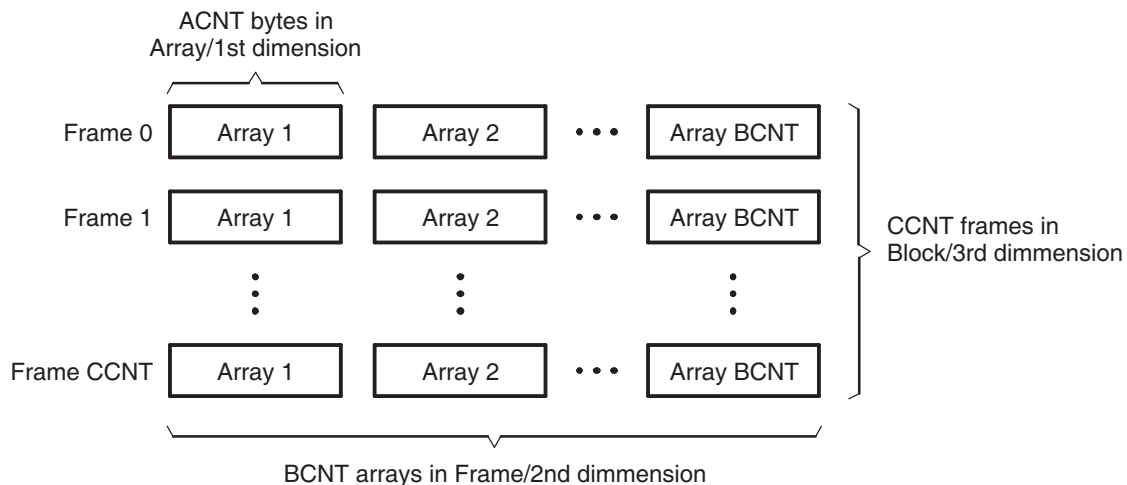
### 15.2.2 Types of EDMA3 Transfers

An EDMA3 transfer is always defined in terms of three dimensions. Figure 15-4 shows the three dimensions used by EDMA3 transfers. These three dimensions are defined as:

- 1st Dimension or Array (A): The 1st dimension in a transfer consists of ACNT contiguous bytes.
- 2nd Dimension or Frame (B): The 2nd dimension in a transfer consists of BCNT arrays of ACNT bytes. Each array transfer in the 2nd dimension is separated from each other by an index programmed using SRCBIDX or DSTBIDX.
- 3rd Dimension or Block (C): The 3rd dimension in a transfer consists of CCNT frames of BCNT arrays of ACNT bytes. Each transfer in the 3rd dimension is separated from the previous by an index programmed using SRCCIDX or DSTCIDX.

Note that the reference point for the index depends on the synchronization type. The amount of data transferred upon receipt of a trigger/synchronization event is controlled by the synchronization types (SYNCDIM bit in OPT). Of the three dimensions, only two synchronization types are supported: A-synchronized transfers and AB-synchronized transfers.

**Figure 15-4. Definition of ACNT, BCNT, and CCNT**



### 15.2.2.1 A-Synchronized Transfers

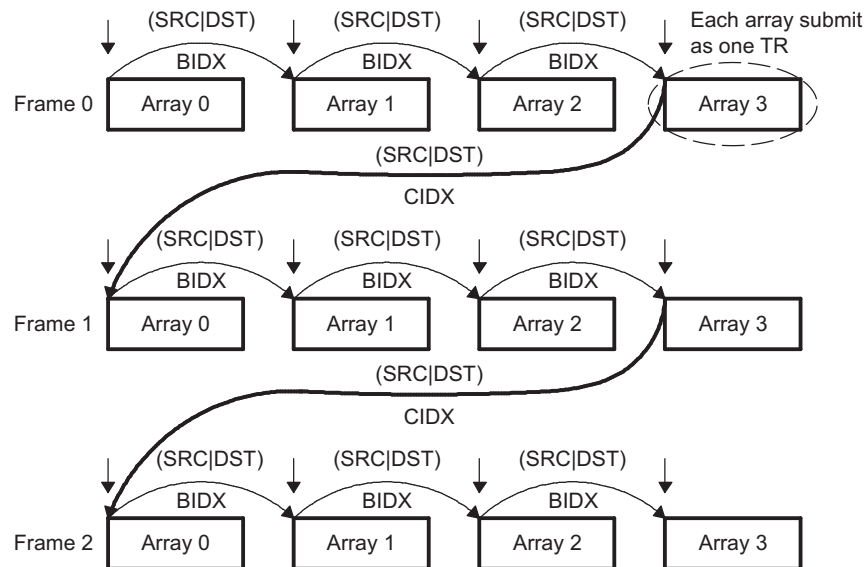
In an A-synchronized transfer, each EDMA3 sync event initiates the transfer of the 1st dimension of ACNT bytes, or one array of ACNT bytes. In other words, each event/TR packet conveys the transfer information for one array only. Thus, BCNT × CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX, as shown in Figure 15-5, where the start address of Array N is equal to the start address of Array N – 1 plus source (SRCBIDX) or destination (DSTBIDX).

Frames are always separated by SRCCIDX and DSTCIDX. For A-synchronized transfers, after the frame is exhausted, the address is updated by adding SRCCIDX/DSTCIDX to the beginning address of the last array in the frame. As in Figure 15-5, SRCCIDX/DSTCIDX is the difference between the start of Frame 0 Array 3 to the start of Frame 1 Array 0.

Figure 15-5 shows an A-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of n (ACNT) bytes. In this example, a total of 12 sync events (BCNT × CCNT) exhaust a PaRAM set. See Section 15.2.3.6 for details on parameter set updates.

**Figure 15-5. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)**



### 15.2.2.2 AB-Synchronized Transfers

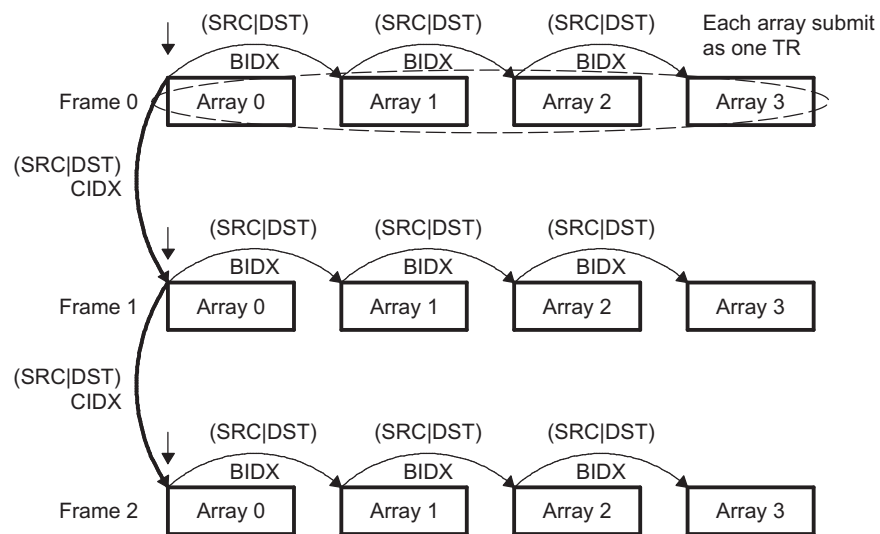
In a AB-synchronized transfer, each EDMA3 sync event initiates the transfer of 2 dimensions or one frame. In other words, each event/TR packet conveys information for one entire frame of BCNT arrays of ACNT bytes. Thus, CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX as shown in Figure 15-6. Frames are always separated by SRCCIDX and DSTCIDX.

Note that for AB-synchronized transfers, after a TR for the frame is submitted, the address update is to add SRCCIDX/DSTCIDX to the beginning address of the beginning array in the frame. This is different from A-synchronized transfers where the address is updated by adding SRCCIDX/DSTCIDX to the start address of the last array in the frame. See Section 15.2.3.6 for details on parameter set updates.

Figure 15-6 shows an AB-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of  $n$  (ACNT) bytes. In this example, a total of 3 sync events (CCNT) exhaust a PaRAM set; that is, a total of 3 transfers of 4 arrays each completes the transfer.

**Figure 15-6. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)**



**NOTE:** ABC-synchronized transfers are not directly supported. But can be logically achieved by chaining between multiple AB-synchronized transfers.

### 15.2.3 Parameter RAM (PaRAM)

The EDMA3 controller is a RAM-based architecture. The transfer context (source/destination addresses, count, indexes, etc.) for DMA or QDMA channels is programmed in a parameter RAM table within the EDMA3CC, referred to as PaRAM. The PaRAM table is segmented into multiple PaRAM sets. Each PaRAM set includes eight 4-byte PaRAM set entries (32-bytes total per PaRAM set), which includes typical DMA transfer parameters such as source address, destination address, transfer counts, indexes, options, etc. See your device-specific data manual for the addresses of the PaRAM set entries.

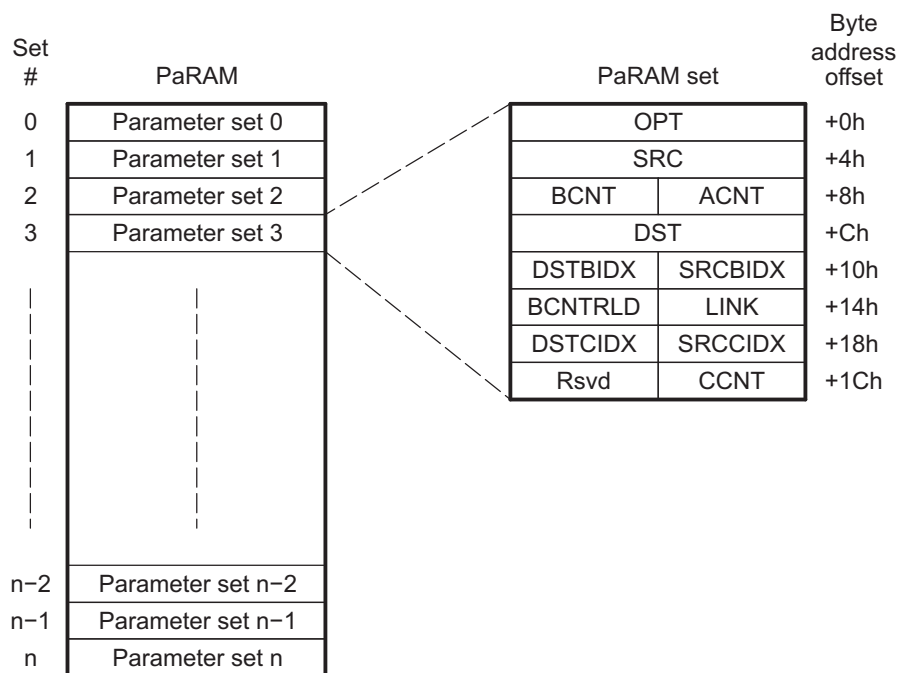
The PaRAM structure supports flexible ping-pong, circular buffering, channel chaining, and autoreloading (linking). The first  $n$  PaRAM sets are directly mapped to the DMA channels (where  $n$  is the number of DMA channels supported in the EDMA3CC for a specific device). The remaining PaRAM sets can be used for link entries or associated with QDMA channels. Additionally if the DMA channels are not used, the PaRAM sets associated with the unused DMA channels can also be used for link entries or QDMA channels.

**NOTE:** By default, QDMA channels are mapped to PaRAM set 0. These should be remapped before use, see [Section 15.2.6.2](#).

#### 15.2.3.1 PaRAM Set

Each parameter set of PaRAM is organized into eight 32-bit words or 32 bytes, as shown in [Figure 15-7](#) and described in [Table 15-1](#). Each PaRAM set consists of 16-bit and 32-bit parameters.

**Figure 15-7. PaRAM Set**



Note:  $n$  is the number of PaRAM sets supported in the EDMA3CC for a specific device.

**Table 15-1. EDMA3 Channel Parameter Description**

Offset Address (bytes)	Acronym	Parameter	Description
0h	OPT	Channel Options	Transfer Configuration Options
4h	SRC	Channel Source Address	The byte address from which data is transferred.
8h <sup>(1)</sup>	ACNT	Count for 1st Dimension	Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535.
	BCNT	Count for 2nd Dimension	Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535.
Ch	DST	Channel Destination Address	The byte address to which data is transferred.
10h <sup>(1)</sup>	SRCBIDX	Source BCNT Index	Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
	DSTBIDX	Destination BCNT Index	Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
14h <sup>(1)</sup>	LINK	Link Address	The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. A value of FFFFh specifies a null link.
	BCNTRLD	BCNT Reload	The count value used to reload BCNT when BCNT decrements to 0 (TR submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers.
18h <sup>(1)</sup>	SRCCIDX	Source CCNT Index	Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767.  A-synchronized transfers: The byte address offset from the beginning of the last source array in a frame to the beginning of the first source array in the next frame.  AB-synchronized transfers: The byte address offset from the beginning of the first source array in a frame to the beginning of the first source array in the next frame.
	DSTCIDX	Destination CCNT index	Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767.  A-synchronized transfers: The byte address offset from the beginning of the last destination array in a frame to the beginning of the first destination array in the next frame.  AB-synchronized transfers: The byte address offset from the beginning of the first destination array in a frame to the beginning of the first destination array in the next frame.
1Ch	CCNT	Count for 3rd Dimension	Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535.
	RSVD	Reserved	Reserved

<sup>(1)</sup> If OPT, SRC, or DST is the trigger word for a QDMA transfer then it is required to do a 32-bit access to that field. Furthermore, it is recommended to perform only 32-bit accesses on the parameter RAM for best code compatibility. For example, switching the endianness of the processor swaps addresses of the 16-bit fields, but 32-bit accesses avoid the issue entirely.

### 15.2.3.2 EDMA3 Channel Parameter Set Fields

#### 15.2.3.2.1 Channel Options Parameter (OPT)

The 32-bit channel options parameter (OPT) specifies the transfer configuration options. The channel options parameter (OPT) is described in [Section 15.4.1.1](#).

#### 15.2.3.2.2 Channel Source Address (SRC)

The 32-bit source address parameter specifies the starting byte address of the source. For SAM in increment mode, there are no alignment restrictions imposed by EDMA3. For SAM in constant addressing mode, you must program the source address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See [Section 15.2.11.2](#) for additional details.

#### 15.2.3.2.3 Channel Destination Address (DST)

The 32-bit destination address parameter specifies the starting byte address of the destination. For DAM in increment mode, there are no alignment restrictions imposed by EDMA3. For DAM in constant addressing mode, you must program the destination address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See [Section 15.2.11.2](#) for additional details.

#### 15.2.3.2.4 Count for 1st Dimension (ACNT)

ACNT represents the number of bytes within the 1st dimension of a transfer. ACNT is a 16-bit unsigned value with valid values between 0 and 65 535. Therefore, the maximum number of bytes in an array is 65 535 bytes (64K – 1 bytes). ACNT must be greater than or equal to 1 for a TR to be submitted to EDMA3TC. A transfer with ACNT equal to 0 is considered either a null or dummy transfer.

See [Section 15.2.3.5](#) and [Section 15.2.5.3](#) for details on dummy/null completion conditions.

#### 15.2.3.2.5 Count for 2nd Dimension (BCNT)

BCNT is a 16-bit unsigned value that specifies the number of arrays of length ACNT. For normal operation, valid values for BCNT are between 1 and 65 535. Therefore, the maximum number of arrays in a frame is 65 535 (64K – 1 arrays). A transfer with BCNT equal to 0 is considered either a null or dummy transfer.

See [Section 15.2.3.5](#) and [Section 15.2.5.3](#) for details on dummy/null completion conditions.

#### 15.2.3.2.6 Count for 3rd Dimension (CCNT)

CCNT is a 16-bit unsigned value that specifies the number of frames in a block. Valid values for CCNT are between 1 and 65 535. Therefore, the maximum number of frames in a block is 65 535 (64K – 1 frames). A transfer with CCNT equal to 0 is considered either a null or dummy transfer.

See [Section 15.2.3.5](#) and [Section 15.2.5.3](#) for details on dummy/null completion conditions.

#### 15.2.3.2.7 BCNT Reload (BCNTRLD)

BCNTRLD is a 16-bit unsigned value used to reload the BCNT field once the last array in the 2nd dimension is transferred. This field is only used for A-synchronized transfers. In this case, the EDMA3CC decrements the BCNT value by 1 on each TR submission. When BCNT reaches 0, the EDMA3CC decrements CCNT and uses the BCNTRLD value to reinitialize the BCNT value.

For AB-synchronized transfers, the EDMA3CC submits the BCNT in the TR and the EDMA3TC decrements BCNT appropriately. For AB-synchronized transfers, BCNTRLD is not used.



### 15.2.3.2.8 Source B Index (SRCBIDX)

SRCBIDX is a 16-bit signed value (2s complement) used for source address modification between each array in the 2nd dimension. Valid values for SRCBIDX are between  $-32\,768$  and  $32\,767$ . It provides a byte address offset from the beginning of the source array to the beginning of the next source array. It applies to both A-synchronized and AB-synchronized transfers. Some examples:

- SRCBIDX = 0000h (0): no address offset from the beginning of an array to the beginning of the next array. All arrays are fixed to the same beginning address.
- SRCBIDX = 0003h (+3): the address offset from the beginning of an array to the beginning of the next array in a frame is 3 bytes. For example, if the current array begins at address 1000h, the next array begins at 1003h.
- SRCBIDX = FFFFh (−1): the address offset from the beginning of an array to the beginning of the next array in a frame is −1 byte. For example, if the current array begins at address 5054h, the next array begins at 5053h.

### 15.2.3.2.9 Destination B Index (DSTBIDX)

DSTBIDX is a 16-bit signed value (2s complement) used for destination address modification between each array in the 2nd dimension. Valid values for DSTBIDX are between  $-32\,768$  and  $32\,767$ . It provides a byte address offset from the beginning of the destination array to the beginning of the next destination array within the current frame. It applies to both A-synchronized and AB-synchronized transfers. See SRCBIDX (Section 15.2.3.2.8) for examples.

### 15.2.3.2.10 Source C Index (SRCCIDX)

SRCCIDX is a 16-bit signed value (2s complement) used for source address modification in the 3rd dimension. Valid values for SRCCIDX are between  $-32\,768$  and  $32\,767$ . It provides a byte address offset from the beginning of the current array (pointed to by SRC address) to the beginning of the first source array in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when SRCCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 15-5), while the current array in an AB-synchronized transfer is the first array in the frame (Figure 15-6).

### 15.2.3.2.11 Destination C Index (DSTCIDX)

DSTCIDX is a 16-bit signed value (2s complement) used for destination address modification in the 3rd dimension. Valid values are between  $-32\,768$  and  $32\,767$ . It provides a byte address offset from the beginning of the current array (pointed to by DST address) to the beginning of the first destination array TR in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when DSTCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 15-5), while the current array in a AB-synchronized transfer is the first array in the frame (Figure 15-6).

### 15.2.3.2.12 Link Address (LINK)

The EDMA3CC provides a mechanism, called linking, to reload the current PaRAM set upon its natural termination (that is, after the count fields are decremented to 0) with a new PaRAM set. The 16-bit parameter LINK specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking.

You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0.

The EDMA3CC ignores the upper 2 bits of the LINK entry, allowing the programmer the flexibility of programming the link address as either an absolute/literal byte address or use the PaRAM-base-relative offset address. Therefore, if you make use of the literal address with a range from 4000h to 7FFFh, it will be treated as a PaRAM-base-relative value of 0000h to 3FFFh.

You should make sure to program the LINK field correctly, so that link update is requested from a PaRAM address that falls in the range of the available PaRAM addresses on the device.



A LINK value of FFFFh is referred to as a NULL link that should cause the EDMA3CC to perform an internal write of 0 to all entries of the current PaRAM set, except for the LINK field that is set to FFFFh. Also, see [Section 15.2.5](#) for details on terminating a transfer.

### 15.2.3.3 Null PaRAM Set

A null PaRAM set is defined as a PaRAM set where all count fields (ACNT, BCNT, and CCNT) are cleared to 0. If a PaRAM set associated with a channel is a NULL set, then when serviced by the EDMA3CC, the bit corresponding to the channel is set in the associated event missed register (EMR or QEMR). This bit remains set in the associated secondary event register (SER or QSER). *This implies that any future events on the same channel are ignored by the EDMA3CC and you are required to clear the bit in SER or QSER for the channel.* This is considered an error condition, since events are not expected on a channel that is configured as a null transfer. See [Section 15.4.2.5.8](#) and [Section 15.4.2.2.1](#) for more information on the SER and EMR registers, respectively.

### 15.2.3.4 Dummy PaRAM Set

A dummy PaRAM set is defined as a PaRAM set where at least one of the count fields (ACNT, BCNT, or CCNT) is cleared to 0 and at least one of the count fields is nonzero.

If a PaRAM set associated with a channel is a dummy set, then when serviced by the EDMA3CC, it will not set the bit corresponding to the channel (DMA/QDMA) in the event missed register (EMR or QEMR) and the secondary event register (SER or QSER) bit gets cleared similar to a normal transfer. Future events on that channel are serviced. A dummy transfer is a legal transfer of 0 bytes. See [Section 15.4.2.5.8](#) and [Section 15.4.2.2.1](#) for more information on the SER and EMR registers, respectively.

### 15.2.3.5 Dummy Versus Null Transfer Comparison

There are some differences in the way the EDMA3CC logic treats a dummy versus a null transfer request. A null transfer request is an error condition, but a dummy transfer is a legal transfer of 0 bytes. A null transfer causes an error bit ( $E_n$ ) in EMR to get set and the  $E_n$  bit in SER remains set, essentially preventing any further transfers on that channel without clearing the associated error registers.

[Table 15-2](#) summarizes the conditions and effects of null and dummy transfer requests.

**Table 15-2. Dummy and Null Transfer Request**

Feature	Null TR	Dummy TR
EMR/QEMR is set	Yes	No
SER/QSER remains set	Yes	No
Link update (STATIC = 0 in OPT)	Yes	Yes
QER is set	Yes	Yes
IPR and CER is set using early completion	Yes	Yes

### 15.2.3.6 Parameter Set Updates

When a TR is submitted for a given DMA/QDMA channel and its corresponding PaRAM set, the EDMA3CC is responsible for updating the PaRAM set in anticipation of the next trigger event. For nonfinal events, this includes address and count updates; for final events, this includes the link update.

The specific PaRAM set entries that are updated depend on the channel's synchronization type (A-synchronized or B-synchronized) and the current state of the PaRAM set. A B-update refers to the decrementing of BCNT in the case of A-synchronized transfers after the submission of successive TRs. A C-update refers to the decrementing of CCNT in the case of A-synchronized transfers after BCNT TRs for ACNT byte transfers have submitted. For AB-synchronized transfers, a C-update refers to the decrementing of CCNT after submission of every transfer request.

See [Table 15-3](#) for details and conditions on the parameter updates. A link update occurs when the PaRAM set is exhausted, as described in [Section 15.2.3.7](#).

After the TR is read from the PaRAM (and is in process of being submitted to EDMA3TC), the following fields are updated if needed:

- A-synchronized: BCNT, CCNT, SRC, DST
- AB-synchronized: CCNT, SRC, DST

The following fields are not updated (except for during linking, where all fields are overwritten by the link PaRAM set):

- A-synchronized: ACNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK
- AB-synchronized: ACNT, BCNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK

Note that PaRAM updates only pertain to the information that is needed to properly submit the next transfer request to the EDMA3TC. Updates that occur while data is moved within a transfer request are tracked within the transfer controller, and is detailed in [Section 15.2.11](#). For A-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes (BCNT = 1 and CCNT = 1). For AB-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes of BCNT arrays (CCNT = 1). The EDMA3TC is responsible for updating source and destination addresses within the array based on ACNT and FWID (in OPT). For AB-synchronized transfers, the EDMA3TC is also responsible to update source and destination addresses between arrays based on SRCBIDX and DSTBIDX.

[Table 15-3](#) shows the details of parameter updates that occur within EDMA3CC for A-synchronized and AB-synchronized transfers.

**Table 15-3. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set)**

Condition:	A-Synchronized Transfer			AB-Synchronized Transfer		
	B-Update	C-Update	Link Update	B-Update	C-Update	Link Update
	BCNT > 1	BCNT == 1 && CCNT > 1	BCNT == 1 && CCNT == 1	N/A	CCNT > 1	CCNT == 1
SRC	+= SRCBIDX	+= SRCCIDX	= Link.SRC	in EDMA3TC	+= SRCCIDX	= Link.SRC
DST	+= DSTBIDX	+= DSTCIDX	= Link.DST	in EDMA3TC	+= DSTCIDX	= Link.DST
ACNT	None	None	= Link.ACNT	None	None	= Link.ACNT
BCNT	-= 1	= BCNTRLD	= Link.BCNT	in EDMA3TC	N/A	= Link.BCNT
CCNT	None	-= 1	= Link.CCNT	in EDMA3TC	-= 1	= Link.CCNT
SRCBIDX	None	None	= Link.SRCBIDX	in EDMA3TC	None	= Link.SRCBIDX
DSTBIDX	None	None	= Link.DSTBIDX	None	None	= Link.DSTBIDX
SRCCIDX	None	None	= Link.SRCBIDX	in EDMA3TC	None	= Link.SRCBIDX
DSTCIDX	None	None	= Link.DSTBIDX	None	None	= Link.DSTBIDX
LINK	None	None	= Link.LINK	None	None	= Link.LINK
BCNTRLD	None	None	= Link.BCNTRLD	None	None	= Link.BCNTRLD
OPT <sup>(1)</sup>	None	None	= LINK.OPT	None	None	= LINK.OPT

<sup>(1)</sup> In all cases, no updates occur if OPT.STATIC == 1 for the current PaRAM set.

**NOTE:** The EDMA3CC includes no special hardware to detect when an indexed address update calculation overflows/underflows. The address update will wrap across boundaries as programmed by the user. You should ensure that no transfer is allowed to cross internal port boundaries between peripherals. A single TR must target a single source/destination slave endpoint.

### 15.2.3.7 Linking Transfers

The EDMA3CC provides a mechanism known as linking, which allows the entire PaRAM set to be reloaded from a location within the PaRAM memory map (for both DMA and QDMA channels). Linking is especially useful for maintaining ping-pong buffers, circular buffering, and repetitive/continuous transfers all with no CPU intervention. Upon completion of a transfer, the current transfer parameters are reloaded with the parameter set pointed to by the 16-bit link address field (of the current parameter set). Linking only occurs when the STATIC bit in OPT is cleared to 0.

---

**NOTE:** A transfer (DMA or QDMA) should always be linked to another useful transfer. If it is required to terminate a transfer, the transfer should be linked to a NULL set.

---

The link update occurs after the current PaRAM set event parameters have been exhausted. An event's parameters are exhausted when the EDMA3 channel controller has submitted all the transfers associated with the PaRAM set.

A link update occurs for null and dummy transfers depending on the state of the STATIC bit in OPT and the LINK field. In both cases (null or dummy), if the value of LINK is FFFFh then a null PaRAM set (with all 0s and LINK set to FFFFh) is written to the current PaRAM set. Similarly, if LINK is set to a value other than FFFFh then the appropriate PaRAM location pointed to by LINK is copied to the current PaRAM set.

Once the channel completion conditions are met for an event, the transfer parameters located at the link address are loaded into the current DMA or QDMA channel's associated parameter set. The EDMA3CC reads the entire PaRAM set (8 words) from the PaRAM set specified by LINK and writes all 8 words to the PaRAM set associated with the current channel. [Figure 15-8](#) shows an example of a linked transfer.

Any PaRAM set in the PaRAM can be used as a link/reload parameter set; however, it is recommended that the PaRAM sets associated with peripheral synchronization events (see [Section 15.2.6](#)) should only be used for linking if the synchronization event isolated with the channel mapped to that PaRAM set is disabled.

If a PaRAM set location is mapped to a QDMA channel (by QCHMAP $n$ ), then copying the link PaRAM set onto the current QDMA channel PaRAM set is recognized as a trigger event and is latched in QER since a write to the trigger word was performed. This feature can be used to create a linked list of transfers using a single QDMA channel and multiple PaRAM sets.

Link-to-self transfers replicate the behavior of autoinitialization, which facilitates the use of circular buffering and repetitive transfers. After an EDMA3 channel exhausts its current PaRAM set, it reloads all the parameter set entries from another PaRAM set, which is initialized with values identical to the original PaRAM set. [Figure 15-9](#) shows an example of a linked-to-self transfer. In [Figure 15-9](#), parameter set 127 has the LINK field address pointing to the address of parameter set 127, that is, linked-to-self.

---

**NOTE:** If the STATIC bit in OPT is set for a PaRAM set, then link updates are not performed. The link updates performed internally by the EDMA3CC are atomic. This implies that when the EDMA3CC is updating a PaRAM set, accesses to PaRAM by other EDMA3 programmer's (for example, CPU configuration accesses) are not allowed. Also for QDMA, for example, if the first word of the PaRAM entry is defined as a trigger word, EDMA3CC logic assures that all 8 PaRAM words are updated before the new QDMA event can trigger the transfer for that PaRAM entry.

---

### 15.2.3.7.1 Constant Addressing Mode Transfers/Alignment Issues

If either SAM or DAM is set to 1 (constant addressing mode), then the source or destination address must be aligned to a 256-bit aligned address, respectively, and the corresponding BIDX should be an even multiple of 32 bytes (256 bit). The EDMA3CC does not recognize errors here but the EDMA3TC asserts an error, if this is not true. See [Section 15.2.11.2](#).

---

**NOTE:** The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. On the C674x/OMAP-L1x processors, no peripherals, memory, or memory controller support constant addressing mode. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM = 0) by appropriately programming the count and indices values.

---

### 15.2.3.7.2 Element Size

The EDMA3 controller does not use the concept of element-size and element-indexing. Instead, all transfers are defined in terms of all three dimensions: ACNT, BCNT, and CCNT. An element-indexed transfer is logically achieved by programming ACNT to the size of the element and BCNT to the number of elements that need to be transferred. For example, if you have 16-bit audio data and 256 audio samples that needed to be transferred to a serial port, this can be done by programming the ACNT = 2 (2 bytes) and BCNT = 256.

Figure 15-8. Linked Transfer Example

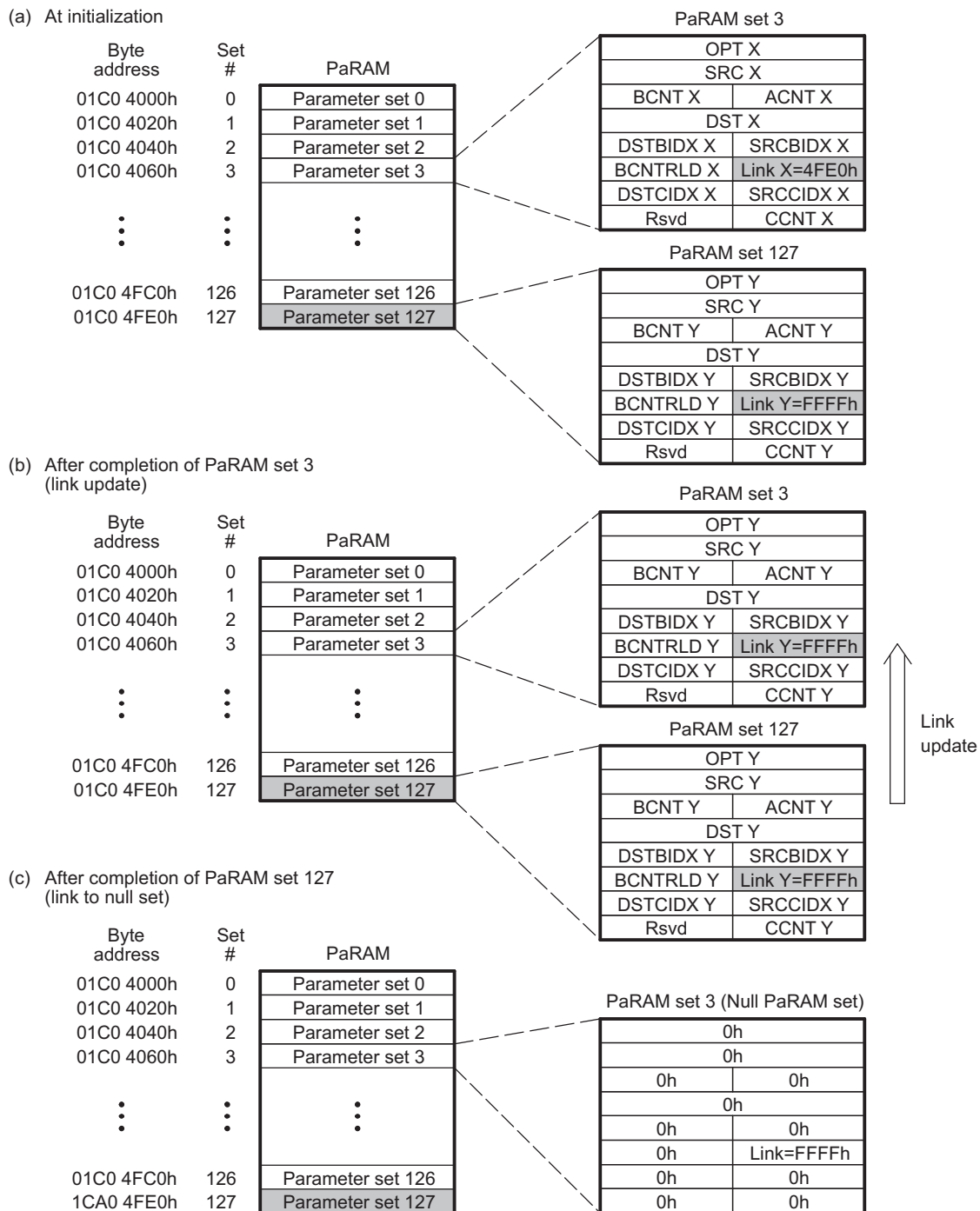
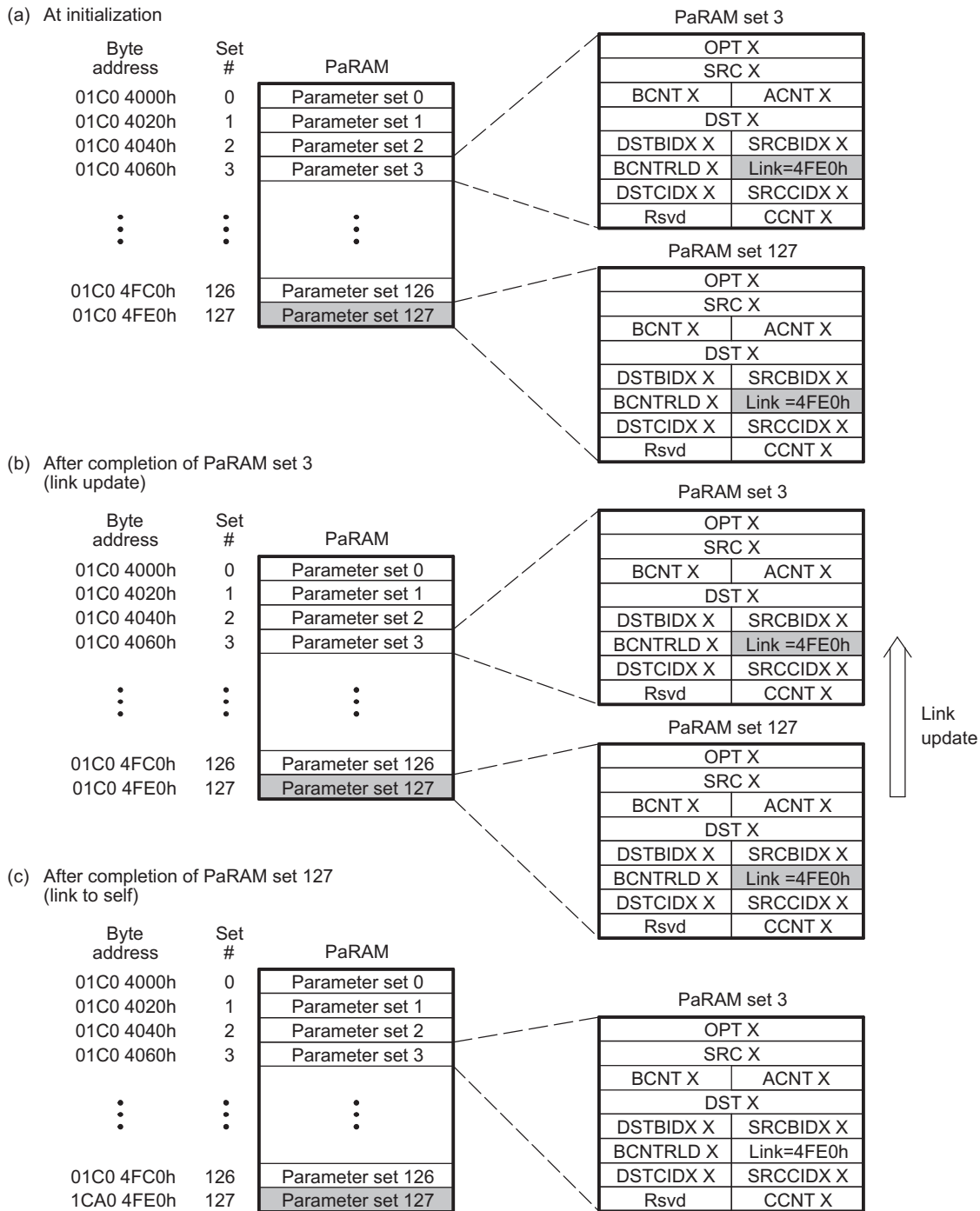


Figure 15-9. Link-to-Self Transfer Example



## 15.2.4 Initiating a DMA Transfer

There are multiple ways to initiate a programmed data transfer using the EDMA3 channel controller. Transfers on DMA channels are initiated by three sources:

- **Event-triggered transfer request** (this is the more typical usage of EDMA3): Allows for a peripheral, system, or externally-generated event to trigger a transfer request.
- **Manually-triggered transfer request:** The CPU manually triggers a transfer by writing a 1 to the corresponding bit in the event set register (ESR).
- **Chain-triggered transfer request:** A transfer is triggered on the completion of another transfer or subtransfer.

Transfers on QDMA channels are initiated by two sources:

- **Autotriggered transfer request:** A transfer is triggered when the PaRAM set entry programmed trigger word is written to.
- **Link-triggered transfer requests:** When linking occurs, the transfer is triggered when the PaRAM set entry programmed trigger word is written to.

### 15.2.4.1 DMA Channel

#### 15.2.4.1.1 Event-Triggered Transfer Request

When an event is asserted from a peripheral or device pins, it gets latched in the corresponding bit of the event register ( $ER.En = 1$ ). If the corresponding event in the event enable register (EER) is enabled ( $EER.En = 1$ ), then the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

If the PaPARAM set is valid (not a NULL set), then a transfer request packet (TRP) is submitted to the EDMA3TC and the  $En$  bit in ER is cleared. At this point, a new event can be safely received by the EDMA3CC.

If the PaPARAM set associated with the channel is a NULL set (see [Section 15.2.3.3](#)), then no transfer request (TR) is submitted and the corresponding  $En$  bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register ( $EMR.En = 1$ ) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include cleaning the event missed error before retriggering the DMA channel.

When an event is received, the corresponding event bit in the event register is set ( $ER.En = 1$ ), regardless of the state of  $EER.En$ . If the event is disabled when an external event is received ( $ER.En = 1$  and  $EER.En = 0$ ), the  $ER.En$  bit remains set. If the event is subsequently enabled ( $EER.En = 1$ ), then the pending event is processed by the EDMA3CC and the TR is processed/submitted, after which the  $ER.En$  bit is cleared.

If an event is being processed (prioritized or is in the event queue) and another sync event is received for the same channel prior to the original being cleared ( $ER.En \neq 0$ ), then the second event is registered as a missed event in the corresponding bit of the event missed register ( $EMR.En = 1$ ).

For the synchronization events associated with each of the programmable DMA channels, see your device-specific data manual to determine the event to channel mapping.



### 15.2.4.1.2 Manually-Triggered Transfer Request

A DMA transfer is initiated by a write to the event set register (ESR) by the CPU (or any EDMA programmer). Writing a 1 to an event bit in the ESR results in the event being prioritized/queued in the appropriate event queue, regardless of the state of the EER.En bit. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see [Section 15.2.3.3](#)), then no transfer request (TR) is submitted and the corresponding En bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.En = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include clearing the event missed error before retriggering the DMA channel.

If an event is being processed (prioritized or is in the event queue) and the same channel is manually set by a write to the corresponding channel bit of the event set register (ESR.En = 1) prior to the original being cleared (ESR.En = 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.En = 1).

### 15.2.4.1.3 Chain-Triggered Transfer Request

Chaining is a mechanism by which the completion of one transfer automatically sets the event for another channel. When a chained completion code is detected, the value of which is dictated by the transfer completion code (TCC[5:0] in OPT of the PaRAM set associated with the channel), it results in the corresponding bit in the chained event register (CER) to be set (CER.E[TCC] = 1).

Once a bit is set in CER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see [Section 15.2.3.3](#)), then no transfer request (TR) is submitted and the corresponding En bit in CER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.En = 1) to indicate that the event was discarded due to a null TR being serviced. In this case, the error condition must be cleared by you before the DMA channel can be retriggered. Good programming practices might include clearing the event missed error before retriggering the DMA channel.

If a chaining event is being processed (prioritized or queued) and another chained event is received for the same channel prior to the original being cleared (CER.En != 0), then the second chained event is registered as a missed event in the corresponding channel bit of the event missed register (EMR.En = 1).

---

**NOTE:** Chained event registers, event registers, and event set registers operate independently. An event (En) can be triggered by any of the trigger sources (event-triggered, manually-triggered, or chain-triggered).

---



## 15.2.4.2 QDMA Channels

### 15.2.4.2.1 Autotriggered and Link-Triggered Transfer Request

---

**NOTE:** If OPT, SRC, or DST is the trigger word for a QDMA transfer then it is required to do a 32-bit access to that field.

---

QDMA-based transfer requests are issued when a QDMA event gets latched in the QDMA event register ( $QER.En = 1$ ). A bit corresponding to a QDMA channel is set in the QDMA event register (QER) when the following occurs:

- A CPU (or any EDMA3 programmer) write occurs to a PaRAM address that is defined as a QDMA channel trigger word (programmed in the QDMA channel  $n$  mapping register ( $QCHMAPn$ )) for the particular QDMA channel and the QDMA channel is enabled via the QDMA event enable register ( $QEER.En = 1$ ).
- EDMA3CC performs a link update on a PaRAM set address that is configured as a QDMA channel (matches  $QCHMAPn$  settings) and the corresponding channel is enabled via the QDMA event enable register ( $QEER.En = 1$ ).

Once a bit is set in QER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If a bit is already set in QER ( $QER.En = 1$ ) and a second QDMA event for the same QDMA channel occurs prior to the original being cleared, the second QDMA event gets captured in the QDMA event miss register ( $QEMR.En = 1$ ).

### 15.2.4.3 Comparison Between DMA and QDMA Channels

The primary difference between DMA and QDMA channels is the event/channel synchronization. QDMA events are either autotriggered or link triggered. Autotriggering allows QDMA channels to be triggered by CPU(s) with a minimum number of linear writes to PaRAM. Link triggering allows a linked list of transfers to be executed, using a single QDMA PaRAM set and multiple link PaRAM sets.

A QDMA transfer is triggered when a CPU (or other EDMA3 programmer) writes to the trigger word of the QDMA channel parameter set (autotriggered) or when the EDMA3CC performs a link update on a PaRAM set that has been mapped to a QDMA channel (link triggered). Note that for CPU triggered (manually triggered) DMA channels, in addition to writing to the PaRAM set, it is required to write to the event set register (ESR) to kick-off the transfer.

QDMA channels are typically for cases where a single event will accomplish a complete transfer since the CPU (or EDMA3 programmer) must reprogram some portion of the QDMA PaRAM set in order to retrigger the channel. In other words, QDMA transfers are programmed with  $BCNT = CCNT = 1$  for A-synchronized transfers, and  $CCNT = 1$  for AB-synchronized transfers.

Additionally, since linking is also supported (if  $STATIC = 0$  in OPT) for QDMA transfers, it allows you to initiate a linked list of QDMAs, so when EDMA3CC copies over a link PaRAM set (including the write to the trigger word), the current PaRAM set mapped to the QDMA channel will automatically be recognized as a valid QDMA event and initiate another set of transfers as specified by the linked set.

### 15.2.5 Completion of a DMA Transfer

A parameter set for a given channel is complete when the required number of transfer requests is submitted (based on receiving the number of synchronization events). The expected number of TRs for a non-null/non-dummy transfer is shown in [Table 15-4](#) for both synchronization types along with state of the PaRAM set prior to the final TR being submitted. When the counts (BCNT and/or CCNT) are this value, the next TR results in a:

- Final chaining or interrupt codes to be sent by the transfer controllers (instead of intermediate).
- Link updates (linking to either null or another valid link set).

**Table 15-4. Expected Number of Transfers for Non-Null Transfer**

Sync Mode	Counts at time 0	Total # Transfers	Counts prior to final TR
A-synchronized	ACNT BCNT CCNT	(BCNT × CCNT ) TRs of ACNT bytes each	BCNT == 1 && CCNT == 1
AB-synchronized	ACNT BCNT CCNT	CCNT TRs for ACNT × BCNT bytes each	CCNT == 1

You must program the PaRAM OPT field with a specific transfer completion code (TCC) along with the other OPT fields (TCCHEN, TCINTEN, ITCCHEN, and ITCINTEN bits) to indicate whether the completion code is to be used for generating a chained event or/and for generating an interrupt upon completion of a transfer.

The specific TCC value (6-bit binary value) programmed dictates which of the 64-bits in the chain event register (CER[TCC]) and/or interrupt pending register (IPR[TCC]) is set.

See [Section 15.2.9](#) for details on interrupts and [Section 15.2.8](#) for details on chaining.

You can also selectively program whether the transfer controller sends back completion codes on completion of the final transfer request (TR) of a parameter set (TCCHEN or TCINTEN), for all but the final transfer request (TR) of a parameter set (ITCCHEN or ITCINTEN), or for all TRs of a parameter set (both). See [Section 15.2.8](#) for details on chaining (intermediate/final chaining) and [Section 15.2.9](#) for details on intermediate/final interrupt completion.

A completion detection interface exists between the EDMA3 channel controller and transfer controller(s). This interface sends back information from the transfer controller to the channel controller to indicate that a specific transfer is completed.

All DMA/QDMA PaRAM sets must also specify a link address value. For repetitive transfers such as ping-pong buffers, the link address value should point to another predefined PaRAM set. Alternatively, a nonrepetitive transfer should set the link address value to the null link value. The null link value is defined as FFFFh. See [Section 15.2.3.7](#) for more details.

---

**NOTE:** Any incoming events that are mapped to a null PaRAM set results in an error condition. The error condition should be cleared before the corresponding channel is used again. See [Section 15.2.3.5](#).

---

There are three ways the EDMA3CC gets updated/informed about a transfer completion: normal completion, early completion, and dummy/null completion. This applies to both chained events and completion interrupt generation.

### 15.2.5.1 Normal Completion

In normal completion mode (TCCMODE = 0 in OPT), the transfer or sub-transfer is considered to be complete when the EDMA3 channel controller receives the completion codes from the EDMA3 transfer controller. In this mode, the completion code to the channel controller is posted by the transfer controller after it receives a signal from the destination peripheral. Normal completion is typically used to generate an interrupt to inform the CPU that a set of data is ready for processing.

### 15.2.5.2 Early Completion

In early completion mode (TCCMODE = 1 in OPT), the transfer is considered to be complete when the EDMA3 channel controller submits the transfer request (TR) to the EDMA3 transfer controller. In this mode, the channel controller generates the completion code internally. Early completion is typically useful for chaining, as it allows subsequent transfers to be chained-triggered while the previous transfer is still in progress within the transfer controller, maximizing the overall throughput of the set of the transfers.

### 15.2.5.3 Dummy or Null Completion

This is a variation of early completion. Dummy or null completion is associated with a dummy set ([Section 15.2.3.4](#)) or null set ([Section 15.2.3.3](#)). In both cases, the EDMA3 channel controller does not submit the associated transfer request to the EDMA3 transfer controller(s). However, if the set (dummy/null) has the OPT field programmed to return completion code (intermediate/final interrupt/chaining completion), then it will set the appropriate bits in the interrupt pending register (IPR) or chained event register (CER). The internal early completion path is used by the channel controller to return the completion codes internally (that is, EDMA3CC generates the completion code).

## 15.2.6 Event, Channel, and PaRAM Mapping

Most of the DMA channels are tied to a specific hardware peripheral event, thus allowing transfers to be triggered by events from device peripherals or external hardware. A DMA channel typically requests a data transfer when it receives its event (apart from manually-triggered, chain-triggered, and other transfers). The amount of data transferred per synchronization event depends on the channel's configuration (ACNT, BCNT, CCNT, etc.) and the synchronization type (A-synchronized or AB-synchronized).

The association of an event to a channel is fixed. Each of the DMA channels has one specific event associated with it. For the synchronization events associated with each of the programmable DMA channels, see your device-specific data manual to determine the event to channel mapping.

If in an application, a channel does not make use of the associated synchronization event or does not have an associated synchronization event (unused), that channel can be used for manually-triggered or chained-triggered transfers, for linking/reloading, or as a QDMA channel.

### 15.2.6.1 DMA Channel to PaRAM Mapping

The mapping between the DMA channel numbers and the PaRAM sets is a fixed, one-to-one mapping (see [Table 15-5](#)). In other words, channel (event) 0 is mapped to PaRAM set 0, channel (event 1) is mapped to PaRAM set 1, etc. So, for example, in order to program a transfer for event number 3, DMA channel 3 is associated with PaRAM set number 3 and you need to program this PaRAM set for configuring transfers associated with event number 3. See your device-specific data manual for the addresses of the PaRAM set entries.

**Table 15-5. EDMA3 DMA Channel to PaRAM Mapping**

PaRAM Set Number	Mapping
PaRAM Set 0	DMA Channel 0/Reload/QDMA
PaRAM Set 1	DMA Channel 1/Reload/QDMA
PaRAM Set 2	DMA Channel 2/Reload/QDMA
PaRAM Set 3	DMA Channel 3/Reload/QDMA
PaRAM Set 4	DMA Channel 4/Reload/QDMA
PaRAM Set 5	DMA Channel 5/Reload/QDMA
PaRAM Set 6	DMA Channel 6/Reload/QDMA
PaRAM Set 7	DMA Channel 7/Reload/QDMA
PaRAM Set 8	DMA Channel 8/Reload/QDMA
PaRAM Set 9	DMA Channel 9/Reload/QDMA
PaRAM Set 10	DMA Channel 10/Reload/QDMA
PaRAM Set 11	DMA Channel 11/Reload/QDMA
PaRAM Set 12	DMA Channel 12/Reload/QDMA
PaRAM Set 13	DMA Channel 13/Reload/QDMA
PaRAM Set 14	DMA Channel 14/Reload/QDMA
PaRAM Set 15	DMA Channel 15/Reload/QDMA
PaRAM Set 16	DMA Channel 16/Reload/QDMA
...	...
PaRAM Set 30	DMA Channel 30/Reload/QDMA
PaRAM Set 31	DMA Channel 31/Reload/QDMA
PaRAM Set 32	Reload/QDMA
PaRAM Set 33	Reload/QDMA
...	...
PaRAM Set $n - 2$	Reload/QDMA
PaRAM Set $n - 1$	Reload/QDMA
PaRAM Set $n$	Reload/QDMA

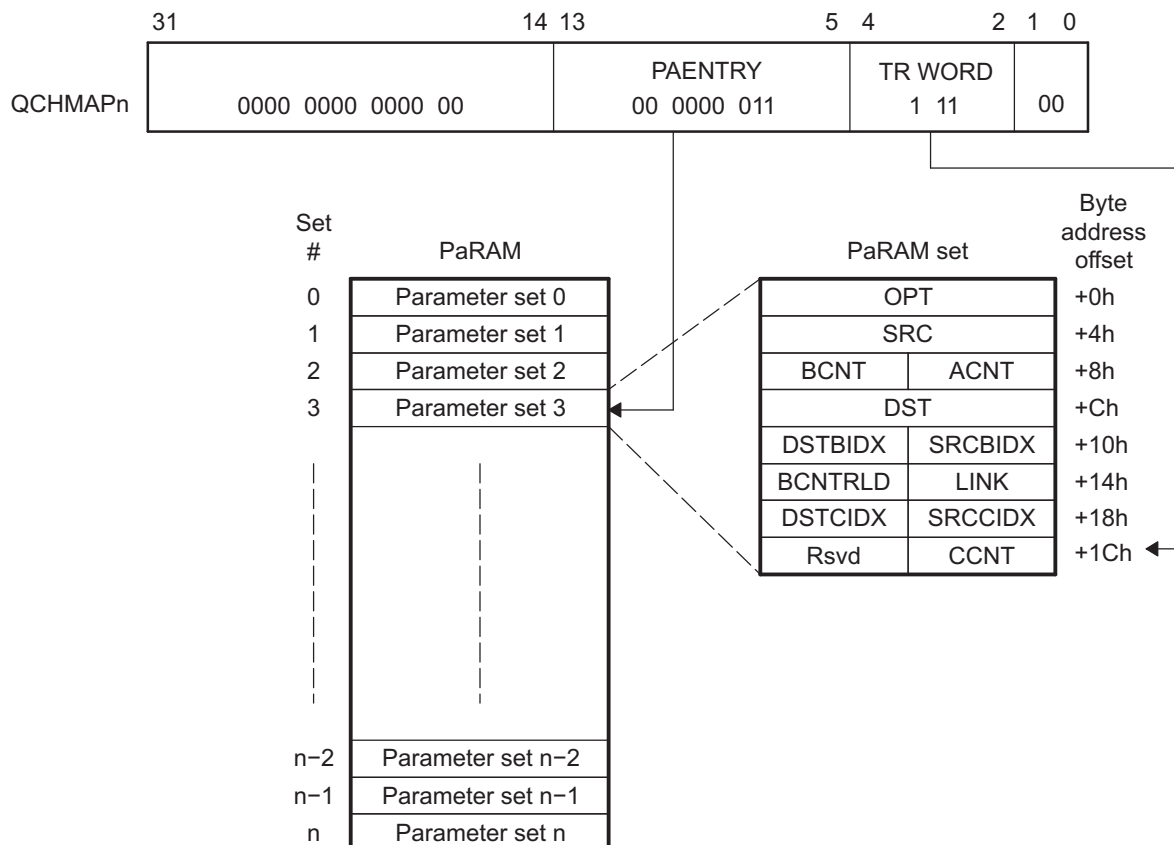
### 15.2.6.2 QDMA Channel to PaRAM Mapping

The mapping between the QDMA channels and the PaRAM sets is programmable. The QDMA channel  $n$  mapping register (QCHMAP $n$ ) in the EDMA3CC provides programmability for the QDMA channels to be mapped to any of the PaRAM sets in the PaRAM memory map. Figure 15-10 illustrates the use of QCHMAP.

Additionally, QCHMAP allows you to program the trigger word in the PaRAM set for the QDMA channel. A trigger word is one of the 8 words in the PaRAM set. For a QDMA transfer to occur, a valid TR synchronization event for EDMA3CC is a write to the trigger word in the PaRAM set pointed to by QCHMAP for a particular QDMA channel.

**NOTE:** By default, QDMA channels are mapped to PaRAM set 0. Care must be taken to appropriately remap PaRAM set 0 before it is used.

**Figure 15-10. QDMA Channel to PaRAM Mapping**



Note:  $n$  is the number of PaRAM sets supported in the EDMA3CC for a specific device.

## 15.2.7 EDMA3 Channel Controller Regions

The EDMA3 channel controller (EDMA3CC) divides its address space into multiple regions. Individual channel resources can be exclusively assigned to a specific region, where each region is typically assigned to a specific EDMA programmer. This allows partitioning of EDMA channel (DMA/QDMA) resources in hetero- or multi-core devices, and devices where certain additional masters (for example, coprocessors) can also program/initiate EDMA3 transfers. The application software running on these cores/coprocessors can operate in these exclusive shadow region memory-maps, minimizing possibilities of resource conflicts.

### 15.2.7.1 Region Overview

The EDMA3CC memory-mapped registers are divided in three main categories:

1. Global registers
2. Global region channel registers
3. Shadow region channel registers

The global registers are located at a single/fixed location in the EDMA3CC memory map. These registers control EDMA3 resource mapping and provide debug visibility and error tracking information. See your device-specific data manual for the EDMA3CC memory map.

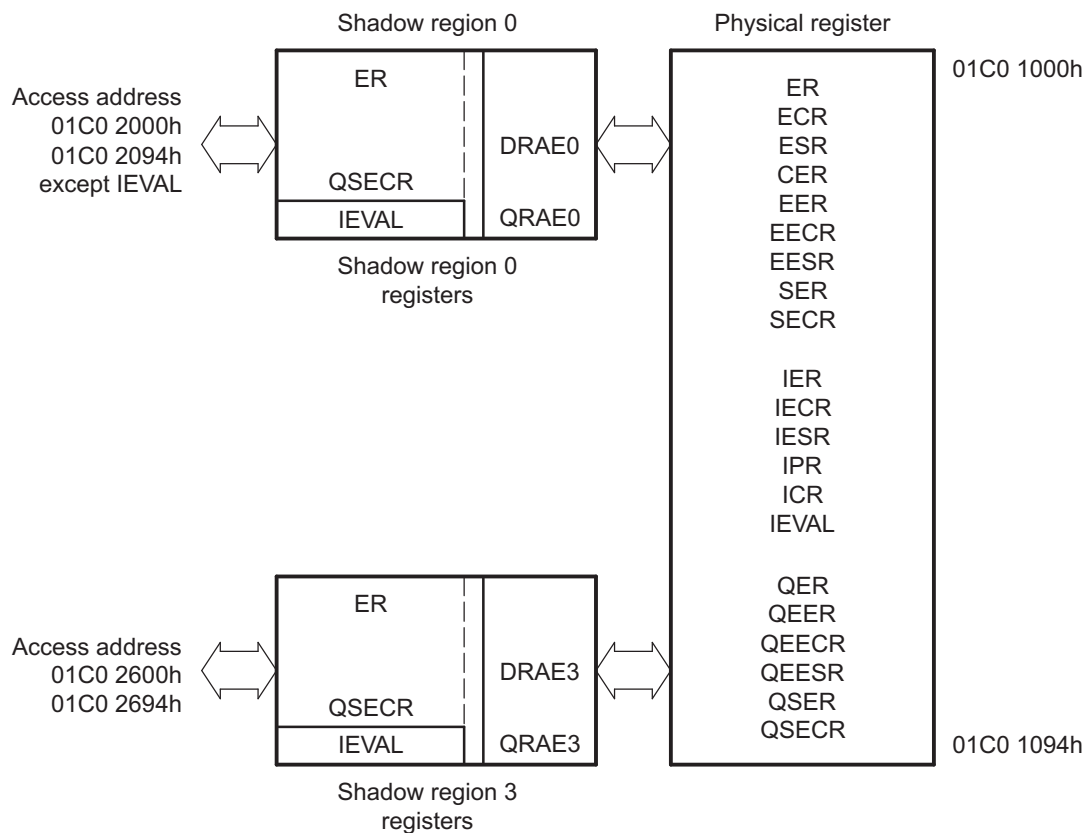
The channel registers (including DMA, QDMA, and interrupt registers) are accessible via the global channel region address range, or in the shadow  $n$  channel region address range(s). For example, the event enable register (EER) is visible in the global region register space at offset 1020h, or region addresses at offset 2020h for region 0 and at offset 2220h for region 1.

The underlying control register bits that are accessible via the shadow region address space (except for IEVAL $n$ ) are controlled by the DMA region access enable registers (DRAE $m$ ) and QDMA region access enable registers (QRAE $m$ ). [Table 15-6](#) lists the registers in the shadow region memory-map. (See EDMA3CC memory-map figure for the complete global and shadow region memory-maps.) [Figure 15-11](#) illustrates the conceptual view of the regions (where  $n$  is the number of shadow regions supported in the EDMA3CC for a specific device).

**Table 15-6. Shadow Region Registers**

DRAE $m$	QRAE $m$
ER	QER
ECR	QEER
ESR	QEECR
CER	QEESR
EER	
EECR	
EESR	
SER	
SECR	
IER	
IECR	
IESR	
IPR	
ICR	
<b>Register not affected by DRAE</b>	
IEVAL	

**Figure 15-11. Shadow Region Registers**



### 15.2.7.2 Channel Controller Shadow Regions

For each EDMA3 shadow region (and associated memory-maps) there is a set of registers associated with the shadow region that allows association of the DMA/QDMA channels and interrupt completion codes to the region. These registers are user-programmed per region to assign ownership of the DMA/QDMA channels and TCC values to a region.

- **DRAEm:** One register exists for each of the shadow regions. The number of bits in each register matches the number of DMA channels. These registers need to be programmed to assign ownership of DMA channels to the respective region. Accesses to DMA event registers and interrupt registers via the shadow region address map are filtered through DRAE. A value of 1 in the corresponding DRAE bit implies that the corresponding DMA/interrupt channel is accessible; a value of 0 in the corresponding DRAE bit forces writes to be discarded and returns a value of 0 for reads.
- **QRAEm:** One register exists for every region. The number of bits in each register matches the number of QDMA channels. These registers must be programmed to assign ownership of QDMA channels to the respective region. To enable a channel in a shadow region using shadow region 0 QEER, the respective bit in QRAE must be set or writing into QEESR will not have the desired effect.

It is typical for an application to have a unique assignment of QDMA/DMA channels (and, therefore, a given bit position) to a given region.

The use of shadow regions allows for restricted access to EDMA3 resources (DMA channels, QDMA channels, TCC, interrupts) by tasks/cores/EDMA3 programmers in a system by setting or clearing bits in the DRAE/QRAE registers. If exclusive access to any given channel/TCC code is required for a region, then only that region's DRAE/QRAE should have the associated bit set.

Additionally, with each shadow region, there is an associated shadow region completion interrupt (EDMA3CC\_INT $n$  where  $n$  denotes the shadow region number). For multi-core/hetero-core devices, the various shadow region interrupts might be tied to the interrupt controllers for different cores. For single core devices, all shadow region interrupts would be routed to the device interrupt controller. See your device-specific data manual for the shadow region interrupt hookup to the device interrupt controller(s). The DRAE associated with each shadow region acts as a secondary interrupt enable (along with the interrupt enable register) for the respective shadow region interrupts. See [Section 15.2.9](#) for more information on interrupts.

#### Example 15-1. Resource Pool Division Across Two Regions

This example illustrates a resource pool division across two regions, assuming region 0 must be allocated 16 DMA channels (0-15) and 1 QDMA channel (0), and 16 TCC codes (0-15). Region 1 needs to be allocated 16 DMA channels (16-31) and 7 QDMA channels (1-7), and 16 TCC codes (16-31). DRAE should be equal to the OR of the bits that are required for the DMA channels and the TCC codes:

```
Region 0: DRAE = 0x0000FFFF QRAE = 0x00000001 Region 1: DRAE = 0xFFFF0000 QRAE = 0x000000FE
```

### 15.2.8 Chaining EDMA3 Channels

The channel chaining capability for the EDMA3 allows the completion of an EDMA3 channel transfer to trigger another EDMA3 channel transfer. The purpose is to allow you the ability to chain several events through one event occurrence.

Chaining is different from linking ([Section 15.2.3.7](#)). The EDMA3 link feature reloads the current channel parameter set with the linked parameter set. The EDMA3 chaining feature does not modify or update any channel parameter set; it provides a synchronization event to the chained channel (see [Section 15.2.4.1.3](#) for chain-triggered transfer requests).

Chaining is achieved at either final transfer completion or intermediate transfer completion, or both, of the current channel. Consider a channel  $m$  (DMA/QDMA) required to chain to channel  $n$ . Channel number  $n$  (0-31) needs to be programmed into the TCC field of channel  $m$  channel options parameter (OPT) set.

- If final transfer completion chaining (TCCHEN = 1 and ITCCHEN = 0 in channel  $m$  OPT) is enabled, the chain-triggered event occurs after the *last* transfer request of channel  $m$  is submitted (early completion) or completed (normal completion).



- If intermediate transfer completion chaining (TCCHEN = 0 and ITCCHEN = 0 in channel *m* OPT) is enabled, the chain-triggered event occurs after every *intermediate* transfer request of channel *m* is submitted (early completion) or completed (normal completion).
- If both final and intermediate transfer completion chaining (TCCHEN = 1 and ITCCHEN = 1 in channel *m* OPT) are enabled, the chain-trigger event occurs after every transfer request of channel *m* is submitted (early completion) or completed (normal completion).

Table 15-7 shows the number of chain event triggers occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

**Table 15-7. Chain Event Triggers**

Options	(Number of chained event triggers on channel 30)	
	A-Synchronized	AB-Synchronized
TCCHEN = 1, ITCCHEN = 0	1 (Last TR)	1 (Last TR)
TCCHEN = 0, ITCCHEN = 1	19 (All but the last TR)	4 (All but the last TR)
TCCHEN = 1, ITCCHEN = 1	20 (All TRs)	5 (All TRs)

### 15.2.9 EDMA3 Interrupts

The EDMA3 interrupts are divided into 2 categories:

- Transfer completion interrupts
- Error interrupts

The transfer completion interrupts are listed in Table 15-8. The error interrupts are listed in Table 15-9.

**Table 15-8. EDMA3 Transfer Completion Interrupts**

Name	Description	DSPINTC
EDMA3_0_CC0_INT0	EDMA3_0 Channel Controller 0 Shadow Region 0 Transfer Completion Interrupt	—
EDMA3_0_CC0_INT1	EDMA3_0 Channel Controller 0 Shadow Region 1 Transfer Completion Interrupt	8
EDMA3_0_CC0_INT2	EDMA3_0 Channel Controller 0 Shadow Region 2 Transfer Completion Interrupt	—
EDMA3_0_CC0_INT3	EDMA3_0 Channel Controller 0 Shadow Region 3 Transfer Completion Interrupt	—
EDMA3_1_CC0_INT0	EDMA3_1 Channel Controller 0 Shadow Region 0 Transfer Completion Interrupt	—
EDMA3_1_CC0_INT1	EDMA3_1 Channel Controller 0 Shadow Region 1 Transfer Completion Interrupt	91
EDMA3_1_CC0_INT2	EDMA3_1 Channel Controller 0 Shadow Region 2 Transfer Completion Interrupt	—
EDMA3_1_CC0_INT3	EDMA3_1 Channel Controller 0 Shadow Region 3 Transfer Completion Interrupt	—

**Table 15-9. EDMA3 Error Interrupts**

Name	Description	DSPINTC
EDMA3_0_CC0_ERRINT	EDMA3_0 Channel Controller 0 Error Interrupt	56
EDMA3_0_TC0_ERRINT	EDMA3_0 Transfer Controller 0 Error Interrupt	57
EDMA3_0_TC1_ERRINT	EDMA3_0 Transfer Controller 1 Error Interrupt	58
EDMA3_1_CC0_ERRINT	EDMA3_1 Channel Controller 0 Error Interrupt	92
EDMA3_1_TC0_ERRINT	EDMA3_1 Transfer Controller 0 Error Interrupt	93

### 15.2.9.1 Transfer Completion Interrupts

The EDMA3CC is responsible for generating transfer completion interrupts to the CPU. The EDMA3 generates a single completion interrupt per shadow region on behalf of all DMA/QDMA channels. Various control registers and bit fields facilitate EDMA3 interrupt generation.

The transfer completion code (TCC) value is directly mapped to the bits of the interrupt pending register (IPR), as shown in [Table 15-10](#). For example, if TCC = 00 0000b, IPR[0] is set after transfer completion, and results in an interrupt generation to the CPU if in the EDMA3CC and device interrupt controller are configured to allow a CPU interrupt. See [Section 15.2.9.1.1](#) for details on enabling EDMA3 transfer completion interrupts.

When a completion code is returned (as a result of early or normal completion), the corresponding bit in IPR is set. For the completion code to be returned, the PaRAM set associated with the transfer must enable the transfer completion interrupt (final/intermediate) in the channel options parameter (OPT).

The transfer completion code (TCC) can be programmed to any value for a DMA/QDMA channel. There does not need to be a direct relation between the channel number and the transfer completion code value. This allows multiple channels having the same transfer completion code value to cause a CPU to execute the same interrupt service routine (ISR) for different channels.

---

**NOTE:** The TCC field in the channel options parameter (OPT) is a 6-bit field and can be programmed for any value between 0-64. For devices with 32 DMA channels, the TCC should have a value between 0 to 31 so that it sets the appropriate bits (0 to 31) in IPR (and can interrupt the CPU(s) on enabling the IER register bits (0-31)).

---

**Table 15-10. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping**

TCC Bits in OPT (TCINTEN/ITCINTEN = 1)	IPR Bit Set
00 0000b	IPR0
00 0001b	IPR1
00 0010b	IPR2
00 0011b	IPR3
00 0100b	IPR4
...	...
...	...
01 1110b	IPR30
01 1111b	IPR31

You can enable interrupt generation at either final transfer completion or intermediate transfer completion, or both. Consider channel  $m$  as an example.

- If the final transfer interrupt (TCINTEN = 1 and ITCINTEN = 0 in OPT) is enabled, the interrupt occurs after the *last* transfer request of channel  $m$  is either submitted or completed (depending on early or normal completion).
- If the intermediate transfer interrupt (TCINTEN = 0 and ITCINTEN = 1 in OPT) is enabled, the interrupt occurs after *every intermediate* transfer request of channel  $m$  is either submitted or completed (depending on early or normal completion).
- If both final and intermediate transfer completion interrupts (TCINTEN = 1 and ITCINTEN = 1 in OPT) are enabled, the interrupt occurs after *every* transfer request of channel  $m$  is submitted or completed (depending on early or normal completion).

[Table 15-11](#) shows the number of interrupts occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

**Table 15-11. Number of Interrupts**

Options	A-Synchronized	AB-Synchronized
TCINTEN = 1, ITCINTEN = 0	1 (Last TR)	1 (Last TR)
TCINTEN = 0, ITCINTEN = 1	19 (All but the last TR)	4 (All but the last TR)
TCINTEN = 1, ITCINTEN = 1	20 (All TRs)	5 (All TRs)

#### 15.2.9.1.1 Enabling Transfer Completion Interrupts

For the EDMA3 channel controller to assert a transfer completion to the external world, the interrupts have to be enabled in the EDMA3CC. This is in addition to setting up the TCINTEN and ITCINTEN bits in OPT of the associated PaRAM set.

The EDMA3 channel controller has interrupt enable registers (IER) and each bit location in IER serves as a primary enable for the corresponding interrupt pending register (IPR).

All the interrupt registers (IER, IESR, IECR, and IPR) are either manipulated from the global DMA channel region or by way of the DMA channel shadow regions. The shadow regions provide a view to the same set of physical registers that are in the global region.

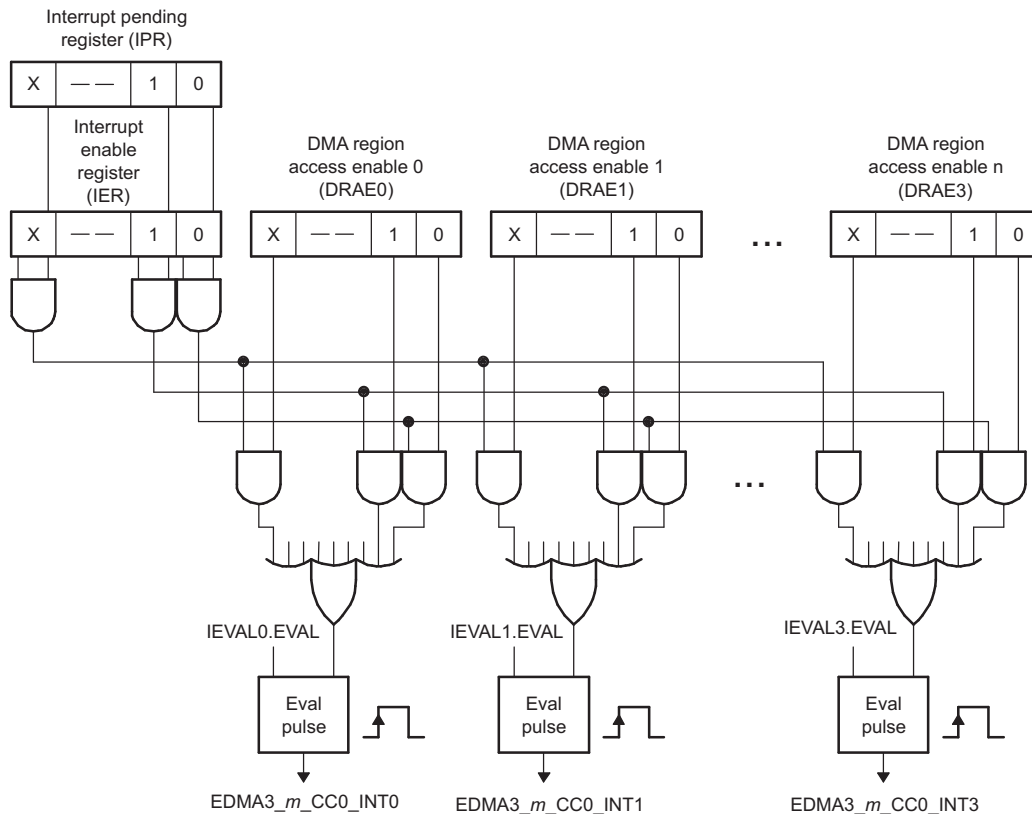
The EDMA3 channel controller has a hierarchical completion interrupt scheme that makes use of a single set of interrupt pending register (IPR) and single set of interrupt enable registers (IER). A second level of interrupt masking is provided by the programmable DMA region access enable registers (DRAE). See [Figure 15-12](#).

For the EDMA3CC to generate the transfer completion interrupts that are associated with each shadow region, the following conditions must be true:

- EDMA3CC\_INT0: (IPR.E0 & IER.E0 & DRAE0.E0) | (IPR.E1 & IER.E1 & DRAE0.E1) | ... | (IPR.En & IER.En & DRAE0.En)
- EDMA3CC\_INT1: (IPR.E0 & IER.E0 & DRAE1.E0) | (IPR.E1 & IER.E1 & DRAE1.E1) | ... | (IPR.En & IER.En & DRAE1.En)

where  $n$  is the number of shadow regions supported in the EDMA3CC for a specific device.

Figure 15-12. Interrupt Diagram



**NOTE:** The DRAE for all regions is expected to be set up at system initialization and to remain static for an extended period of time. The interrupt enable registers should be used for dynamic enable/disable of individual interrupts.

Because there is no relation between the TCC value and the DMA/QDMA channel, it is possible, for example, for DMA channel 0 to have the OPT.TCC = 31 in its associated PaPARAM set. This would mean that if a transfer completion interrupt is enabled (OPT.TCINTEN or OPT.ITCINTEN is set), then based on the TCC value, IPR.E31 is set up on completion. For proper channel operations and interrupt generation using the shadow region map, you must program the DRAE that is associated with the shadow region to have read/write access to both bit 0 (corresponding to channel 0) and bit 31 (corresponding to IPR.E31 bit that is set upon completion).

### 15.2.9.1.2 Clearing Transfer Completion Interrupts

Transfer completion interrupts that are latched to the interrupt pending register (IPR) is cleared by writing a 1 to the corresponding bit in the interrupt pending clear register (ICR). For example, a write of 1 to ICR.E0 clears a pending interrupt in IPR.E0.

If an incoming transfer completion code (TCC) gets latched to a bit in IPR, then additional bits that get set due to a subsequent transfer completion will not result in asserting the EDMA3CC completion interrupt. In order for the completion interrupt to be pulsed, the required transition is from a state where no enabled interrupts are set to a state where at least one enabled interrupt is set.

### 15.2.9.2 EDMA3 Interrupt Servicing

On completion of a transfer (early or normal completion), the EDMA3 channel controller sets the appropriate bit in the interrupt pending register (IPR) as specified by the transfer completion codes. If the completion interrupts are appropriately enabled, then the CPU enters the interrupt service routine (ISR) when the completion interrupt is asserted. Since there is a single completion interrupt for all DMA/QDMA channels.

After servicing the interrupt, the ISR should clear the corresponding bit in IPR; therefore, enabling recognition of future interrupts. Only when all IPR bits are cleared, the EDMA3CC will assert additional completion interrupts.

It is possible that when one interrupt is serviced; many other transfer completions result in additional bits being set in IPR, thereby resulting in additional interrupts. It is likely that each of these bits in IPR would need different types of service; therefore, the ISR must check all pending interrupts and continue until all the posted interrupts are appropriately serviced.

Following are examples (pseudo code) for a CPU interrupt service routine for an EDMA3CC completion interrupt.

The ISR routine in [Example 15-2](#) is more exhaustive and incurs a higher latency.

#### **Example 15-2. Interrupt Servicing**

The pseudo code:

1. Read the interrupt pending register (IPR).
2. Perform the operations needed.
3. Write to the interrupt pending clear register (ICR) to clear the corresponding IPR bit.
4. Read IPR again:
  - (a) If IPR is not equal to 0, repeat from step 2 (implies occurrence of new event between step 2 to step 4).
  - (b) If IPR is equal to 0, this should assure you that all enabled interrupts are inactive.

---

**NOTE:** It is possible that during step 4, an event occurs while the IPR bits are read to be 0 and the application is still in the interrupt service routine. If this happens, a new interrupt is recorded in the device interrupt controller and a new interrupt is generated as soon as the application exits the interrupt service routine.

---

**Example 15-3** is less rigorous, with less burden on the software in polling for set interrupt bits, but can occasionally cause a race condition, as mentioned above.

### **Example 15-3. Interrupt Servicing**

If it is desired to leave any enabled and pending (possibly lower priority) interrupts, it is required to force the interrupt logic to reassert the interrupt pulse by setting the EVAL bit in the interrupt evaluation register (IEVAL).

The pseudo code:

1. Enter ISR.
2. Read IPR.
3. For the condition set in IPR that you desire to service:
  - (a) Service interrupt as required by application.
  - (b) Clear bit for serviced conditions (others may still be set, and other transfers may have resulted in returning the TCC to EDMA3CC after step 2).
4. Read IPR prior to exiting ISR:
  - (a) If IPR is equal to 0, then exit ISR.
  - (b) If IPR is not equal to 0, then set IEVAL so that upon exit of ISR, a new interrupt is triggered if any enabled interrupts are still pending.

The EVAL bit must not be set when IPR is read to be 0, to avoid generation of extra interrupt pulses.

---

**NOTE:** Since the DMA region access registers (DRAE) are required to enable the transfer completion region interrupts, it is assumed that there will be a unique and nonoverlapping (in most cases) assignment of the channels and interrupts among the different shadow regions. This allows the interrupt registers (IER, IESR, IECR, IPR, and ICR) in the different shadow regions to functionally operate in an independent manner and nonoverlapping. The above examples for the interrupt service routine is based on this assumption.

---

### **15.2.9.3 Interrupt Evaluation Operations**

The EDMA3CC has interrupt evaluate registers (IEVAL) in each shadow region. These registers are the only registers in the DMA channel shadow region memory map that are not affected by the settings for the DMA region access enable registers (DRAE). A write of 1 to the EVAL bit in these registers associated with a particular shadow region results in pulsing the associated region interrupt, if any enabled interrupt (via IER) is still pending (IPR). This register can be used in order to assure that the interrupts are not missed by the CPU (or the EDMA3 master associated with the shadow region) if the software architecture chooses not to use all interrupts. See [Example 15-3](#) for the use of IEVAL in the EDMA3 interrupt service routine (ISR).

Similarly an error evaluate register (EEVAL) exists in the global region. A write of 1 to the EVAL bit in EEVAL causes the pulsing of the error interrupt if any pending errors are in EMR, QEMR, or CCERR. See [Section 15.2.9.4](#) for additional details on error interrupts.

---

**NOTE:** While using IEVAL for shadow region completion interrupts, you should make sure that the IEVAL operated upon is from that particular shadow region memory map.

---

### 15.2.9.4 Error Interrupts

The EDMA3CC error registers provide the capability to differentiate error conditions (event missed, threshold exceed, etc.). Additionally, if the error bits are set in these registers, it results in asserting the EDMA3CC error interrupt. If EDMA3CC error interrupt is enabled in the device interrupt controller, then it allows the CPU to handle the error conditions.

The EDMA3CC has a single error interrupt (EDMA3\_m\_CC0\_ERRINT) that gets asserted for all EDMA3CC error conditions. There are four conditions that cause the error interrupt to be pulsed:

- DMA missed events: for all 32 DMA channels. These get latched in the event missed registers (EMR).
- QDMA missed events: for all QDMA channels. These get latched in the QDMA event missed register (QEMR).
- Threshold exceed: for all event queues. These get latched in EDMA3CC error register (CCERR).
- TCC error: for outstanding transfer requests expected to return completion code (TCCHEN or TCINTEN bit in OPT is set to 1) exceeding the maximum limit of 31. This also gets latched in the EDMA3CC error register (CCERR).

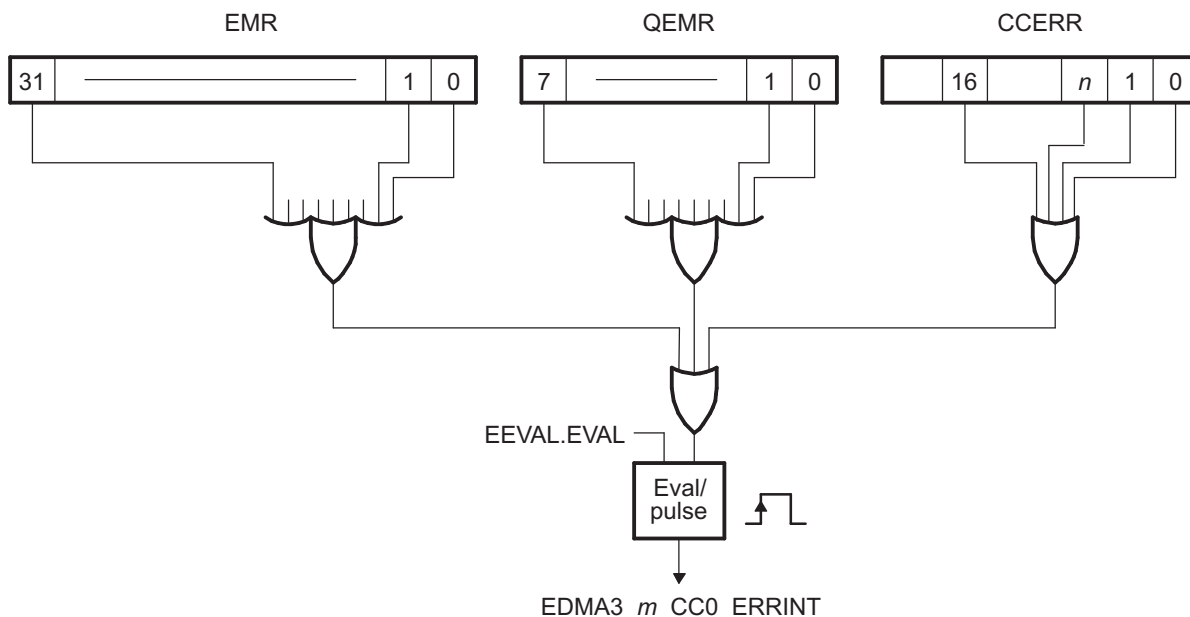
Figure 15-13 illustrates the EDMA3CC error interrupt generation operation.

If any of the bits are set in the error registers due to any error condition, the (EDMA3\_m\_CC0\_ERRINT) always is asserted, as there are no enables for masking these error events. Similar to the transfer completion interrupts, the error interrupt also is pulsed only when the error interrupt condition transitions from a state where no errors are set to a state where at least one error bit is set. If additional error events are latched prior to the original error bits being cleared, the EDMA3CC does not generate additional interrupt pulses.

To reduce the burden on the software, similar to the interrupt evaluate register (IEVAL), there is an error evaluate register (EEVAL) that allows reevaluation of pending set error events/bits. This can be used so that the CPU(s) does not miss any error events.

**NOTE:** It is a good practice to have the error interrupt enabled in the device interrupt controller and associate an interrupt service routine with it to address the various error conditions appropriately. This puts less burden on software (polling for error status) and additionally provides a good debug mechanism for unexpected error conditions.

**Figure 15-13. Error Interrupt Operation**



Note: *n* is the number of queues supported in the EDMA3CC for a specific device.

### 15.2.10 Event Queue(s)

Event queues are a part of the EDMA3 channel controller. Event queues form the interface between the event detection logic in the EDMA3CC and the transfer request (TR) submission logic of the EDMA3CC. Each queue is 16 entries deep, that is, a maximum of 16 queued events per event queue. If there are more than 16 events, then the events that cannot find a place in the event queue remain set in the associated event register.

The number of event queues in the EDMA3CC determines the number of transfer controllers connected to the EDMA3CC. By default, there is a one-to-one mapping between the queues and transfer controllers. Therefore, the transfer requests (TRs) associated with events in Q0 get submitted to TC0. Similarly, transfer requests associated with events in Q1 get submitted to TC1, and so on.

An event that wins prioritization against other DMA and/or QDMA pending events is placed at the end of the appropriate event queue. Each event queue is serviced in a FIFO (first in–first out) order. Once the event reaches the head of its queue and the corresponding transfer controller is ready to receive another TR, the event is dequeued and the PaRAM set corresponding to the dequeued event is processed and submitted as a transfer request packet (TRP) to the associated EDMA3 transfer controller.

A lower numbered queue has a higher dequeuing priority than a higher numbered queue. For example, Q0 has higher priority than Q1, if Q0 and Q1 both have at least one event entry and if both TC0 and TC1 can accept transfer requests, then the event in Q0 is dequeued first and its associated PaRAM set is processed and submitted as a transfer request (TR) to TC0.

All the event entries in all the event queues are software readable (not writeable) by accessing the event queue entry registers (QxEy). Each event entry register characterizes the queued event in terms of the type of event (manual, event, chained or autotriggered) and the event number. See [Section 15.4.2.4.1](#) for a description of the bit fields in the queue event entry registers.

#### 15.2.10.1 DMA/QDMA Channel to Event Queue Mapping

Each DMA channel and QDMA channel is independently programmed to map to a specific queue using the DMA queue number register  $n$  (DMAQNUM $n$ ) and the QDMA channel queue number register (QDMANUM). The mapping of DMA/QDMA channels is critical to achieving the desired performance level for the EDMA and most importantly in meeting real-time deadlines.

---

**NOTE:** If an event is ready to be queued and both the event queue and the EDMA3 transfer controller associated to the event queue are empty, then the event bypasses the event queue, and goes to the PaRAM processing logic and eventually to the transfer request submission logic for submission to the EDMA3TC. In this case, the event is not logged in the event queue status registers.

---



### 15.2.10.2 Queue RAM Debug Visibility

Each event queue has 16 entries. These 16 entries are managed in a circular FIFO manner. All event queue entries for all event queues are software readable by the event queue entry register (QxEx). Additionally, for each queue there is a queue status register (QSTAT $n$ ).

These registers provide user visibility and may be helpful while debugging real-time issues (typically post-mortem), involving multiple events and event sources. The event queue entry register (QxEx) uniquely identifies the specific event type (event-triggered, manually-triggered, chain-triggered, and QDMA events) along with the event number (for DMA/QDMA channels) that are in the queue or have been de-queued (passed through the queue). QSTAT $n$  includes fields for the start pointer (STRTPTR) that provides the offset to the head entry of an event. It also includes a NUMVAL field that provides the total number of valid entries residing in the event queue at a given instance of time. The STRTPTR field may be used to index appropriately into the 16 event entries. The NUMVAL number of entries starting from STRTPTR are indicative of events still queued in the respective queue. The remaining entries may be read to determine which events have already been de-queued and submitted to the associated transfer controller.

### 15.2.10.3 Queue Resource Tracking

The EDMA3CC event queue includes watermarking/threshold logic that allows you to keep track of maximum usage of all event queues. This is useful for debugging real-time deadline violations that may result from head-of-line blocking on a given EDMA3 event queue.

You can program the maximum number of events that can queue up in an event queue by programming the threshold value (between 0 to 15) in the queue watermark threshold A register (QWMTHRA). The maximum queue usage is recorded actively in the watermark (WM) field of the queue status register (QSTAT $n$ ) that keeps getting updated based on a comparison of number of valid entries, which is also visible in the NUMVAL bit in QSTAT $n$  and the maximum number of entries (WM bit in QSTAT $n$ ).

If the queue usage is exceeded, this status is visible in the EDMA3CC registers: the QTHRXCDC $n$  bit in the channel controller error register (CCERR) and the THRXCDC bit in QSTAT $n$ , where  $n$  stands for the event queue number. Any bits that are set in CCERR also generate an EDMA3CC error interrupt.

## 15.2.11 EDMA3 Transfer Controller (EDMA3TC)

The EDMA3 channel controller is the user-interface of the EDMA3 and the EDMA3 transfer controller (EDMA3TC) is the data movement engine of the EDMA3. The EDMA3CC submits transfer requests (TR) to the EDMA3TC and the EDMA3TC performs the data transfers dictated by the TR.

### 15.2.11.1 Architecture Details

#### 15.2.11.1.1 EDMA3TC Configuration

Each transfer controller on a device is designed differently based on considerations like performance requirements, system topology (main SCR bus width, external memory bus width), gate count, etc. The parameters that determine the TC configurations are:

- **FIFOSIZE:** Determines the size in bytes for the Data FIFO that is the temporary buffer for the in-flight data. The data FIFO is where the read return data read by the TC read controller from the source endpoint is stored and subsequently written out to the destination endpoint by the TC write controller.
- **Default Burst Size (DBS):** The DBS is the maximum number of bytes per read/write command issued by a transfer controller.
- **BUSWIDTH:** The width of the read and write data buses in bytes, for the TC read and write controller, respectively. This is typically equal to the bus width of the main SCR interface.
- **DSTREGDEPTH:** This determines the number of Destination FIFO register set. The number of Destination FIFO register set for a transfer controller, determines the maximum number of outstanding transfer requests (TR pipelining).

Of the four parameters, the FIFOSIZE, BUSWIDTH, and DSTREGDEPTH values are fixed in design for a given device. The default burst size (DBS) for EDMA3\_0\_TC0 and EDMA3\_0\_TC1 is configurable by the chip configuration 0 register (CFGCHIP0) in the System Configuration Module and for EDMA3\_1\_TC0 is configurable by the chip configuration 1 register (CFGCHIP1) in the System Configuration Module. [Table 15-12](#) provides the configuration of the individual EDMA3 transfer controllers on the device.

The burst size for each transfer controlled can be programmed to be 16-, 32-, or 64-bytes. The default values for DBS are typically chosen for optimal performance in most intended-use conditions; therefore, if you decide to use a value other than the default, you should evaluate the impact on performance. Depending on the FIFOSIZE and source/destination locations the performance for the transfer can vary significantly for different burst size values.

---

**NOTE:** It is expected that the DBS value for a transfer controller is static and should be based on the application requirement. It is not recommended that the DBS value be changed on-the-fly.

---

**Table 15-12. EDMA3 Transfer Controller Configurations**

Parameter	EDMA3_0_TC0	EDMA3_0_TC1	EDMA3_1_TC0
FIFOSIZE	128 bytes	128 bytes	256 bytes
BUSWIDTH	8 bytes (64 bits)	8 bytes (64 bits)	8 bytes (64 bits)
DSTREGDEPTH	4 entries	4 entries	4 entries
DBS (default)	16 bytes	16 bytes	16 bytes
Error interrupt	EDMA3_0_TC0_ERRINT	EDMA3_0_TC1_ERRINT	EDMA3_1_TC0_ERRINT
EDMA3 channel controller used	EDMA3_0_CC0	EDMA3_0_CC0	EDMA3_1_CC0

### 15.2.11.1.2 Command Fragmentation

The TC read and write controllers in conjunction with the source and destination register sets are responsible for issuing optimally-sized reads and writes to the slave endpoints. The transfer controller read/write transaction as specified by the transfer request packet is internally broken down into smaller bursts; this determines the default burst size (DBS) for the transfer controller. See [Section 15.2.11.1.1](#) for the DBS value of each EDMA3TC.

The EDMA3TC attempts to issue the largest possible command size as limited by the DBS value or the ACNT/BCNT value of the TR. EDMA3TC obeys the following rules:

- The read/write controllers always issue commands less than or equal to the DBS value.
- The first command of a 1D transfer is always issued so that subsequent commands align to the DBS value.

[Example 15-4](#) shows the command fragmentation for a DBS of 32 bytes. In summary, if the ACNT value is larger than the DBS value, then the EDMA3TC breaks the ACNT array into DBS-sized commands to the source/destination addresses. Each BCNT number of arrays are then serviced in succession.

#### Example 15-4. Command Fragmentation (DBS = 32)

The pseudo code:

1. ACNT = 8, BCNT = 8, SRCBIDX = 8, DSTBIDX = 10, SRCADDR = 64, DSTADDR = 191

Read Controller: This is optimized from a 2D-transfer to a 1D-transfer such that the read side is equivalent to ACNT = 64, BCNT = 1.

Cmd0 = 32 byte, Cmd0 = 32 byte

Write Controller: Since DSTBIDX != ACNT, it is not optimized.

Cmd0 = 8 byte, Cmd1 = 8 byte, Cmd2 = 8 byte, Cmd3 = 8 byte, Cmd4 = 8 byte, Cmd5 = 8 byte, Cmd6 = 8 byte, Cmd7 = 8 byte.

2. ACNT = 64, BCNT = 1, SRCADDR = 31, DSTADDR = 513

Read Controller: Read address is not aligned.

Cmd0 = 1 byte, (now the SRCADDR is aligned to 32 for the next command)

Cmd1 = 32 bytes

Cmd2 = 31 bytes

Write Controller: The write address is also not aligned.

Cmd0 = 31 bytes, (now the DSTADDR is aligned to 32 for the next command)

Cmd1 = 32 bytes

Cmd2 = 1 byte

### 15.2.11.1.3 TR Pipelining and Data Ordering

The transfer controller(s) can issue back-to-back transfer requests (TR). The number of outstanding TRs for a TC is limited by the number of destination FIFO register entries that is controlled by the DSTREGDEPTH parameter (fixed in design for a given transfer controller). TR pipelining refers to the ability of the TC read controller to issue read commands for a subsequent TR, while the TC write controller is still performing writes for the previous TR. Consider the case of 2 TRs (TR0 followed by TR1), because of TR pipelining, the TC read controller can start issuing the read commands for TR1 as soon as the last read command for TR0 has been issued, meanwhile the write commands and write data for TR0 are tracked by the destination FIFO registers. In summary, the TC read controller is able to process  $n$  TRs ahead of the write controller, where  $n$  is the number of destination FIFO register entries (typically 4).

TR pipelining is useful for maintaining throughput on back-to-back small TRs. It eliminates the read overhead because reads start in the background of a previous TR writes.

It should be noted that back-to-back TRs are targeted to different end points even though the read return data for the two TRs might get returned out of order (that is, read data for TR1 might come in before read data for TR0), the transfer controller issues that the write commands are issued in order (that is, write commands for TR0 will be issued before write commands for TR1).

### 15.2.11.2 Error Generation

Similar to the channel controller, the transfer controllers are capable of detecting and reporting several error conditions. The TC errors are generated, under three main conditions:

- **BUSERR:** The TC read or write controllers detect an error signaled by the source or destination address. The additional details on the type of error is also recorded in the ERRDET register, which indicates whether it is a read error (source address errors) or write error (destination address error).
- **MMRAERR:** CPU accesses illegal/reserved addresses in the EDMA3CC/TC memory-map.
- **TRERR:** A transfer request packet is detected to be violating the constant addressing mode transfer rules (the source/destination addresses and source/destination indexes must be aligned to 32 bytes).

You can poll for the errors, as the status of the errors can be read from the ERRSTAT registers, additionally if the error bits are enabled in the ERREN register, a bit set in the ERRSTAT will cause the error condition to interrupt the CPU(s). You can decide to enable/disable either or all error types.

### 15.2.11.3 Debug Features

The DMA program register set, DMA source active register set, and the destination FIFO register set are used to derive a brief history of TRs serviced through the transfer controller.

Additionally, the EDMA3TC status register (TCSTAT) has dedicated bit fields to indicate the ongoing activity within different parts of the transfer controller:

- The SRCACTV bit indicates whether the source active set is active.
- The DSTACTV bit indicates the number of TRs resident in the destination register active set at a given instance.
- The PROGBUSY bit indicates whether a valid TR is present in the DMA program set.

If the TRs are in progression, caution must be used and you must realize that there is a chance that the values read from the EDMA3TC status registers will be inconsistent since the EDMA3TC may change the values of these registers due to ongoing activities.

It is recommended that you ensure no additional submission of TRs to the EDMA3TC in order to facilitate ease of debug.

#### 15.2.11.3.1 Destination FIFO Register Pointer

The destination FIFO register pointer is implemented as a circular buffer with the start pointer being DFSTRTPTR and a buffer depth of usually 2 or 4. The EDMA3TC maintains two important status details in TCSTAT that may be used during advanced debugging, if necessary. The DFSTRTPTR is a start pointer, that is, the index to the head of the destination FIFO register. The DSTACTV is a counter for the number of valid (occupied) entries. These registers may be used to get a brief history of transfers.

Examples of some register field values and their interpretation:

- DFSTRTPTR = 0 and DSTACTV = 0 implies that no TRs are stored in the destination FIFO register.
- DFSTRTPTR = 1 and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 1 and the second pending TR is read from the destination FIFO register entry 2.
- DFSTRTPTR = 3h and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 3 and the second pending TR is read from the destination FIFO register entry 0.

### 15.2.12 Event Dataflow

This section summarizes the data flow of a single event, from the time the event is latched to the channel controller to the time the transfer completion code is returned. The following steps list the sequence of EDMA3CC activity:

1. Event is asserted from an external source (peripheral or external interrupt). This also is similar for a manually-triggered, chained-triggered, or QDMA-triggered event. The event is latched into the *ER.En* (or *CER.En*, *ESR.En*, *QER.En*) bit.
2. Once an event is prioritized and queued into the appropriate event queue, the *SER.En* (or *QSER.En*) bit is set to inform the event prioritization/processing logic to disregard this event since it is already in the queue. Alternatively, if the transfer controller and the event queue are empty, then the event bypasses the queue.
3. The EDMA3CC processing and the submission logic evaluates the appropriate PaRAM set and determines whether it is a non-null and non-dummy transfer request (TR).
4. The EDMA3CC clears the *ER.En* (or *CER.En*, *ESR.En*, *QER.En*) bit and the *SER.En* bit as soon as it determines the TR is non-null. In the case of a null set, the *SER.En* bit remains set. It submits the non-null/non-dummy TR to the associated transfer controller. If the TR was programmed for early completion, the EDMA3CC immediately sets the interrupt pending register (*IPR.I[TCC]*).
5. If the TR was programmed for normal completion, the EDMA3CC sets the interrupt pending register (*IPR.I[TCC]*) when the EDMA3TC informs the EDMA3CC about completion of the transfer (returns transfer completion codes).
6. The EDMA3CC programs the associated EDMA3TC $n$  Program Register Set with the TR.
7. The TR is then passed to the Source Active set and the Dst FIFO Register Set, if both the register sets are available.
8. The Read Controller processes the TR by issuing read commands to the source slave endpoint. The Read Data lands in the Data FIFO of the EDMA3TC $n$ .
9. As soon as sufficient data is available, the Write Controller begins processing the TR by issuing write commands to the destination slave endpoint.
10. This continues until the TR completes and on receiving the acknowledgement signal from the destination slave end point, the EDMA3TC $n$  then signals completion status to the EDMA3CC.

### 15.2.13 EDMA3 Prioritization

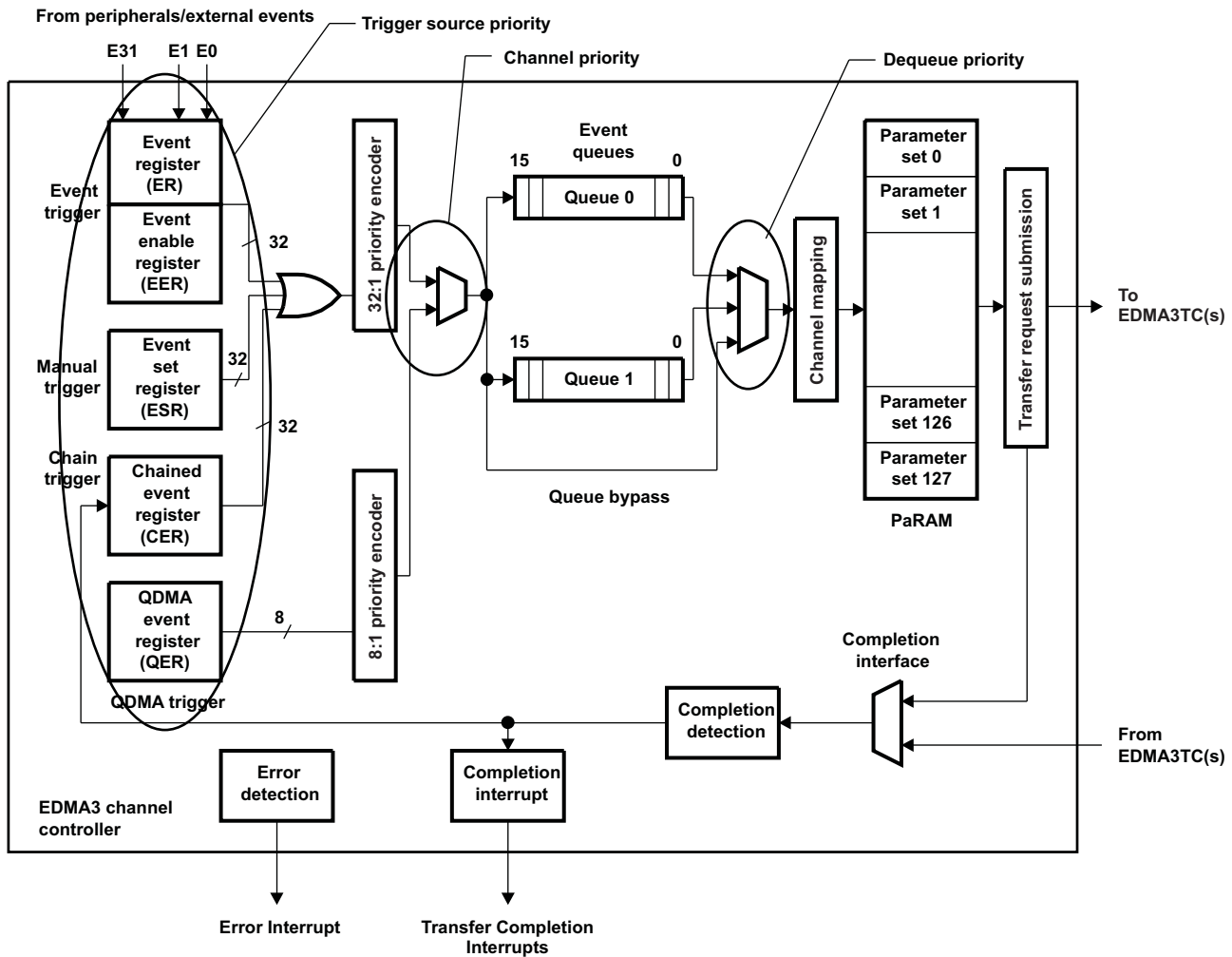
The EDMA3 controller has many implementation rules to deal with concurrent events/channels, transfers, etc. The following subsections detail various arbitration details whenever there might be occurrence of concurrent activity. Figure 15-14 shows the different places EDMA3 priorities come into play.

#### 15.2.13.1 Channel Priority

The DMA event register (ER) captures all external/peripheral events connected to the EDMA3CC; likewise, the QDMA event register (QER) captures QDMA events for all QDMA channels; therefore, it is possible for events to occur simultaneously on the DMA/QDMA event inputs. For events arriving simultaneously, the event associated with the lowest channel number is prioritized for submission to the event queues (for DMA events, channel 0 has the highest priority and channel 31 has the lowest priority; similarly, for QDMA events, channel 0 has the highest priority and channel 7 has the lowest priority). This mechanism only sorts simultaneous events for submission to the event queues.

If a DMA and QDMA event occurs simultaneously, the DMA event always has prioritization against the QDMA event for submission to the event queues.

Figure 15-14. EDMA3 Prioritization



### 15.2.13.2 Trigger Source Priority

If a DMA channel is associated with more than one trigger source (event trigger, manual trigger, and chain trigger), and if multiple events are set simultaneously for the same channel ( $ER.En = 1$ ,  $ESR.En = 1$ ,  $CER.En = 1$ ), then the EDMA3CC always services these events in the following priority order: event trigger (via ER) is higher priority than chain trigger (via CER) and chain trigger is higher priority than manual trigger (via ESR).

This implies that if for channel 0, both  $ER.E0 = 1$  and  $CER.E0 = 1$  at the same time, then the  $ER.E0$  event is always queued before the  $CER.E0$  event.

### 15.2.13.3 Dequeue Priority

The priority of the associated transfer request (TR) is further mitigated by which event queue is being used for event submission (dictated by  $DMAQNUMn$  and  $QDMAQNUM$ ). For submission of a TR to the transfer controller, events need to be dequeued from the event queues. A lower numbered queue has a higher dequeuing priority than a higher numbered queue. For example, if there are events in Q0 and Q1 and the respective transfer controllers (TC0 and TC1) are ready to receive the next TR from the EDMA3CC, then the transfer requests associated with events in Q0 will get submitted to TC0 prior to any transfer requests associated with events in Q1 getting submitted to TC1.

---

**NOTE:** At any given time, if there are outstanding events in multiple queues, when the transfer controller associated with the lower numbered (higher priority) queue is busy processing earlier transfer requests and the transfer controller associated with the higher numbered (lower priority) queue is idle, then the event in the higher numbered (lower priority) queue will dequeue first.

---

### 15.2.13.4 Master (Transfer Controller) Priority

All master peripherals on the device have a programmable priority level. When multiple masters are trying to access common shared resources (slave memory or peripherals), this priority value allows the system interconnect to arbitrate requests from different masters based on their priority. This priority assignment is determined in the Master Priority Registers (MSTPRI0-MSTPRI2) in the System Configuration Module (see the *System Configuration (SYSCFG) Module* chapter), where each master has an allocated priority value (power on reset default value), which can be re programmed based on the applications prioritization requirements. The priority value can be configured between 0 to 7, with 0 being the highest priority and 7 being the lowest priority.

Each transfer controller on the device is also a master peripheral. The priority of the transfer requests (read/write commands) issued by the individual EDMA3TC read/write ports in the system can be programmed via these registers.

The dequeue priority has a relatively secondary effect as compared to this Master priority; therefore, it is important to program the priority of each transfer controller with respect to each other and also with respect to other masters in the system.

---

**NOTE:** On previous architectures, the EDMA3TC priority was controlled by the QUEPRI register in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module.

---



## 15.2.14 EDMA3CC and EDMA3TC Performance and System Considerations

### 15.2.14.1 System Priority Considerations

The main switched central resource (SCR) (see your device-specific data manual) arbitrates bus requests from all the masters (CPU, master peripherals, and the EDMA3 transfer controllers) to the shared slave resources (peripherals and memories). The priorities of transfer requests (read and write commands) from the EDMA3 transfer controllers with respect to each other and the other masters within the system is configured as explained in [Section 15.2.13.4](#).

It is recommended that this priority be altered based on system level considerations. For example, peripherals servicing audio/video/display threads that typically have real-time deadlines should be programmed as highest priority requestors in the systems, where as, peripherals responsible for doing bulk/block/paging transfers with no real-time deadlines, should be programmed as a lower system priority.

The default priority for all transfer controllers is the same, 0 or highest priority relative to other masters; therefore, it is recommended that a TC servicing audio data requests from serial ports should be configured at a higher priority as compared to TC service memory to memory (paging/bulk) transfer requests.

### 15.2.14.2 TC Transfer Optimization Considerations

The transfer controller can internally optimize the way it issues read commands and write commands for a given transfer under certain conditions. For 2D transfers (that is, BCNT arrays of ACNT bytes), if the ACNT value is less than or equal to the DBS value, then the transfer controller will try to optimize the TR into a 1D transfer in order to maximize efficiency. The optimization only takes place if the EDMA3TC recognizes that the 2D transfer is organized as a single dimension (SAM/DAM = 0, increment mode), SRC/DST BIDX = ACNT, the ACNT value is a power of 2, and the BCNT value is less than or equal to 1023. If these conditions are met, then instead of issuing ACNT bytes worth read and/or write commands, the TC will try to optimize the bus usage by issuing commands as if  $ACNT' = ACNT \times BCNT$  and  $BCNT = 1$ .

[Table 15-13](#) summarizes the conditions in which the optimizations are performed.

**Table 15-13. Read/Write Command Optimization Rules**

ACNT ≤ DBS	ACNT is power of 2	BIDX = ACNT	BCNT ≤ 1023	SAM/DAM = 0 (Increment)	Description
Yes	Yes	Yes	Yes	Yes	Optimized
Yes	No	x	x	Yes	Not Optimized
Yes	x	No	x	Yes	Not Optimized
No	x	x	x	Yes	Not Optimized
x	x	x	x	No	Not Optimized

Consider a case in which it is needed to transfer 4096 bytes where the data is arranged linearly in both the source and destination locations (SAM/DAM = 0, SRC/DST BIDX = ACNT): Scenario A programs the ACNT = 4, BCNT = 1024, AB-synchronized transfer; and Scenario B programs the ACNT = 64, BCNT = 64. Scenario B will yield a much optimized transfer and higher throughput, as the transfer meets all the optimization rules, which would result in TC internally treating it as a transfer with an  $ACNT' = 4096$  ( $ACNT \times BCNT$ ). The TC will optimally size, default burst size worth read and write commands. In the case of Scenario B, since one of the optimization rules is not met (BCNT value is greater than 1023), the TC will end up issuing several ACNT byte (4 byte) size commands to complete the transfers, which will result in inefficient usage of the read/write buses.



### 15.2.14.3 Throttling the Read Command Rate in a Transfer Controller

By default, the transfer controller issues reads as fast as possible. In some cases, the reads issued by the EDMA3TCC could fill the available command buffering for a slave, delaying other (potentially higher priority) masters from successfully submitting commands to that slave. The rate at which read commands are issued by the EDMA3TC is controlled by the read command rate register (RDRATE), and this can be used to throttle the rate at which the commands are issued from the TC read interface. RDRATE defines the number of cycles that the EDMA3TC read controller waits before issuing subsequent commands for a given TR, thus minimizing the chance of the EDMA3TC consuming all available slave resources. The RDRATE value should be set to a relatively small value (or kept at default, which implies issuing read requests as fast as possible) if the transfer controller is targeted for high-priority transfers and set to a high value if the transfer controller is targeted for low-priority transfers. In contrast, the write Interface does not have any performance turning knobs because writes always have an interval between commands as write commands are submitted along with the associated write data.

### 15.2.15 EDMA3 Operating Frequency (Clock Control)

The EDMA3 channel controller and transfer controller are clocked from PLL controller 0 (PLL0). For details, see the *Phase-Locked Loop Controller (PLL)* chapter.

### 15.2.16 Reset Considerations

A hardware reset resets the EDMA3 (EDMA3CC and EDMA3TC) and the EDMA3 configuration registers. The PaRAM memory contents are undefined after device reset and you should not rely on parameters to be reset to a known state. The PaRAM set must be initialized to a desired value before it is used.

### 15.2.17 Power Management

The EDMA3 (EDMA3CC and EDMA3TC) can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the device Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

The EDMA3 controller can be idled on receiving a clock stop request from the PSC. The requests to EDMA3CC and EDMA3TC are separate. In general, you should verify that there are no pending activities in the EDMA3 controller before issuing a clock stop request via PSC.

The EDMA3CC checks for the following conditions:

- No pending DMA/QDMA events
- No outstanding events in the event queues
- Transfer request processing logic is not active
- No completion requests outstanding (early or normal completion)
- No configuration bus requests in progress

The first four conditions are software readable by the channel controller status register (CCSTAT) in the EDMA3CC.

Similarly, from the EDMA3TC perspective, you should check that there are no outstanding TRs that are getting processed and essentially the read/write controller is not busy processing a TR. The activity of EDMA3TC logic is read in TCSTAT for each EDMA3TC.

It is generally recommended to first disable the EDMA3CC and then the EDMA3TC(s) to put the EDMA3 controller in reduced-power modes.

Additionally, when EDMA3 is involved in servicing a peripheral and it is required to power-down both the peripheral and the EDMA, the recommended sequence is to first disable the peripheral, then disable the DMA channel associated with the peripheral (clearing the EER bit for the channel), then disable the EDMA3CC, and finally disable the EDMA3TC(s).

### 15.2.18 Emulation Considerations

During debug when using the emulator, the CPU(s) may be halted on an execute packet boundary for single-stepping, benchmarking, profiling, or other debug purposes. During an emulation halt, the EDMA3 channel controller and transfer controller operations continue. Events continue to be latched and processed and transfer requests continue to be submitted and serviced.

Since EDMA3 is involved in servicing multiple master and slave peripherals, it is not feasible to have an independent behavior of the EDMA3 for emulation halts. EDMA3 functionality would be coupled with the peripherals it is servicing, which might have different behavior during emulation halts. For example, if a multichannel buffered serial port (McBSP) is halted during an emulation access (FREE = 0 and SOFT = 0 or 1 in the McBSP registers), the McBSP stops generating the McBSP receive or transmit events (REVT or XEVT) to the EDMA. From the point of view of the McBSP, the EDMA3 is suspended, but other peripherals (for example, a timer) still assert events and will be serviced by the EDMA.

## 15.3 Transfer Examples

The EDMA3 channel controller performs a variety of transfers depending on the parameter configuration. The following sections provides a description and PaRAM configuration for some typical use case scenarios.

### 15.3.1 Block Move Example

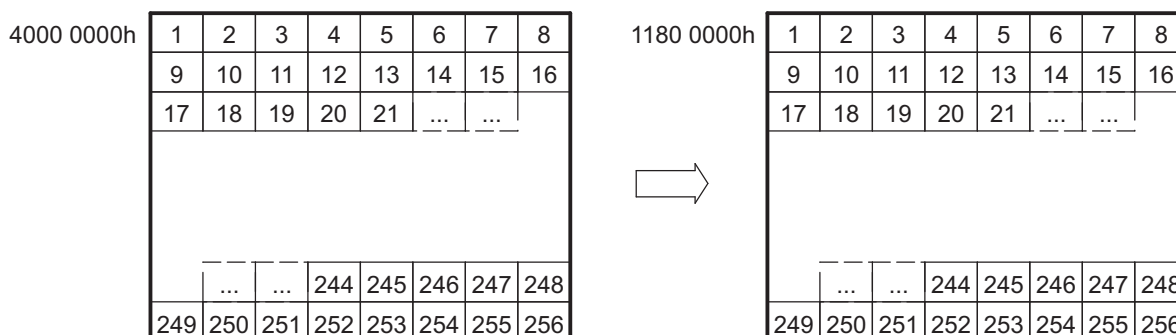
The most basic transfer performed by the EDMA3 is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, a section of data is to be copied from external memory to internal L2 SRAM. A data block of 256 bytes residing at address 4000 0000h (external memory ) needs to be transferred to internal address 1180 0000h (L2), as shown in Figure 15-15. Figure 15-16 shows the parameters for this transfer.

The source address for the transfer is set to the start of the data block in external memory, and the destination address is set to the start of the data block in L2. If the data block is less than 64K bytes, the PaRAM configuration in Figure 15-16 holds true with the synchronization type set to A-synchronized and indexes cleared to 0. If the amount of data is greater than 64K bytes, BCNT and the B-indexes need to be set appropriately with the synchronization type set to AB-synchronized. The STATIC bit in OPT is set to prevent linking.

This transfer example may also be set up using QDMA. For successive transfer submissions, of a similar nature, the number of cycles used to submit the transfer are fewer depending on the number of changing transfer parameters. You may program the QDMA trigger word to be the highest numbered offset in the PaRAM set that undergoes change.

Figure 15-15. Block Move Example



**Figure 15-16. Block Move Example PaRAM Configuration**

## (a) EDMA Parameters

Parameter Contents		Parameter	
0010 0008h		Channel Options Parameter (OPT)	
4000 0000h		Channel Source Address (SRC)	
0001h	0100h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
1180 0000h		Channel Destination Address (DST)	
0000h	0000h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0000h	FFFFh	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

## (b) Channel Options Parameter (OPT) Content

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	0	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0000	0	000	0000	1	0	0	0					
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM					

### 15.3.2 Subframe Extraction Example

The EDMA3 can efficiently extract a small frame of data from a larger frame of data. By performing a 2D-to-1D transfer, the EDMA3 retrieves a portion of data for the CPU to process. In this example, a 640 × 480-pixel frame of video data is stored in external memory, SDRAM. Each pixel is represented by a 16-bit halfword. The CPU extracts a 16 × 12-pixel subframe of the image for processing. To facilitate more efficient processing time by the CPU, the EDMA3 places the subframe in internal L2 SRAM. Figure 15-17 shows the transfer of a subframe from external memory to L2. Figure 15-18 shows the parameters for this transfer.

The same PaPARAM set options are used for QDMA channels, as well as DMA channels. The STATIC bit in OPT is set to 1 to prevent linking. For successive transfers, only changed parameters need to be programmed before triggering the channel.

Figure 15-17. Subframe Extraction Example

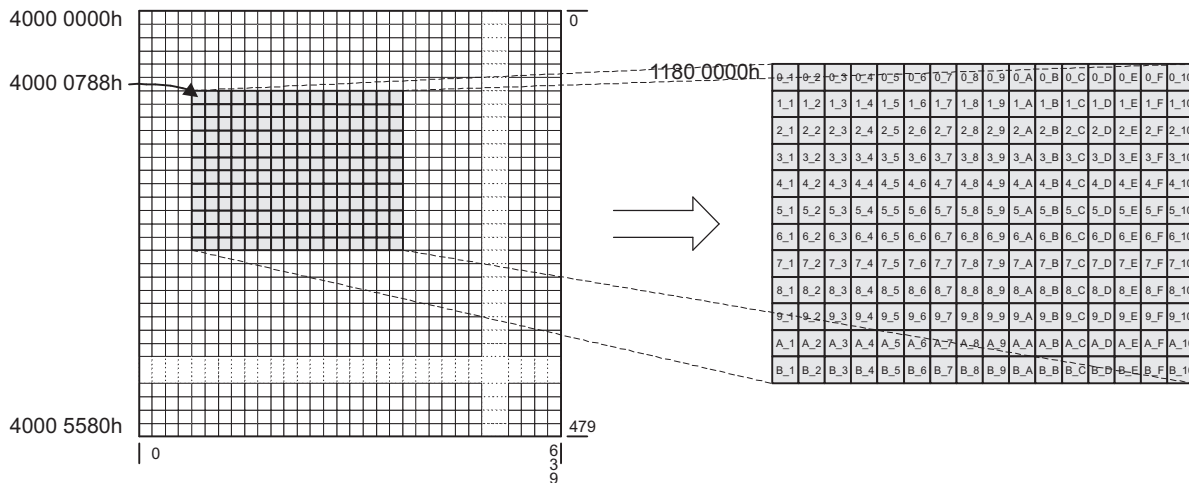


Figure 15-18. Subframe Extraction Example PaPARAM Configuration

(a) EDMA Parameters

Parameter Contents	
0010 000Ch	
4000 0788h	
000Ch	0020h
1180 0000h	
0020h	0500h
0000h	FFFFh
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	0	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0000	0	000	0000	0000	0000	1	1	0	0			
TCC	TCCMOD	FWID	Reserved	Reserved	Reserved	STATIC	SYNCDIM	DAM	SAM			

### 15.3.3 Data Sorting Example

Many applications require the use of multiple data arrays; it is often desirable to have the arrays arranged such that the first elements of each array are adjacent, the second elements are adjacent, and so on. Often this is not how the data is presented to the device. Either data is transferred via a peripheral with the data arrays arriving one after the other or the arrays are located in memory with each array occupying a portion of contiguous memory spaces. For these instances, the EDMA3 can reorganize the data into the desired format. [Figure 15-19](#) shows the data sorting.

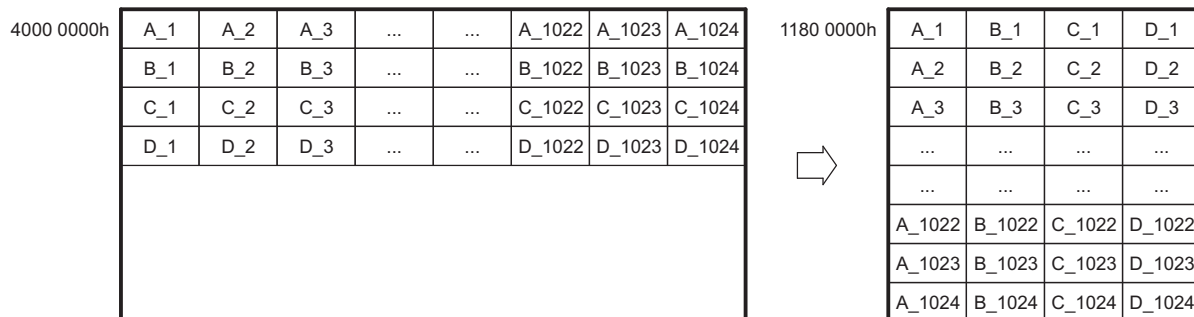
In order to determine the parameter entry values, the following need to be considered:

- ACNT – Program this to be the size in bytes of an array.
- BCNT – Program this to be the number of arrays in a frame.
- CCNT – Program this to be the number of frames.
- SRCBIDX – Program this to be the size of the array or ACNT.
- DSTBIDX = CCNT × ACNT
- SRCCIDX = ACNT × BCNT
- DSTCIDX = ACNT

The synchronization type needs to be AB-synchronized and the STATIC bit is 0 to allow updates to the parameter set. It is advised to use normal DMA channels for sorting.

It is not possible to sort this with a single trigger event. Instead, the channel can be programmed to be chained to itself. After BCNT arrays get sorted, intermediate chaining could be used to trigger the channel again causing the transfer of the next BCNT arrays and so on. [Figure 15-20](#) shows the parameter set programming for this transfer, assuming channel 0 and an array size of 4 bytes.

**Figure 15-19. Data Sorting Example**



**Figure 15-20. Data Sorting Example PaRAM Configuration**

## (a) EDMA Parameters

Parameter Contents	
0090 0004h	
4000 0000h	
0400h	0004h
1180 0000h	
0010h	0004h
0000h	FFFFh
0004h	1000h
0000h	0004h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

## (b) Channel Options Parameter (OPT) Content

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	1	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0000	0	000	0000	0	1	0	0					
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM					

### 15.3.4 Peripheral Servicing Example

**NOTE:** Examples in this section are sample examples. The peripherals, channels, and addresses used in these examples may not apply to your specific device. See your device-specific data manual for supported peripherals.

The EDMA3 channel controller also services peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the DMA channels, they can be configured to continuously service on-chip and off-chip peripherals throughout the device operation. Each event available to the EDMA3 has its own dedicated channel, and all channels operate simultaneously. The only requirements are to use the proper channel for a particular transfer and to enable the channel event in the event enable register (EER). When programming a DMA channel to service a peripheral, it is necessary to know how data is to be presented to the CPU. Data is always provided with some kind of synchronization event as either one element per event (nonbursting) or multiple elements per event (bursting).

#### 15.3.4.1 Nonbursting Peripherals

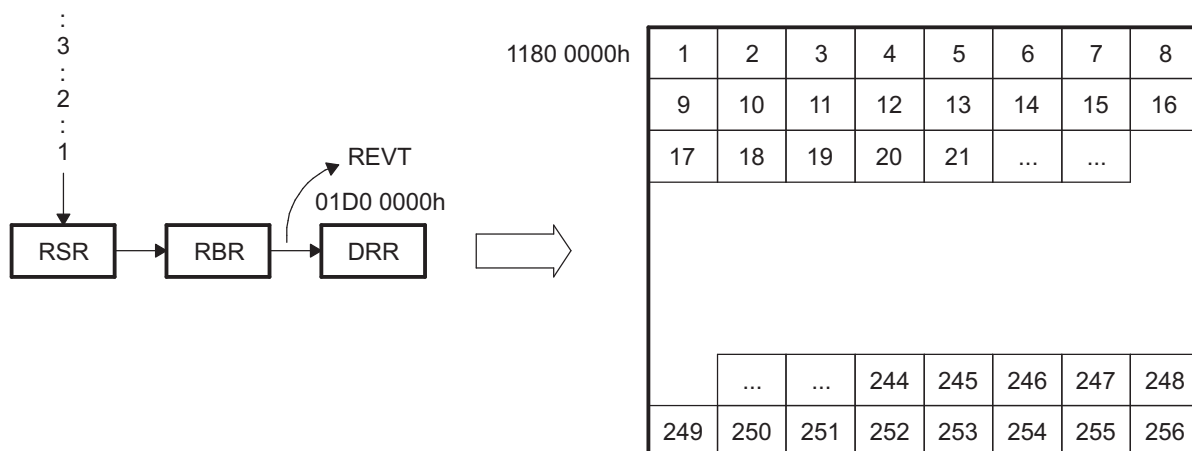
Nonbursting peripherals include the on-chip multichannel buffered serial port (McBSP) and many external devices, such as codecs. Regardless of the peripheral, the DMA channel configuration is the same.

The McBSP transmit and receive data streams are treated independently by the EDMA3. The transmit and receive data streams can have completely different counts, data sizes, and formats. [Figure 15-21](#) shows servicing incoming McBSP data.

To transfer the incoming data stream to its proper location in L2 memory, the DMA channel must be set up for a 1D-to-1D transfer with A-synchronization. Since an event (REVT) is generated for every word as it arrives, it is necessary to have the EDMA3 issue the transfer request for each element individually. [Figure 15-22](#) shows the parameters for this transfer. The source address of the DMA channel is set to the data receive register (DRR) address for the McBSP, and the destination address is set to the start of the data block in L2. Since the address of DRR is fixed, the source B index is cleared to 0 (no modification) and the destination B index is set to 01b (increment).

Based on the premise that serial data is typically a high priority, the DMA channel should be programmed to be on queue 0.

**Figure 15-21. Servicing Incoming McBSP Data Example**



**Figure 15-22. Servicing Incoming McBSP Data Example PaRAM**

## (a) EDMA Parameters

Parameter Contents	
0010 0000h	
01D0 0000h	
0100h	0001h
1180 0000h	
0001h	0000h
0000h	FFFFh
0000h	0000h
0000h	0004h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

## (b) Channel Options Parameter (OPT) Content

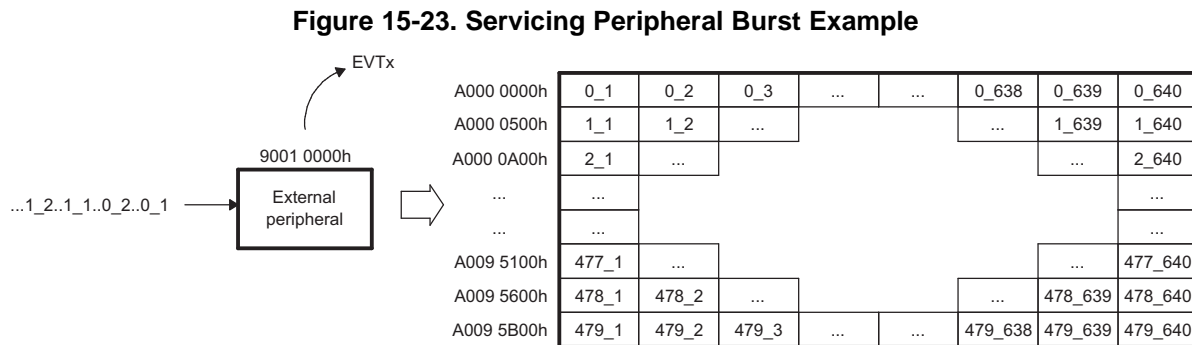
31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	0	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0000	0	000	0000	0	0	0	0					
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM					



### 15.3.4.2 Bursting Peripherals

Higher bandwidth applications require that multiple data elements be presented to the CPU for every synchronization event. This frame of data can either be from multiple sources that are working simultaneously or from a single high-throughput peripheral that streams data to/from the CPU. In this example, a port is receiving a video frame from a camera and presenting it to the CPU one array at a time. The video image is 640 × 480 pixels, with each pixel represented by a 16-bit element. The image is to be stored in external memory. Figure 15-23 shows this example.

To transfer data from an external peripheral to an external buffer one array at a time based on  $EVT_n$ , channel  $n$  must be configured. Due to the nature of the data (a video frame made up of arrays of pixels) the destination is essentially a 2D entity. Figure 15-24 shows the parameters to service the incoming data with a 1D-to-2D transfer using AB-synchronization. The source address is set to the location of the video framer peripheral, and the destination address is set to the start of the data buffer. Since the input address is static, the SRCBIDX is 0 (no modification to the source address). The destination is made up of arrays of contiguous, linear elements; therefore, the DSTBIDX is set to pixel size, 2 bytes. ANCT is equal to the pixel size, 2 bytes. BCNT is set to the number of pixels in an array, 640. CCNT is equal to the total number of arrays in the block, 480. SRCCIDX is 0 since the source address undergoes no increment. The DSTCIDX is equal to the difference between the starting addresses of each array. Since a pixel is 16 bits (2 bytes), DSTCIDX is equal to 640 × 2.



**Figure 15-24. Servicing Peripheral Burst Example PaRAM**

(a) EDMA Parameters

Parameter Contents	
0010 0004h	
Channel Source Address	
0280h	0002h
4000 0000h	
0002h	0000h
0000h	FFFFh
0500h	0000h
0000h	01E0h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	0	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0000	0	000	0000	0000	0000	0	1	0	0			
TCC	TCCMOD	FWID	Reserved	Reserved	Reserved	STATIC	SYNCDIM	DAM	SAM			

### 15.3.4.3 Continuous Operation

Configuring a DMA channel to receive a single frame of data is useful, and is applicable to some systems. A majority of the time, however, data is going to be continuously transmitted and received throughout the entire operation of the CPU. In this case, it is necessary to implement some form of linking such that the DMA channels continuously reload the necessary parameter sets. In this example, the multichannel buffered serial port (McBSP) is configured to transmit and receive data on an array. To simplify the example, only two channels are active for both transmit and receive data streams. Each channel receives packets of 128 elements. The packets are transferred from the serial port to L2 memory and from L2 memory to the serial port, as shown in Figure 15-25.

The McBSP generates REVT for every element received and generates XEVT for every element transmitted. To service the data streams, the DMA channels associated with the McBSP must be set up for 1D-to-1D transfers with A-synchronization.

Figure 15-26 shows the parameters for the parameter entries for the channel for these transfers. In order to service the McBSP continuously throughout CPU operation, the channels must be linked to a duplicate PaRAM set in the PaRAM. After all frames have been transferred, the DMA channels reload and continue. Figure 15-27 shows the reload parameters for the channel.

#### 15.3.4.3.1 Receive Channel

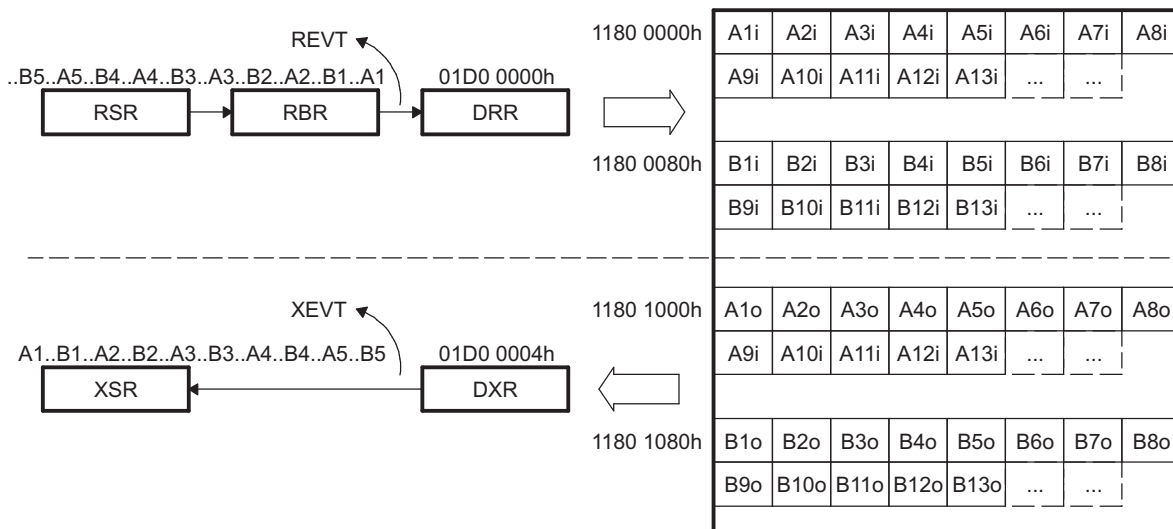
DMA channel 3 services the incoming data stream of the McBSP. The source address is set to that of the data receiver register (DRR), and the destination address is set to the first element of the data block. Since there are two data channels being serviced, A and B, they are to be located separately within the L2 SRAM.

In order to facilitate continuous operation, a copy of the PaRAM set for the channel is placed in PaRAM set 64. The LINK option is set and the link address is provided in the PaRAM set. Upon exhausting the channel 3 parameter set, the parameters located at the link address are loaded into the channel 3 parameter set and operation continues. This function continues throughout device operation until halted by the CPU.

#### 15.3.4.3.2 Transmit Channel

DMA channel 2 services the outgoing data stream of the McBSP. In this case the destination address needs no update, hence, the parameter set changes accordingly. Linking is also used to allow continuous operation by the DMA channel, with duplicate PaRAM set entries at PaRAM set 65.

Figure 15-25. Servicing Continuous McBSP Data Example



**Figure 15-26. Servicing Continuous McBSP Data Example PaRAM**

(a) EDMA Parameters for Receive Channel (PaRAM Set 3) being Linked to PaRAM Set 64

Parameter Contents	
0010 0000h	
01D0 0000h	
0080h	0001h
1180 0000h	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 3)

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000		0	0	0	1	00		00		
PRIV	Reserved		PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved		TCC		
15	12	11	10	8	7	4		3	2	1	0	
0000		0	000	0000			0	0	0	0		
TCC		TCCMOD	FWID	Reserved			STATIC	SYNCDIM	DAM	SAM		

(c) EDMA Parameters for Transmit Channel (PaRAM Set 2) being Linked to PaRAM Set 65

Parameter Contents	
0010 1000h	
1180 1000h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4820h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 2)

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000		0	0	0	1	00		00		
PRIV	Reserved		PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved		TCC		
15	12	11	10	8	7	4		3	2	1	0	
0001		0	000	0000			0	0	0	0		
TCC		TCCMOD	FWID	Reserved			STATIC	SYNCDIM	DAM	SAM		

**Figure 15-27. Servicing Continuous McBSP Data Example Reload PaRAM**

(a) EDMA Reload Parameters (PaRAM Set 64) for Receive Channel

Parameter Contents	
0010 0000h	
01D0 0000h	
0080h	0001h
1180 0000h	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 64)

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	0	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0000	0	000	0000	0	0	0	0					
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM					

(c) EDMA Reload Parameters (PaRAM Set 65) for Transmit Channel

Parameter Contents	
0010 1000h	
1180 1000h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4820h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 65)

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000	0	0	0	1	00	00				
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC				
15	12	11	10	8	7	4	3	2	1	0		
0001	0	000	0000	0	0	0	0					
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM					

#### 15.3.4.4 Ping-Pong Buffering

Although the previous configuration allows the EDMA3 to service a peripheral continuously, it presents a number of restrictions to the CPU. Since the input and output buffers are continuously being filled/emptied, the CPU must match the pace of the EDMA3 very closely in order to process the data. The EDMA3 receive data must always be placed in memory before the CPU accesses it, and the CPU must provide the output data before the EDMA3 transfers it. Though not impossible, this is an unnecessary challenge. It is particularly difficult in a 2-level cache scheme.

Ping-pong buffering is a simple technique that allows the CPU activity to be distanced from the EDMA3 activity. This means that there are multiple (usually two) sets of data buffers for all incoming and outgoing data streams. While the EDMA3 transfers the data into and out of the ping buffers, the CPU manipulates the data in the pong buffers. When both CPU and EDMA3 activity completes, they switch. The EDMA3 then writes over the old input data and transfers the new output data. [Figure 15-28](#) shows the ping-pong scheme for this example.

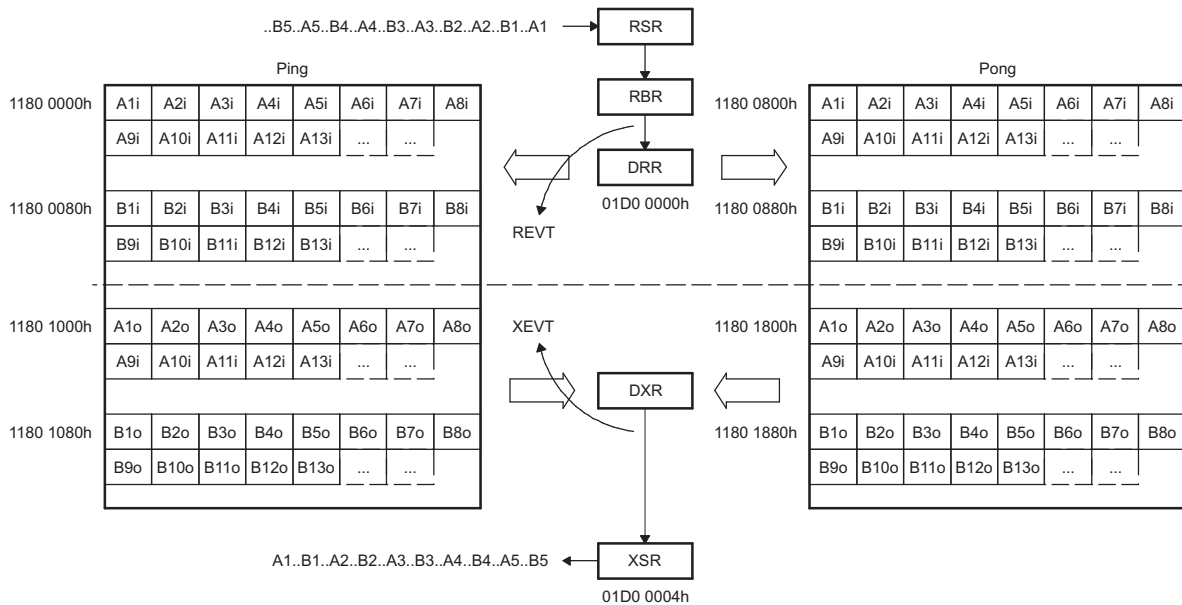
To change the continuous operation example, such that a ping-pong buffering scheme is used, the DMA channels need only a moderate change. Instead of one link parameter set, there are two; one for transferring data to/from the ping buffers and one for transferring data to/from the pong buffers. As soon as one transfer completes, the channel loads the PaRAM set for the other and the data transfers continue. [Figure 15-29](#) shows the DMA channel configuration required.

Each channel has two link parameter sets, ping and pong. The DMA channel is initially loaded with the ping parameters ([Figure 15-29](#)). The link address for the ping set is set to the PaRAM offset of the pong parameter set ([Figure 15-30](#)). The link address for the pong set is set to the PaRAM offset of the ping parameter set ([Figure 15-31](#)). The channel options, count values, and index values are all identical between the ping and pong parameters for each channel. The only differences are the link address provided and the address of the data buffer.

### 15.3.4.4.1 Synchronization with the CPU

In order to utilize the ping-pong buffering technique, the system must signal the CPU when to begin to access the new data set. After the CPU finishes processing an input buffer (ping), it waits for the EDMA3 to complete before switching to the alternate (pong) buffer. In this example, both channels provide their channel numbers as their report word and set the TCINTEN bit to 1 to generate an interrupt after completion. When channel 3 fills an input buffer, the E3 bit in the interrupt pending register (IPR) is set to 1; when channel 2 empties an output buffer, the E2 bit in IPR is set to 1. The CPU must manually clear these bits. With the channel parameters set, the CPU polls IPR to determine when to switch. The EDMA3 and CPU could alternatively be configured such that the channel completion interrupts the CPU. By doing this, the CPU could service a background task while waiting for the EDMA3 to complete.

Figure 15-28. Ping-Pong Buffering for McBSP Data Example



**Figure 15-29. Ping-Pong Buffering for McBSP Example PaRAM**

(a) EDMA Parameters for Channel 3 (Using PaRAM Set 3 Linked to Pong Set 64)

Parameter Contents		Parameter	
0010 3000h		Channel Options Parameter (OPT)	
01D0 0000h		Channel Source Address (SRC)	
0080h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
1180 0000h		Channel Destination Address (DST)	
0001h	0000h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0080h	4800h	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Channel 3

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000		0	0	0	1	00		00		
PRIV	Reserved		PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved		TCC		
15	12	11	10	8	7	4		3	2	1	0	
0011		0	000	0000			0	0	0	0		
TCC		TCCMOD	FWID	Reserved			STATIC	SYNCDIM	DAM	SAM		

(c) EDMA Parameters for Channel 2 (Using PaRAM Set 2 Linked to Pong Set 65)

Parameter Contents		Parameter	
0010 2000h		Channel Options Parameter (OPT)	
1180 1000h		Channel Source Address (SRC)	
0080h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
01D0 0004h		Channel Destination Address (DST)	
0000h	0001h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0080h	4840h	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Channel 2

31	30	28	27	24	23	22	21	20	19	18	17	16
0	000	0000		0	0	0	1	00		00		
PRIV	Reserved		PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved		TCC		
15	12	11	10	8	7	4		3	2	1	0	
0010		0	000	0000			0	0	0	0		
TCC		TCCMOD	FWID	Reserved			STATIC	SYNCDIM	DAM	SAM		

**Figure 15-30. Ping-Pong Buffering for McBSP Example Pong PaRAM**

(a) EDMA Pong Parameters for Channel 3 at Set 64 Linked to Set 65

Parameter Contents	
0010 D000h	
01D0 0000h	
0080h	0001h
1180 0800h	
0001h	0000h
0080h	4820h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) EDMA Pong Parameters for Channel 2 at Set 66 Linked to Set 67

Parameter Contents	
0010 C000h	
1180 1800h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4860h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

**Figure 15-31. Ping-Pong Buffering for McBSP Example Ping PaRAM**

(a) EDMA Ping Parameters for Channel 3 at Set 65 Linked to Set 64

Parameter Contents	
0010 D000h	
01D0 0000h	
0080h	0001h
1180 0000h	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) EDMA Ping Parameters for Channel 2 at Set 67 Linked to Set 66

Parameter Contents	
0010 C000h	
1180 1000h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4840h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)



### 15.3.4.5 Transfer Chaining Examples

The following examples explain the intermediate transfer complete chaining function.

#### 15.3.4.5.1 Servicing Input/Output FIFOs with a Single Event

Many systems require the use of a pair of external FIFOs that must be serviced at the same rate. One FIFO buffers data input, and the other buffers data output. The EDMA3 channels that service these FIFOs can be set up for AB-synchronized transfers. While each FIFO is serviced with a different set of parameters, both can be signaled from a single event. For example, an external interrupt pin can be tied to the status flags of one of the FIFOs. When this event arrives, the EDMA3 needs to perform servicing for both the input and output streams. Without the intermediate transfer complete chaining feature this would require two events, and thus two external interrupt pins. The intermediate transfer complete chaining feature allows the use of a single external event (for example, a GPIO event). [Figure 15-32](#) shows the EDMA3 setup and illustration for this example.

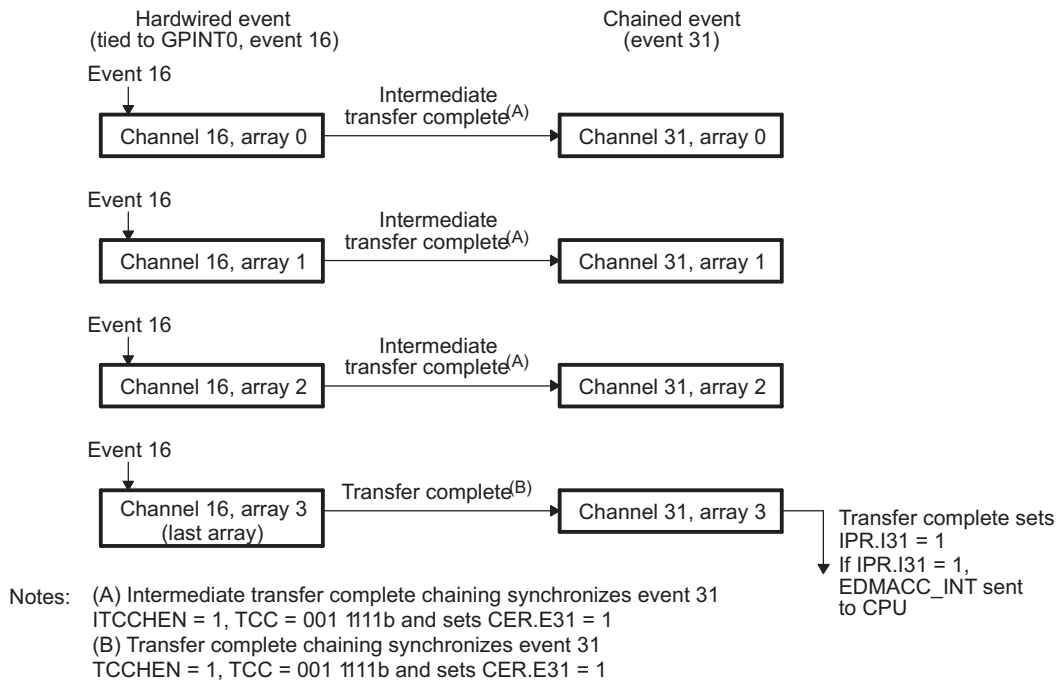
A GPIO event (in this case, GPINT0) triggers an array transfer. Upon completion of each intermediate array transfer of channel 16, intermediate transfer complete chaining sets the E31 bit (specified by TCC of 31) in the chained event register (CER) and provides a synchronization event to channel 31. Upon completion of the last array transfer of channel 16, transfer complete chaining—not intermediate transfer complete chaining—sets the E31 bit in CER (specified by TCCMODE:TCC) and provides a synchronization event to channel 31. The completion of channel 31 sets the I31 bit (specified by TCCMODE:TCC) in the interrupt pending register (IPR), which can generate an interrupt to the CPU, if the I31 bit in the interrupt enable register (IER) is set to 1.

#### 15.3.4.5.2 Breaking Up Large Transfers with Intermediate Chaining

Another feature of intermediate transfer chaining (ITCCHEN) is for breaking up large transfers. A large transfer may lock out other transfers of the same priority level for the duration of the transfer. For example, a large transfer on queue 0 from the internal memory to the external memory using the EMIF may starve other EDMA3 transfers on the same queue. In addition, this large high-priority transfer may prevent the EMIF for a long duration to service other lower priority transfers. When a large transfer is considered to be high priority, it should be split into multiple smaller transfers. [Figure 15-33](#) shows the EDMA3 setup and illustration of an example single large block transfer.

The intermediate transfer chaining enable (ITCCHEN) provides a method to break up a large transfer into smaller transfers. For example, to move a single large block of memory (16K bytes), the EDMA3 performs an A-synchronized transfer. The element count is set to a reasonable value, where reasonable derives from the amount of time it would take to move this smaller amount of data. Assume 1K byte is a reasonable small transfer in this example. The EDMA3 is set up to transfer 16 arrays of 1K byte elements, for a total of 16K byte elements. The TCC field in the channel options parameter (OPT) is set to the same value as the channel number and ITCCHEN are set. In this example, DMA channel 25 is used and TCC is also set to 25. The TCINTEN may also be set to trigger interrupt 25 when the last 1K byte array is transferred. The CPU starts the EDMA3 transfer by writing to the appropriate bit of the event set register (ESR.E25). The EDMA3 transfers the first 1K byte array. Upon completion of the first array, intermediate transfer complete code chaining generates a synchronization event to channel 25, a value specified by the TCC field. This intermediate transfer completion chaining event causes DMA channel 25 to transfer the next 1K byte array. This process continues until the transfer parameters are exhausted, at which point the EDMA3 has completed the 16K byte transfer. This method breaks up a large transfer into smaller packets, thus providing natural time slices in the transfer such that other events may be processed. [Figure 15-34](#) shows the EDMA3 setup and illustration of the broken up smaller packet transfers.

**Figure 15-32. Intermediate Transfer Completion Chaining Example**



Setup

Channel 16 parameters for chaining

- Enable transfer complete chaining:  
OPT.TCCHEN = 1  
OPT.TCC = 001 1111b
- Enable intermediate transfer complete chaining:  
OPT.ITCCHEN = 1  
OPT.TCC = 001 1111b

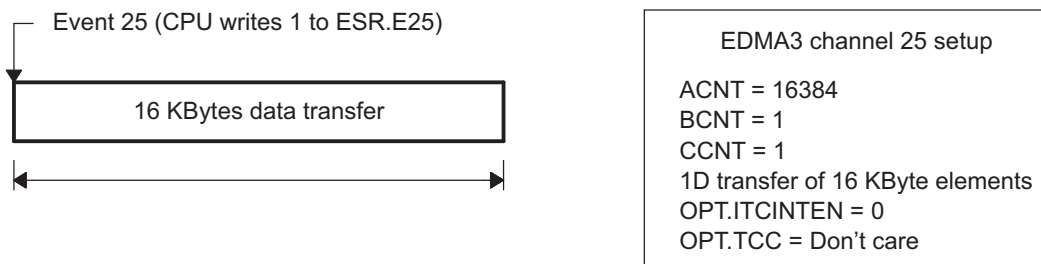
Channel 16 parameters for chaining

- Enable transfer completion interrupt:  
OPT.TCINTEN = 1  
OPT.TCC = 001 1111b
- Disable intermediate transfer complete chaining:  
OPT.ITCCHEN = 0

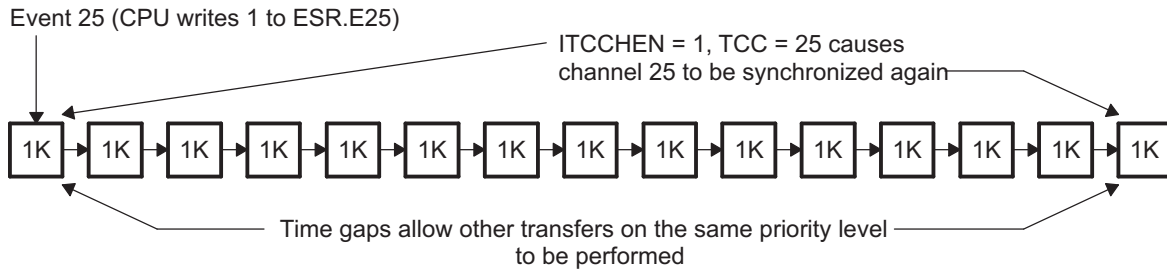
Event enable register (EER)

- Enable channel 16  
EER.E16 = 1

**Figure 15-33. Single Large Block Transfer Example**



**Figure 15-34. Smaller Packet Data Transfers Example**



EDMA channel 25 setup  
 ACNT = 1024  
 BCNT = 16  
 CCNT = 1  
 OPT.SYNCDIM = A SYNC  
 OPT.ITCCHEN = 1  
 OPT.TCINTEN = 1  
 OPT.TCC = 25

## 15.4 Registers

This section discusses the registers of the EDMA3 controller.

### 15.4.1 Parameter RAM (PaRAM) Entries

Table 15-14 lists the parameter RAM (PaRAM) entries for the EDMA3 channel controller (EDMA3CC). See your device-specific data manual for the memory address of these registers.

**Table 15-14. EDMA3 Channel Controller (EDMA3CC) Parameter RAM (PaRAM) Entries**

Offset	Acronym	Parameter	Section
0h	OPT	Channel Options	<a href="#">Section 15.4.1.1</a>
4h	SRC	Channel Source Address	<a href="#">Section 15.4.1.2</a>
8h	A_B_CNT	A Count/B Count	<a href="#">Section 15.4.1.3</a>
Ch	DST	Channel Destination Address	<a href="#">Section 15.4.1.4</a>
10h	SRC_DST_BIDX	Source B Index/Destination B Index	<a href="#">Section 15.4.1.5</a>
14h	LINK_BCNTRLD	Link Address/B Count Reload	<a href="#">Section 15.4.1.6</a>
18h	SRC_DST_CIDX	Source C Index/Destination C Index	<a href="#">Section 15.4.1.7</a>
1Ch	CCNT	C Count	<a href="#">Section 15.4.1.8</a>

### 15.4.1.1 Channel Options Parameter (OPT)

The channel options parameter (OPT) is shown in [Figure 15-35](#) and described in [Table 15-15](#).

**NOTE:** The TCC field in OPT is a 6-bit field and can be programmed for any value between 0-64. For devices with 32 DMA channels, the TCC field should have a value between 0 to 31 so that it sets the appropriate bits (0 to 31) in the interrupt pending register (IPR) (and can interrupt the CPU(s) on enabling the interrupt enable register (IER) bits (0-31)).

**Figure 15-35. Channel Options Parameter (OPT)**

31	28	27	24	23	22	21	20	19	18	17	16	
Reserved		PRIVID		ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved		TCC		
R-0		R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0		
15	12	11	10	8	7		4	3	2	1	0	
TCC		TCMOD	FWID	Reserved				STATIC	SYNCDIM	DAM	SAM	
R/W-0		R/W-0	R/W-0	R-0				R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-15. Channel Options Parameters (OPT) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	PRIVID	0-Fh	Privilege identification for the external host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege identification value when any part of the PaRAM set is written.
23	ITCCHEN	0 1	Intermediate transfer completion chaining enable. 0 Intermediate transfer complete chaining is disabled. 1 Intermediate transfer complete chaining is enabled.  When enabled, the chained event register (CER) bit is set on every intermediate chained transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in CER is the TCC value specified.
22	TCCHEN	0 1	Transfer complete chaining enable. 0 Transfer complete chaining is disabled. 1 Transfer complete chaining is enabled.  When enabled, the chained event register (CER) bit is set on final chained transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in CER is the TCC value specified.
21	ITCINTEN	0 1	Intermediate transfer completion interrupt enable. 0 Intermediate transfer complete interrupt is disabled. 1 Intermediate transfer complete interrupt is enabled.  When enabled, the interrupt pending register (IPR) bit is set on every intermediate transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in IPR is the TCC value specified. In order to generate a completion interrupt to the CPU, the corresponding IER[TCC] bit must be set to 1.
20	TCINTEN	0 1	Transfer complete interrupt enable. 0 Transfer complete interrupt is disabled. 1 Transfer complete interrupt is enabled.  When enabled, the interrupt pending register (IPR) bit is set on transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in IPR is the TCC value specified. In order to generate a completion interrupt to the CPU, the corresponding IER[TCC] bit must be set to 1.
19	Reserved	0	Reserved. Always write 0 to this bit.
18	Reserved	0	Reserved

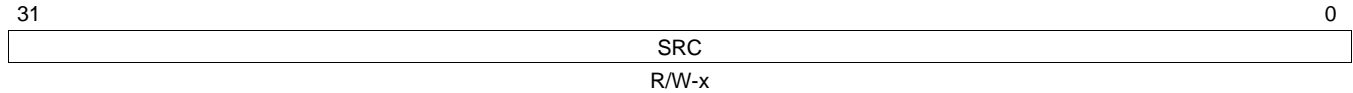
**Table 15-15. Channel Options Parameters (OPT) Field Descriptions (continued)**

Bit	Field	Value	Description
17-12	TCC	0-3Fh 0-1Fh 20h-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in chaining enable register (CER[TCC]) for chaining or in interrupt pending register (IPR[TCC]) for interrupts. Valid values Reserved
11	TCCMODE	0 1	Transfer complete code mode. Indicates the point at which a transfer is considered completed for chaining and interrupt generation. 0 Normal completion: A transfer is considered completed after the data has been transferred. 1 Early completion: A transfer is considered completed after the EDMA3CC submits a TR to the EDMA3TC. TC may still be transferring data when interrupt/chain is triggered.
10-8	FWID	0-7h 0 1h 2h 3h 4h 5h 6h-7h	FIFO Width. Applies if either SAM or DAM is set to constant addressing mode. 0 FIFO width is 8-bit. 1h FIFO width is 16-bit. 2h FIFO width is 32-bit. 3h FIFO width is 64-bit. 4h FIFO width is 128-bit. 5h FIFO width is 256-bit. 6h-7h Reserved
7-4	Reserved	0	Reserved
3	STATIC	0 1	Static PaRAM set. 0 PaRAM set is not static. PaRAM set is updated or linked after TR is submitted. A value of 0 should be used for DMA channels and for nonfinal transfers in a linked list of QDMA transfers. 1 PaRAM set is static. PaRAM set is not updated or linked after TR is submitted. A value of 1 should be used for isolated QDMA transfers or for the final transfer in a linked list of QDMA transfers.
2	SYNCDIM	0 1	Transfer synchronization dimension. 0 A-synchronized. Each event triggers the transfer of a single array of ACNT bytes. 1 AB-synchronized. Each event triggers the transfer of BCNT arrays of ACNT bytes.
1	DAM	0 1	Destination address mode. 0 Increment (INCR) mode. Destination addressing within an array increments. Destination is not a FIFO. 1 Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.  Note: The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. On the C674x/OMAP-L1x processors, no peripherals, memory, or memory controller support constant addressing mode. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM = 0) by appropriately programming the count and indices values.
0	SAM	0 1	Source address mode. 0 Increment (INCR) mode. Source addressing within an array increments. Source is not a FIFO. 1 Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.  Note: The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. On the C674x/OMAP-L1x processors, no peripherals, memory, or memory controller support constant addressing mode. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM = 0) by appropriately programming the count and indices values.

### 15.4.1.2 Channel Source Address Parameter (SRC)

The channel source address parameter (SRC) specifies the starting byte address of the source. The SRC is shown in [Figure 15-36](#) and described in [Table 15-16](#).

**Figure 15-36. Channel Source Address Parameter (SRC)**



LEGEND: R = Read only; -n = value after reset

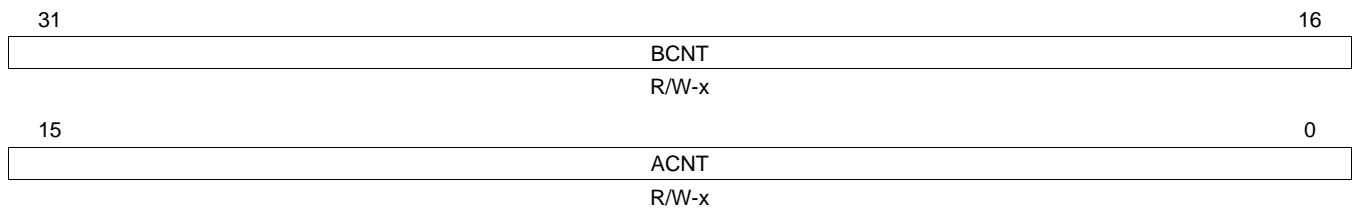
**Table 15-16. Channel Source Address Parameter (SRC) Field Descriptions**

Bit	Field	Value	Description
31-0	SRC	0-FFFF FFFFh	Source address. Specifies the starting byte address of the source.

### 15.4.1.3 A Count/B Count Parameter (A\_B\_CNT)

The A count/B count parameter (A\_B\_CNT) specifies the number of bytes within the 1st dimension of a transfer and the number of arrays of length ACNT. The A\_B\_CNT is shown in [Figure 15-37](#) and described in [Table 15-17](#).

**Figure 15-37. A Count/B Count Parameter (A\_B\_CNT)**



LEGEND: R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

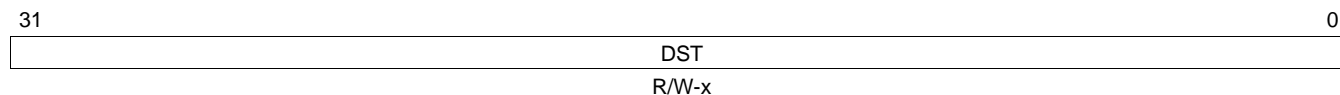
**Table 15-17. A Count/B Count Parameter (A\_B\_CNT) Field Descriptions**

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B count. Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535.
15-0	ACNT	0-FFFFh	A count for 1st Dimension. Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535.

#### 15.4.1.4 Channel Destination Address Parameter (DST)

The channel destination address parameter (DST) specifies the starting byte address of the source. The DST is shown in [Figure 15-38](#) and described in [Table 15-18](#).

**Figure 15-38. Channel Destination Address Parameter (DST)**



LEGEND: R = Read only; -n = value after reset

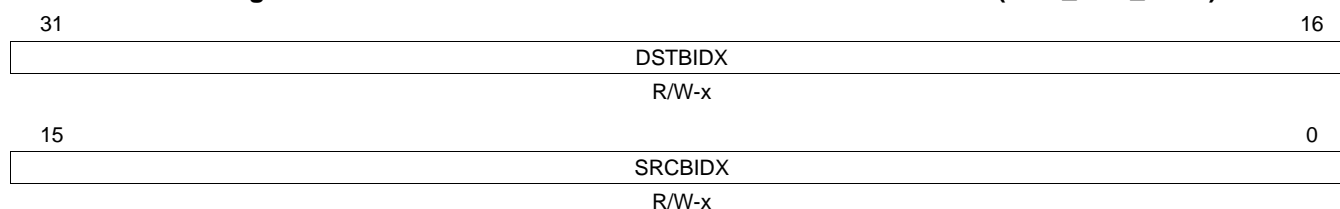
**Table 15-18. Channel Destination Address Parameter (DST) Field Descriptions**

Bit	Field	Value	Description
31-0	DST	0-FFFF FFFFh	Destination address. Specifies the starting byte address of the destination where data is transferred.

#### 15.4.1.5 Source B Index/Destination B Index Parameter (SRC\_DST\_BIDX)

The source B index/destination B index parameter (SRC\_DST\_BIDX) specifies the value (2s complement) used for source address modification between each array in the 2nd dimension and the value (2s complement) used for destination address modification between each array in the 2nd dimension. The SRC\_DST\_BIDX is shown in [Figure 15-39](#) and described in [Table 15-19](#).

**Figure 15-39. Source B Index/Destination B Index Parameter (SRC\_DST\_BIDX)**



LEGEND: R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

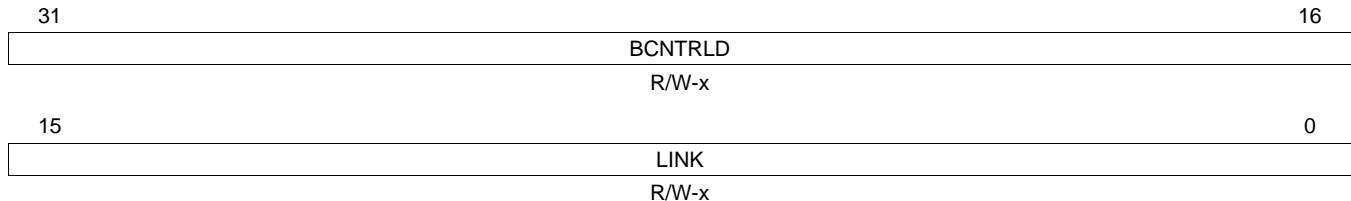
**Table 15-19. Source B Index/Destination B Index Parameter (SRC\_DST\_BIDX) Field Descriptions**

Bit	Field	Value	Description
31-16	DSTBIDX	0-FFFFh	Destination B index. Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from -32 768 and 32 767.
15-0	SRCBIDX	0-FFFFh	Source B index. Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from -32 768 and 32 767.

### 15.4.1.6 Link Address/B Count Reload Parameter (LINK\_BCNTRLD)

The link address/B count reload parameter (LINK\_BCNTRLD) specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking and the value used to reload the BCNT field in the A count/B count parameter (A\_B\_CNT) once the last array in the 2nd dimension is transferred. The LINK\_BCNTRLD is shown in [Figure 15-40](#) and described in [Table 15-20](#).

**Figure 15-40. Link Address/B Count Reload Parameter (LINK\_BCNTRLD)**



LEGEND: R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

**Table 15-20. Link Address/B Count Reload Parameter (LINK\_BCNTRLD) Field Descriptions**

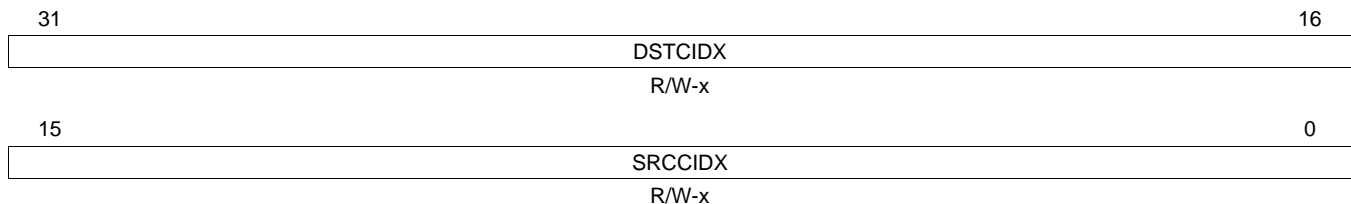
Bit	Field	Value	Description
31-16	BCNTRLD	0-FFFFh	B count reload. The count value used to reload BCNT in the A count/B count parameter (A_B_CNT) when BCNT decrements to 0 (TR submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers.
15-0	LINK	0-FFFFh	Link address. The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0. A value of FFFFh specifies a null link.



### 15.4.1.7 Source C Index/Destination C Index Parameter (SRC\_DST\_CIDX)

The source C index/destination C index parameter (SRC\_DST\_CIDX) specifies the value (2s complement) used for source address modification between each array in the 3rd dimension and the value (2s complement) used for destination address modification between each array in the 3rd dimension. The SRC\_DST\_CIDX is shown in [Figure 15-41](#) and described in [Table 15-21](#).

**Figure 15-41. Source C Index/Destination C Index Parameter (SRC\_DST\_CIDX)**



LEGEND: R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

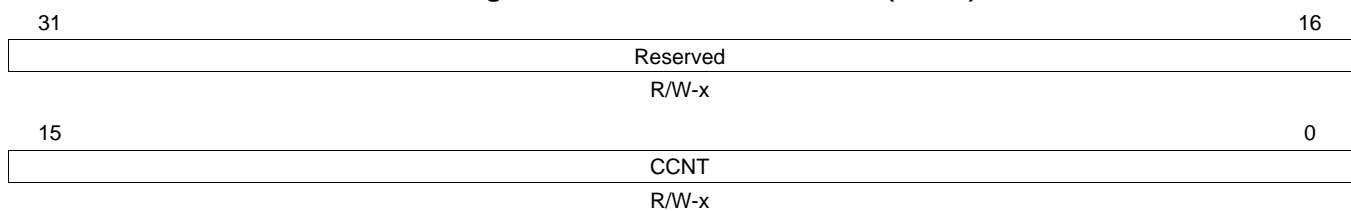
**Table 15-21. Source C Index/Destination C Index Parameter (SRC\_DST\_CIDX) Field Descriptions**

Bit	Field	Value	Description
31-16	DSTCIDX	0-FFFFh	Destination C index. Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from -32 768 and 32 767.
15-0	SRCCIDX	0-FFFFh	Source C index. Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from -32 768 and 32 767.

### 15.4.1.8 C Count Parameter (CCNT)

The C count parameter (CCNT) specifies the number of frames in a block. The CCNT is shown in [Figure 15-42](#) and described in [Table 15-22](#).

**Figure 15-42. C Count Parameter (CCNT)**



LEGEND: R/W = Read/Write; -n = value after reset; -x = value is indeterminate after reset

**Table 15-22. C Count Parameter (CCNT) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	CCNT	0-FFFFh	C counter. Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535.

## 15.4.2 EDMA3 Channel Controller (EDMA3CC) Registers

Table 15-23 lists the memory-mapped registers for the EDMA3 channel controller (EDMA3CC). See your device-specific data manual for the memory address of these registers and for the shadow region addresses. All other register offset addresses not listed in Table 15-23 should be considered as reserved locations and the register contents should not be modified.

**Table 15-23. EDMA3 Channel Controller (EDMA3CC) Registers**

Offset	Acronym	Register Description	Section
0h	REVID	Revision Identification Register	<a href="#">Section 15.4.2.1.1</a>
4h	CCCFCG	EDMA3CC Configuration Register	<a href="#">Section 15.4.2.1.2</a>
<b>Global Registers</b>			
200h	QCHMAP0	QDMA Channel 0 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
204h	QCHMAP1	QDMA Channel 1 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
208h	QCHMAP2	QDMA Channel 2 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
20Ch	QCHMAP3	QDMA Channel 3 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
210h	QCHMAP4	QDMA Channel 4 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
214h	QCHMAP5	QDMA Channel 5 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
218h	QCHMAP6	QDMA Channel 6 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
21Ch	QCHMAP7	QDMA Channel 7 Mapping Register	<a href="#">Section 15.4.2.1.3</a>
240h	DMAQNUM0	DMA Channel Queue Number Register 0	<a href="#">Section 15.4.2.1.4</a>
244h	DMAQNUM1	DMA Channel Queue Number Register 1	<a href="#">Section 15.4.2.1.4</a>
248h	DMAQNUM2	DMA Channel Queue Number Register 2	<a href="#">Section 15.4.2.1.4</a>
24Ch	DMAQNUM3	DMA Channel Queue Number Register 3	<a href="#">Section 15.4.2.1.4</a>
260h	QDMAQNUM	QDMA Channel Queue Number Register	<a href="#">Section 15.4.2.1.5</a>
284h	QUEPRI	Queue Priority Register <sup>(1)</sup>	<a href="#">Section 15.4.2.1.6</a>
300h	EMR	Event Missed Register	<a href="#">Section 15.4.2.2.1</a>
308h	EMCR	Event Missed Clear Register	<a href="#">Section 15.4.2.2.2</a>
310h	QEMR	QDMA Event Missed Register	<a href="#">Section 15.4.2.2.3</a>
314h	QEMCR	QDMA Event Missed Clear Register	<a href="#">Section 15.4.2.2.4</a>
318h	CCERR	EDMA3CC Error Register	<a href="#">Section 15.4.2.2.5</a>
31Ch	CCERRCLR	EDMA3CC Error Clear Register	<a href="#">Section 15.4.2.2.6</a>
320h	EEVAL	Error Evaluate Register	<a href="#">Section 15.4.2.2.7</a>
340h	DRAE0	DMA Region Access Enable Register for Region 0	<a href="#">Section 15.4.2.3.1</a>
348h	DRAE1	DMA Region Access Enable Register for Region 1	<a href="#">Section 15.4.2.3.1</a>
350h	DRAE2	DMA Region Access Enable Register for Region 2	<a href="#">Section 15.4.2.3.1</a>
358h	DRAE3	DMA Region Access Enable Register for Region 3	<a href="#">Section 15.4.2.3.1</a>
380h	QRAE0	QDMA Region Access Enable Register for Region 0	<a href="#">Section 15.4.2.3.2</a>
384h	QRAE1	QDMA Region Access Enable Register for Region 1	<a href="#">Section 15.4.2.3.2</a>
388h	QRAE2	QDMA Region Access Enable Register for Region 2	<a href="#">Section 15.4.2.3.2</a>
38Ch	QRAE3	QDMA Region Access Enable Register for Region 3	<a href="#">Section 15.4.2.3.2</a>
400h-43Ch	Q0E0-Q0E15	Event Queue Entry Registers Q0E0-Q0E15	<a href="#">Section 15.4.2.4.1</a>
440h-47Ch	Q1E0-Q1E15	Event Queue Entry Registers Q1E0-Q1E15	<a href="#">Section 15.4.2.4.1</a>
600h	QSTAT0	Queue 0 Status Register	<a href="#">Section 15.4.2.4.2</a>
604h	QSTAT1	Queue 1 Status Register	<a href="#">Section 15.4.2.4.2</a>
620h	QWMTHRA	Queue Watermark Threshold A Register	<a href="#">Section 15.4.2.4.3</a>
640h	CCSTAT	EDMA3CC Status Register	<a href="#">Section 15.4.2.4.4</a>

<sup>(1)</sup> On previous architectures, the EDMA3TC priority was controlled by the queue priority register (QUEPRI) in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module. You should use the chip-level registers and not QUEPRI to configure the TC priority.

**Table 15-23. EDMA3 Channel Controller (EDMA3CC) Registers (continued)**

Offset	Acronym	Register Description	Section
<b>Global Channel Registers</b>			
1000h	ER	Event Register	<a href="#">Section 15.4.2.5.1</a>
1008h	ECR	Event Clear Register	<a href="#">Section 15.4.2.5.2</a>
1010h	ESR	Event Set Register	<a href="#">Section 15.4.2.5.3</a>
1018h	CER	Chained Event Register	<a href="#">Section 15.4.2.5.4</a>
1020h	EER	Event Enable Register	<a href="#">Section 15.4.2.5.5</a>
1028h	EECR	Event Enable Clear Register	<a href="#">Section 15.4.2.5.6</a>
1030h	EESR	Event Enable Set Register	<a href="#">Section 15.4.2.5.7</a>
1038h	SER	Secondary Event Register	<a href="#">Section 15.4.2.5.8</a>
1040h	SECR	Secondary Event Clear Register	<a href="#">Section 15.4.2.5.9</a>
1050h	IER	Interrupt Enable Register	<a href="#">Section 15.4.2.6.1</a>
1058h	IECR	Interrupt Enable Clear Register	<a href="#">Section 15.4.2.6.2</a>
1060h	IESR	Interrupt Enable Set Register	<a href="#">Section 15.4.2.6.3</a>
1068h	IPR	Interrupt Pending Register	<a href="#">Section 15.4.2.6.4</a>
1070h	ICR	Interrupt Clear Register	<a href="#">Section 15.4.2.6.5</a>
1078h	IEVAL	Interrupt Evaluate Register	<a href="#">Section 15.4.2.6.6</a>
1080h	QER	QDMA Event Register	<a href="#">Section 15.4.2.7.1</a>
1084h	QEER	QDMA Event Enable Register	<a href="#">Section 15.4.2.7.2</a>
1088h	QEECR	QDMA Event Enable Clear Register	<a href="#">Section 15.4.2.7.3</a>
108Ch	QEESR	QDMA Event Enable Set Register	<a href="#">Section 15.4.2.7.4</a>
1090h	QSER	QDMA Secondary Event Register	<a href="#">Section 15.4.2.7.5</a>
1094h	QSECR	QDMA Secondary Event Clear Register	<a href="#">Section 15.4.2.7.6</a>
<b>Shadow Region 0 Channel Registers</b>			
2000h	ER	Event Register	—
2008h	ECR	Event Clear Register	—
2010h	ESR	Event Set Register	—
2018h	CER	Chained Event Register	—
2020h	EER	Event Enable Register	—
2028h	EECR	Event Enable Clear Register	—
2030h	EESR	Event Enable Set Register	—
2038h	SER	Secondary Event Register	—
2040h	SECR	Secondary Event Clear Register	—
2050h	IER	Interrupt Enable Register	—
2058h	IECR	Interrupt Enable Clear Register	—
2060h	IESR	Interrupt Enable Set Register	—
2068h	IPR	Interrupt Pending Register	—
2070h	ICR	Interrupt Clear Register	—
2078h	IEVAL	Interrupt Evaluate Register	—
2080h	QER	QDMA Event Register	—
2084h	QEER	QDMA Event Enable Register	—
2088h	QEECR	QDMA Event Enable Clear Register	—
208Ch	QEESR	QDMA Event Enable Set Register	—
2090h	QSER	QDMA Secondary Event Register	—
2094h	QSECR	QDMA Secondary Event Clear Register	—

**Table 15-23. EDMA3 Channel Controller (EDMA3CC) Registers (continued)**

Offset	Acronym	Register Description	Section
<b>Shadow Region 1 Channel Registers</b>			
2200h	ER	Event Register	—
2208h	ECR	Event Clear Register	—
2210h	ESR	Event Set Register	—
2218h	CER	Chained Event Register	—
2220h	EER	Event Enable Register	—
2228h	EECR	Event Enable Clear Register	—
2230h	EESR	Event Enable Set Register	—
2238h	SER	Secondary Event Register	—
2240h	SECR	Secondary Event Clear Register	—
2250h	IER	Interrupt Enable Register	—
2258h	IECR	Interrupt Enable Clear Register	—
2260h	IESR	Interrupt Enable Set Register	—
2268h	IPR	Interrupt Pending Register	—
2270h	ICR	Interrupt Clear Register	—
2278h	IEVAL	Interrupt Evaluate Register	—
2280h	QER	QDMA Event Register	—
2284h	QEER	QDMA Event Enable Register	—
2288h	QEECR	QDMA Event Enable Clear Register	—
228Ch	QEESR	QDMA Event Enable Set Register	—
2290h	QSER	QDMA Secondary Event Register	—
2294h	QSECR	QDMA Secondary Event Clear Register	—
4000h-4FFFh	—	Parameter RAM (PaRAM)	—

## 15.4.2.1 Global Registers

### 15.4.2.1.1 Revision Identification Register (REVID)

The revision identification register (REVID) uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC. The REVID is shown in [Figure 15-43](#) and described in [Table 15-24](#).

**Figure 15-43. Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

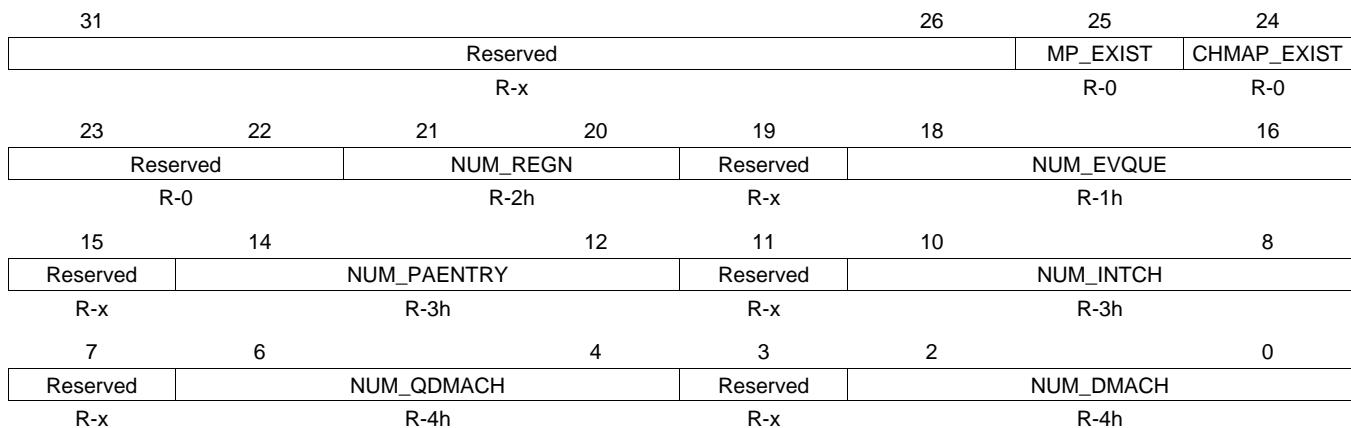
**Table 15-24. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4001 5300h	Peripheral identifier. Uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC.

### 15.4.2.1.2 EDMA3CC Configuration Register (CCCFG)

The EDMA3CC configuration register (CCCFG) provides the features/resources for the EDMA3CC in a particular device. The CCCFG is shown in [Figure 15-44](#) and described in [Table 15-25](#).

**Figure 15-44. EDMA3CC Configuration Register (CCCFG)**



LEGEND: R = Read only; -n = value after reset; -x = value is indeterminate after reset

**Table 15-25. EDMA3CC Configuration Register (CCCFG) Field Descriptions**

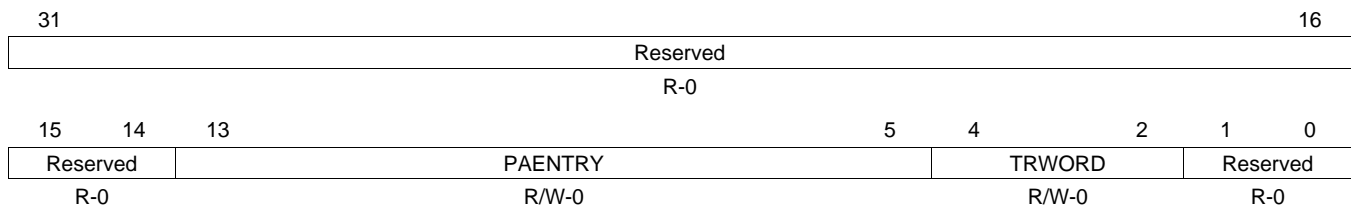
Bit	Field	Value	Description
31-26	Reserved	0-3Fh	Reserved
25	MP_EXIST	0 1	Memory protection existence. No memory protection. Reserved
24	CHMAP_EXIST	0 1	Channel mapping existence. No channel mapping. This implies that there is fixed association for a channel number to a parameter entry number or, in other words, PaRAM entry <i>n</i> corresponds to channel <i>n</i> . Reserved
23-22	Reserved	0	Reserved
21-20	NUM_REGN	0-3h 0-1h 2h 3h	Number of shadow regions. Reserved 4 regions Reserved
19	Reserved	0	Reserved
18-16	NUM_EVQUE	0-7h 0 1h 2h 3h-7h	Number of queues/number of transfer controllers. Reserved 2 event queues 2 transfer controllers Reserved
15	Reserved	0	Reserved
14-12	NUM_PAENTRY	0-7h 0-2h 3h 4h-7h	Number of PaRAM sets. Reserved 128 PaRAM sets Reserved
11	Reserved	0	Reserved
10-8	NUM_INTCH	0-7h 0-2h 3h 4h-7h	Number of interrupt channels. Reserved 32 interrupt channels Reserved
7	Reserved	0	Reserved
6-4	NUM_QDMACH	0-7h 0-3h 4h 5h-7h	Number of QDMA channels. Reserved 8 QDMA channels Reserved
3	Reserved	0	Reserved
2-0	NUM_DMACH	0-7h 0-3h 4h 5h-7h	Number of DMA channels. Reserved 32 DMA channels Reserved

### 15.4.2.1.3 QDMA Channel $n$ Mapping Register (QCHMAP $n$ )

Each QDMA channel in EDMA3CC can be associated with any PaRAM set available on the device. Furthermore, the specific trigger word (0-7) of the PaRAM set can be programmed. The PaRAM set association and trigger word for every QDMA channel register is configurable using the QDMA channel  $n$  mapping register (QCHMAP $n$ ). The QCHMAP $n$  is shown in [Figure 15-45](#) and described in [Table 15-26](#).

**NOTE:** At reset the QDMA channel mapping registers for all QDMA channels point to the PaRAM set 0. Prior to using any QDMA channel, QCHMAP $n$  should be programmed appropriately to point to a different PaRAM set.

**Figure 15-45. QDMA Channel  $n$  Mapping Register (QCHMAP $n$ )**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 15-26. QDMA Channel  $n$  Mapping Register (QCHMAP $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-5	PAENTRY	0-1FFh 0-7Fh 80h-1FFh	PAENTRY points to the PaRAM set number for QDMA channel $n$ . PaRAM set number 0 through 127 Reserved
4-2	TRWORD	0-7h	Points to the specific PaRAM entry or the trigger word in the PaRAM set pointed to by PAENTRY. A write to the trigger word results in a QDMA event being recognized.
1-0	Reserved	0	Reserved

**15.4.2.1.4 DMA Channel Queue Number Register *n* (DMAQNUM<sub>*n*</sub>)**

The DMA channel queue number register *n* (DMAQNUM<sub>*n*</sub>) allows programmability of each of the 32 DMA channels in the EDMA3CC to submit its associated synchronization event to any event queue in the EDMA3CC. At reset, all channels point to event queue 0. The DMAQNUM<sub>*n*</sub> is shown in Figure 15-46 and described in . Table 15-28 shows the channels and their corresponding bits in DMAQNUM<sub>*n*</sub>.

**NOTE:** Since the event queues in EDMA3CC have a fixed association to the transfer controllers, that is, Q0 TRs are submitted to TC0 and Q1 TRs are submitted to TC1, by programming DMAQNUM<sub>*n*</sub> for a particular DMA channel also dictates which transfer controller is utilized for the data movement (or which EDMA3TC receives the TR request).

**Figure 15-46. DMA Channel Queue Number Register *n* (DMAQNUM<sub>*n*</sub>)**

31	30	28	27	26	24	23	22	20	19	18	16
Rsvd	En	Rsvd	En	Rsvd	En	Rsvd	En	Rsvd	En	Rsvd	En
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0
15	14	12	11	10	8	7	6	4	3	2	0
Rsvd	En	Rsvd	En	Rsvd	En	Rsvd	En	Rsvd	En	Rsvd	En
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 15-27. DMA Channel Queue Number Register *n* (DMAQNUM<sub>*n*</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	En	0-7h	DMA queue number. Contains the event queue number to be used for the corresponding DMA channel. Programming DMAQNUM <sub><i>n</i></sub> for an event queue number to a value more than the number of queues available in the EDMA3CC results in undefined behavior.
		0	Event <i>n</i> is queued on Q0.
		1h	Event <i>n</i> is queued on Q1. Available only for EDMA3_0_CC0; for EDMA3_1_CC0, this value is reserved and all events are queued on Q0.
		2h-7h	Reserved

**Table 15-28. Bits in DMAQNUM<sub>*n*</sub>**

En bit	DMAQNUM <sub><i>n</i></sub>			
	0	1	2	3
0-2	E0	E8	E16	E24
4-6	E1	E9	E17	E25
8-10	E2	E10	E18	E26
12-14	E3	E11	E19	E27
16-18	E4	E12	E20	E28
20-22	E5	E13	E21	E29
24-26	E6	E14	E22	E30
28-30	E7	E15	E23	E31



### 15.4.2.1.5 QDMA Channel Queue Number Register (QDMAQNUM)

The QDMA channel queue number register (QDMAQNUM) is used to program all the QDMA channels in the EDMA3CC to submit the associated QDMA event to any of the event queues in the EDMA3CC. The QDMAQNUM is shown in [Figure 15-47](#) and described in .

**Figure 15-47. QDMA Channel Queue Number Register (QDMAQNUM)**

31	30	28	27	26	24	23	22	20	19	18	16
Rsvd	E7		Rsvd	E6		Rsvd	E5		Rsvd	E4	
R-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	
15	14	12	11	10	8	7	6	4	3	2	0
Rsvd	E3		Rsvd	E2		Rsvd	E1		Rsvd	E0	
R-0	R/W-0		R-0	R/W-0		R-0	R/W-0		R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-29. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>	0-7h	QDMA queue number. Contains the event queue number to be used for the corresponding QDMA channel.
		0	Event <i>n</i> is queued on Q0.
		1h	Event <i>n</i> is queued on Q1. Available only for EDMA3_0_CC0; for EDMA3_1_CC0, this value is reserved and all events are queued on Q0.
		2h-7h	Reserved

### 15.4.2.1.6 Queue Priority Register (QUEPRI)

On previous architectures, the EDMA3TC priority was controlled by the queue priority register (QUEPRI) in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module. You should use the chip-level registers and not QUEPRI to configure the TC priority.

### 15.4.2.2 Error Registers

The EDMA3CC contains a set of registers that provide information on missed DMA and/or QDMA events, and instances when event queue thresholds are exceeded. If any of the bits in these registers is set, it results in the EDMA3CC generating an error interrupt.

#### 15.4.2.2.1 Event Missed Registers (EMR)

For a particular DMA channel, if a second event is received prior to the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the event missed register (EMR). All trigger types are treated individually, that is, manual triggered (ESR), chain triggered (CER), and event triggered (ER) are all treated separately. The EMR bit for a channel is also set if an event on that channel encounters a NULL entry (or a NULL TR is serviced). If any EMR bit is set (and all errors, including bits in other error registers (QEMR, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See [Section 15.2.9.4](#) for details on EDMA3CC error interrupt generation.

The EMR is shown in [Figure 15-48](#) and described in [Table 15-30](#).

**Figure 15-48. Event Missed Register (EMR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-30. Event Missed Register (EMR) Field Descriptions**

Bit	Field	Value	Description
31-0	$E_n$		Channel 0-31 event missed. $E_n$ is cleared by writing a 1 to the corresponding bit in the event missed clear register (EMCR).
		0	No missed event.
		1	Missed event occurred.

### 15.4.2.2.2 Event Missed Clear Registers (EMCR)

Once a missed event is posted in the event missed register (EMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the event missed clear register (EMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in EMR; writing a 0 has no effect.

The EMCR is shown in [Figure 15-49](#) and described in [Table 15-31](#).

**Figure 15-49. Event Missed Clear Register (EMCR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-31. Event Missed Clear Register (EMCR) Field Descriptions**

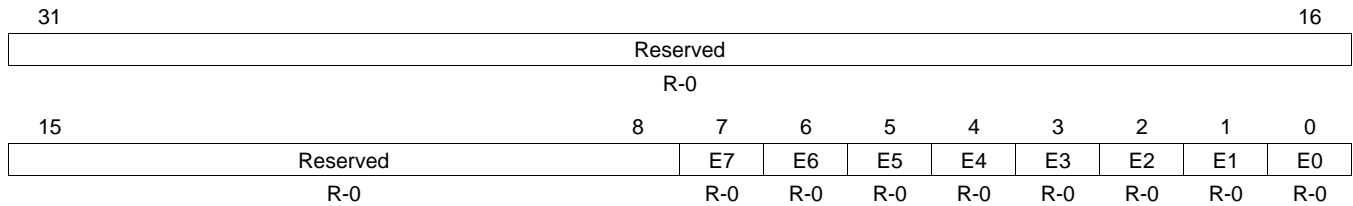
Bit	Field	Value	Description
31-0	$E_n$		Event missed 0-31 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.
		0	No effect.
		1	Corresponding missed event bit in the event missed register (EMR) is cleared ( $E_n = 0$ ).

**15.4.2.2.3 QDMA Event Missed Register (QEMR)**

For a particular QDMA channel, if two QDMA events are detected without the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the QDMA event missed register (QEMR). The QEMR bits for a channel are also set if a QDMA event on the channel encounters a NULL entry (or a NULL TR is serviced). If any QEMR bit is set (and all errors, including bits in other error registers (EMR or CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See [Section 15.2.9.4](#) for details on EDMA3CC error interrupt generation.

The QEMR is shown in [Figure 15-50](#) and described in [Table 15-32](#).

**Figure 15-50. QDMA Event Missed Register (QEMR)**



LEGEND: R = Read only; -n = value after reset

**Table 15-32. QDMA Event Missed Register (QEMR) Field Descriptions**

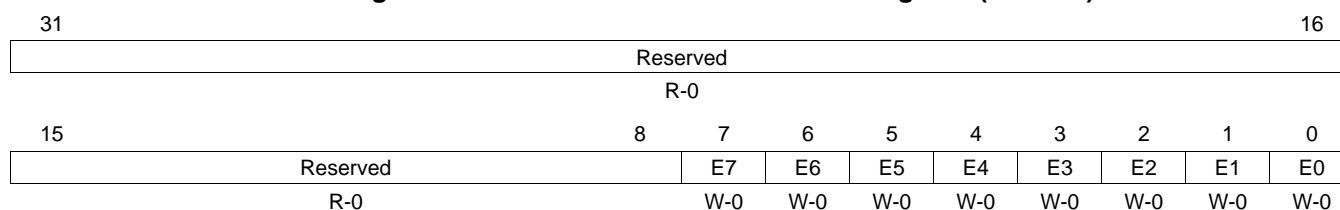
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	$E_n$	0	Channel 0-7 QDMA event missed. $E_n$ is cleared by writing a 1 to the corresponding bit in the QDMA event missed clear register (QEMCR).
		1	Missed event occurred.

#### 15.4.2.2.4 QDMA Event Missed Clear Register (QEMCR)

Once a missed event is posted in the QDMA event missed registers (QEMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the QDMA event missed clear registers (QEMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in QEMR; writing a 0 has no effect.

The QEMCR is shown in [Figure 15-51](#) and described in [Table 15-33](#).

**Figure 15-51. QDMA Event Missed Clear Register (QEMCR)**



LEGEND: W = Write only; -n = value after reset

**Table 15-33. QDMA Event Missed Clear Register (QEMCR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	$E_n$	0	QDMA event missed clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC. No effect.
		1	Corresponding missed event bit in the QDMA event missed register (QEMR) is cleared ( $E_n = 0$ ).

### 15.4.2.2.5 EDMA3CC Error Register (CCERR)

The EDMA3CC error register (CCERR) indicates whether or not at any instant of time the number of events queued up in any of the event queues exceeds or equals the threshold/watermark value that is set in the queue watermark threshold register (QWMTHRA). Additionally, CCERR also indicates if when the number of outstanding TRs that have been programmed to return transfer completion code (TRs that have the TCINTEN or TCCHEN bit in OPT set to 1) to the EDMA3CC has exceeded the maximum allowed value of 31. If any bit in CCERR is set (and all errors, including bits in other error registers (EMR or QEMR) were previously cleared), the EDMA3CC generates an error interrupt. See [Section 15.2.9.4](#) for details on EDMA3CC error interrupt generation. Once the error bits are set in CCERR, they can only be cleared by writing to the corresponding bits in the EDMA3CC error clear register (CCERRCLR).

The CCERR is shown in [Figure 15-52](#) and described in [Table 15-34](#).

**Figure 15-52. EDMA3CC Error Register (CCERR)**

31	Reserved	17	16
R-0			TCCERR R-0
15	Reserved	2	1      0
R-0		QTHRCD1 R-0	QTHRCD0 R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-34. EDMA3CC Error Register (CCERR) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	TCCERR		Transfer completion code error. TCCERR is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Total number of allowed TCCs outstanding has not been reached.
		1	Total number of allowed TCCs has been reached.
15-2	Reserved	0	Reserved
1	QTHRCD1		Queue threshold error for queue 1. QTHRCD1 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.
0	QTHRCD0		Queue threshold error for queue 0. QTHRCD0 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.

### 15.4.2.2.6 EDMA3CC Error Clear Register (CCERRCLR)

The EDMA3CC error clear register (CCERRCLR) is used to clear any error bits that are set in the EDMA3CC error register (CCERR). In addition, CCERRCLR also clears the values of some bit fields in the queue status registers (QSTAT $n$ ) associated with a particular event queue. Writing a 1 to any of the bits clears the corresponding bit in CCERR; writing a 0 has no effect.

The CCERRCLR is shown in [Figure 15-53](#) and described in [Table 15-35](#).

**Figure 15-53. EDMA3CC Error Clear Register (CCERRCLR)**

31	Reserved	17	TCCERR
	W-0		W-0
15	Reserved	2	1
	W-0	QTHRXC1	QTHRXC0
		W-0	W-0

LEGEND: W= Write only; -n = value after reset

**Table 15-35. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions**

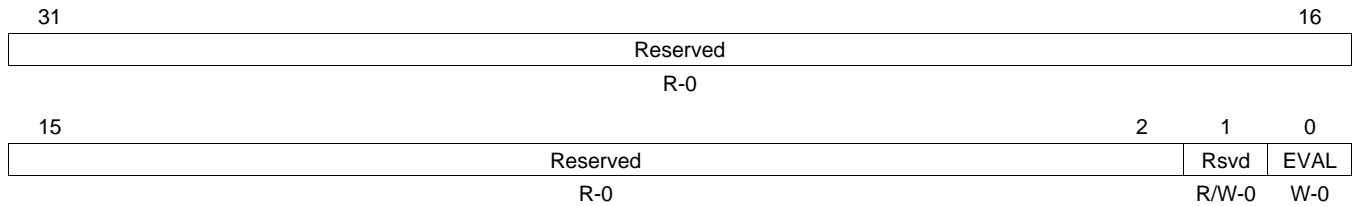
Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	TCCERR	0	Transfer completion code error clear.
		1	No effect. Clears the TCCERR bit in the EDMA3CC error register (CCERR).
15-2	Reserved	0	Reserved
1	QTHRXC1	0	Queue threshold error clear for queue 1.
		1	No effect. Clears the QTHRXC1 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 1 (QSTAT1).
0	QTHRXC0	0	Queue threshold error clear for queue 0.
		1	No effect. Clears the QTHRXC0 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 0 (QSTAT0).

**15.4.2.2.7 Error Evaluate Register (EEVAL)**

The EDMA3CC error interrupt is asserted whenever an error bit is set in any of the error registers (EMR, QEMR, and CCERR). For subsequent error bits that get set, the EDMA3CC error interrupt is reasserted only when transitioning from an “all the error bits cleared” to “at least one error bit is set”. Alternatively, a CPU write of 1 to the EVAL bit in the error evaluate register (EEVAL) results in reasserting the EDMA3CC error interrupt, if there are any outstanding error bits set due to subsequent error conditions. Writes of 0 have no effect.

The EEVAL is shown in [Figure 15-54](#) and described in [Table 15-36](#).

**Figure 15-54. Error Evaluate Register (EEVAL)**



LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

**Table 15-36. Error Evaluate Register (EEVAL) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL	0	Error interrupt evaluate. No effect.
		1	EDMA3CC error interrupt will be pulsed if any errors have not been cleared in any of the error registers (EMR, QEMR, or CCERR).



### 15.4.2.3 Region Access Enable Registers

The region access enable register group consists of the DMA access enable registers (DRAEm) and the QDMA access enable registers (QRAEm). Where *m* is the number of shadow regions in the EDMA3CC memory-map for a device. You can configure these registers to assign ownership of DMA/QDMA channels to a particular shadow region.

#### 15.4.2.3.1 DMA Region Access Enable for Region *m* (DRAEm)

The DMA region access enable registers for shadow region *m* (DRAEm) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all DMA registers in the shadow region *m* view of the DMA channel registers. See the EDMA3CC register memory-map for a list of all the DMA channel and interrupt registers mapped in the shadow region view. Additionally, the DRAEm configuration determines completion of which DMA channels will result in assertion of the shadow region *m* DMA completion interrupt (see [Section 15.2.9](#)).

The DRAEm is shown in [Figure 15-55](#) and described in [Table 15-37](#).

**Figure 15-55. DMA Region Access Enable Register for Region *m* (DRAEm)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 15-37. DMA Region Access Enable Register for Region *m* (DRAEm) Field Descriptions**

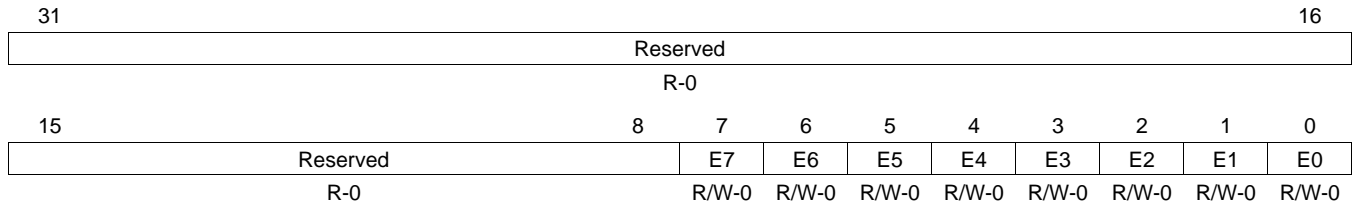
Bit	Field	Value	Description
31-0	<i>En</i>	0	DMA region access enable for bit <i>n</i> /channel <i>n</i> in region <i>m</i> . Accesses via region <i>m</i> address space to bit <i>n</i> in any DMA channel register are not allowed. Reads return 0 on bit <i>n</i> and writes do not modify the state of bit <i>n</i> . Enabled interrupt bits for bit <i>n</i> do not contribute to the generation of a transfer completion interrupt for shadow region <i>m</i> .
		1	Accesses via region <i>m</i> address space to bit <i>n</i> in any DMA channel register are allowed. Reads return the value from bit <i>n</i> and writes modify the state of bit <i>n</i> . Enabled interrupt bits for bit <i>n</i> contribute to the generation of a transfer completion interrupt for shadow region <i>m</i> .

### 15.4.2.3.2 QDMA Region Access Enable Registers (QRAEm)

The QDMA region access enable registers for shadow region  $m$  (QRAEm) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all QDMA registers in the shadow region  $m$  view of the QDMA registers. This includes all 8-bit QDMA registers.

The QRAEm is shown in [Figure 15-56](#) and described in [Table 15-38](#).

**Figure 15-56. QDMA Region Access Enable for Region  $m$  (QRAEm)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 15-38. QDMA Region Access Enable for Region  $m$  (QRAEm) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	$E_n$	0	QDMA region access enable for bit $n$ /QDMA channel $n$ in region $m$ . Accesses via region $m$ address space to bit $n$ in any QDMA channel register are not allowed. Reads return 0 on bit $n$ and writes do not modify the state of bit $n$ .
		1	Accesses via region $m$ address space to bit $n$ in any QDMA channel register are allowed. Reads return the value from bit $n$ and writes modify the state of bit $n$ .

### 15.4.2.4 Status/Debug Visibility Registers

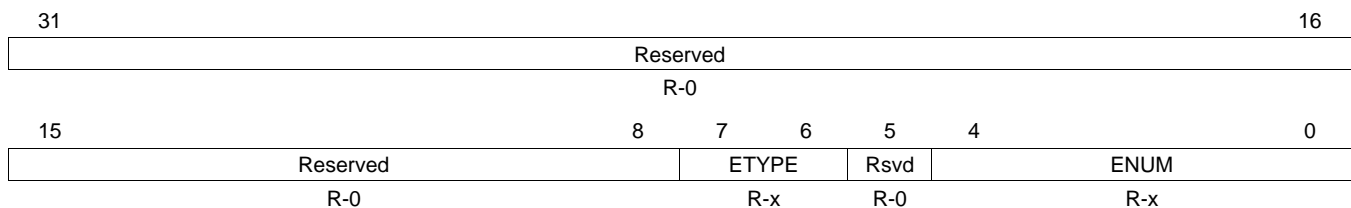
The following set of registers provide visibility into the event queues and a TR lifecycle. These are useful for system debug as they provide in-depth visibility for the events queued up in the event queue and also provide information on what parts of the EDMA3CC logic are active once the event has been received by the EDMA3CC.

#### 15.4.2.4.1 Event Queue Entry Registers (QxEy)

The event queue entry registers (QxEy) exist for all 16 queue entries (the maximum allowed queue entries) for all event queues (Q0 and Q1) in the EDMA3CC: Q0E0 to Q0E15 and Q1E0 to Q1E15. Each register details the event number (ENUM) and the event type (ETYPE). For example, if the value in Q1E4 is read as 0000 004Fh, this means the 4th entry in queue 1 is a manually-triggered event on DMA channel 15.

The QxEy is shown in [Figure 15-57](#) and described in [Table 15-39](#).

**Figure 15-57. Event Queue Entry Registers (QxEy)**



LEGEND: R = Read only; -n = value after reset; -x = value is indeterminate after reset

**Table 15-39. Event Queue Entry Registers (QxEy) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	ETYPE	0-3h	Event entry y in queue x. Specifies the specific event type for the given entry in the event queue.
		0	Event triggered via ER
		1h	Manual triggered via ESR
		2h	Chain triggered via CER
		3h	Autotriggered via QER
5	Reserved	0	Reserved
4-0	ENUM	0-1Fh	Event entry y in queue x. Event number:
		0-7h	QDMA channel number (0 to 7)
		0-1Fh	DMA channel/event number (0 to 31)

### 15.4.2.4.2 Queue *n* Status Registers (QSTAT<sub>*n*</sub>)

The queue *n* status register (QSTAT<sub>*n*</sub>) is shown in [Figure 15-58](#) and described in [Table 15-40](#).

**Figure 15-58. Queue *n* Status Register (QSTAT<sub>*n*</sub>)**

31	Reserved				25	24	23	21	20	16	
					R-0	THRXC <small>D</small>	Reserved		WM		
					R-0	R-0	R-0		R-0		
15	13	12					8	7	4	3	0
Reserved		NUMVAL				Reserved		STRTP <small>TR</small>			
R-0		R-0				R-0		R-0			

LEGEND: R = Read only; -*n* = value after reset

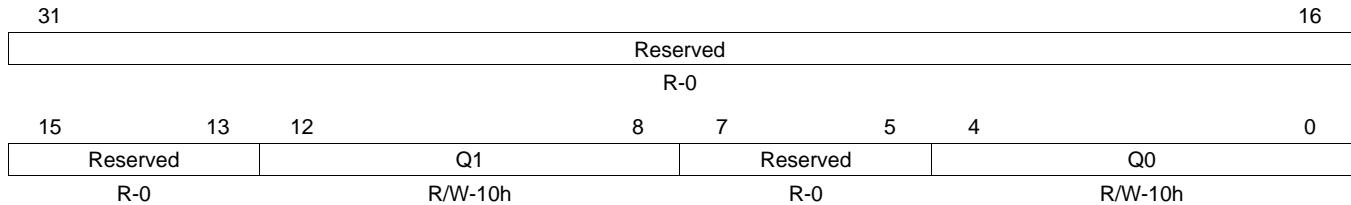
**Table 15-40. Queue *n* Status Register (QSTAT<sub>*n*</sub>) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	THRXC <small>D</small>	0	Threshold exceeded. THRXC <small>D</small> is cleared by writing a 1 to the corresponding QTHRXC <small>D</small> <sub><i>n</i></sub> bit in the EDMA3CC error clear register (CCERRCLR).
		1	Threshold specified by the <i>Qn</i> bit in the queue watermark threshold A register (QWMTHRA) has not been exceeded.
		1	Threshold specified by the <i>Qn</i> bit in the queue watermark threshold A register (QWMTHRA) has been exceeded.
23-21	Reserved	0	Reserved
20-16	WM	0-1Fh	Watermark for maximum queue usage. Watermark tracks the most entries that have been in queue <i>n</i> since reset or since the last time that the watermark (WM) bit was cleared. WM is cleared by writing a 1 to the corresponding QTHRXC <small>D</small> <sub><i>n</i></sub> bit in the EDMA3CC error clear register (CCERRCLR).
		0-10h	Legal values are 0 (empty) to 10h (full).
		11h-1Fh	Reserved
15-13	Reserved	0	Reserved
12-8	NUMVAL	0-1Fh	Number of valid entries in queue <i>n</i> . The total number of entries residing in the queue manager FIFO at a given instant. Always enabled.
		0-10h	Legal values are 0 (empty) to 10h (full).
		11h-1Fh	Reserved
7-4	Reserved	0	Reserved
3-0	STRTP <small>TR</small>	0-Fh	Start pointer. The offset to the head entry of queue <i>n</i> , in units of entries. Always enabled. Legal values are 0 (0th entry) to Fh (15th entry).

### 15.4.2.4.3 Queue Watermark Threshold A Register (QWMTHRA)

The queue watermark threshold A register (QWMTHRA) is shown in [Figure 15-59](#) and described in [Table 15-41](#).

**Figure 15-59. Queue Watermark Threshold A Register (QWMTHRA)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-41. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-8	Q1	0-1Fh	Queue threshold for queue 1 value. The QTHRCD1 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 1 (QSTAT1) are set when the number of events in queue 1 at an instant in time (visible via the NUMVAL bit in QSTAT1) equals or exceeds the value specified by Q1.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved
7-5	Reserved	0	Reserved
4-0	Q0	0-1Fh	Queue threshold for queue 0 value. The QTHRCD0 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 0 (QSTAT0) are set when the number of events in queue 0 at an instant in time (visible via the NUMVAL bit in QSTAT0) equals or exceeds the value specified by Q0.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved

#### 15.4.2.4.4 EDMA3CC Status Register (CCSTAT)

The EDMA3CC status register (CCSTAT) has a number of status bits that reflect which parts of the EDMA3CC logic is active at any given instant of time. The CCSTAT is shown in [Figure 15-60](#) and described in [Table 15-42](#).

**Figure 15-60. EDMA3CC Status Register (CCSTAT)**

31	Reserved	24
	R-0	
23	Reserved	18      17      16
	R-0	R-0      R-0
15	Reserved	13      8
	R-0	COMPACTV R-0
7	Reserved	5      4      3      2      1      0
	R-0	ACTV      WSTATACTV      TRACTV      QEV TACTV      EVTACTV R-0      R-0      R-0      R-0      R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-42. EDMA3CC Status Register (CCSTAT) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	QUEACTV1	0 1	Queue 1 active. No events are queued in queue 1. At least one TR is queued in queue 1.
16	QUEACTV0	0 1	Queue 0 active. No events are queued in queue 0. At least one TR is queued in queue 0.
15-14	Reserved	0	Reserved
13-8	COMPACTV	0-3Fh  0 1h-3Fh	Completion request active. The COMPACTV field reflects the count for the number of completion requests submitted to the transfer controllers. This count increments every time a TR is submitted and is programmed to report completion (the TCINTEN or TCCCHEN bits in OPT in the parameter entry associated with the TR are set to 1). The counter decrements for every valid TCC received back from the transfer controllers. If at any time the count reaches a value of 63, the EDMA3CC will not service any new TRs until the count is less than 63 (or return a transfer completion code from a transfer controller, which would decrement the count). No completion requests outstanding. Total of 1 completion request to 63 completion requests are outstanding.
7-5	Reserved	0	Reserved
4	ACTV	0 1	Channel controller active. Channel controller active is a logical-OR of each of the *ACTV bits. The ACTV bit remains high through the life of a TR. Channel is idle. Channel is busy.
3	WSTATACTV	0 1	Write status interface active. Write status req is idle and write status fifo is idle. Either the write status request is active or additional write status responses are pending in the write status fifo.
2	TRACTV	0 1	Transfer request active. Transfer request processing/submission logic is inactive. Transfer request processing/submission logic is active.

**Table 15-42. EDMA3CC Status Register (CCSTAT) Field Descriptions (continued)**

Bit	Field	Value	Description
1	QEVTACTV	0	QDMA event active. No enabled QDMA events are active within the EDMA3CC.
		1	At least one enabled QDMA event (QER) is active within the EDMA3CC.
0	EVTACTV	0	DMA event active. No enabled DMA events are active within the EDMA3CC.
		1	At least one enabled DMA event (ER and EER, ESR, CER) is active within the EDMA3CC.

### 15.4.2.5 DMA Channel Registers

The following registers pertain to the 32 DMA channels. The 32 DMA channels consist of registers (with the exception of DMAQNUM $n$ ) that each have 32 bits and the bit position of each register matches the DMA channel number.

The DMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region is controlled by the DMA region access registers (DRAE $m$ ). These registers are described in [Section 15.4.2.3.1](#) and the details for shadow region/global region usage is explained in [Section 15.2.7](#).

#### 15.4.2.5.1 Event Register (ER)

All external events are captured in the event register (ER). The events are latched even when the events are not enabled. If the event bit corresponding to the latched event is enabled (EER.E $n$  = 1), then the event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. The event register bits are automatically cleared (ER.E $n$  = 0) once the corresponding events are prioritized and serviced. If ER.E $n$  are already set and another event is received on the same channel/event, then the corresponding event is latched in the event miss register (EMR.E $n$ ), provided that the event was enabled (EER.E $n$  = 1).

Event  $n$  can be cleared by the CPU writing a 1 to corresponding event bit in the event clear register (ECR). The setting of an event is a higher priority relative to clear operations (via hardware or software). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The ER is shown in [Figure 15-61](#) and described in [Table 15-43](#).

**Figure 15-61. Event Register (ER)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; - $n$  = value after reset

**Table 15-43. Event Register (ER) Field Descriptions**

Bit	Field	Value	Description
31-0	E $n$	0	Event 0-31. Events 0-31 are captured by the EDMA3CC and are latched into ER. The events are set (E $n$ = 1) even when events are disabled (E $n$ = 0 in the event enable register, EER). EDMA3CC event is not asserted.
		1	EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.



### 15.4.2.5.2 Event Clear Register (ECR)

Once an event has been posted in the event register (ER), the event is cleared in two ways. If the event is enabled in the event enable register (EER) and the EDMA3CC submits a transfer request for the event to the EDMA3TC, it clears the corresponding event bit in the event register. If the event is disabled in the event enable register (EER), the CPU can clear the event by way of the event clear register (ECR).

Writing a 1 to any of the bits clears the corresponding event; writing a 0 has no effect. Once an event bit is set in the event register, it remains set until EDMA3CC submits a transfer request for that event or the CPU clears the event by setting the corresponding bit in ECR.

The ECR is shown in [Figure 15-62](#) and described in [Table 15-44](#).

**Figure 15-62. Event Clear Register (ECR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-44. Event Clear Register (ECR) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>	0	Event clear for event 0-31. Any of the event bits in ECR is set to 1 to clear the event ( <i>En</i> ) in the event register (ER). A write of 0 has no effect.
		1	No effect.
			EDMA3CC event is cleared in the event register (ER).

### 15.4.2.5.3 Event Set Register (ESR)

The event set register (ESR) allows the CPU (or EDMA programmers) to manually set events to initiate DMA transfer requests. CPU writes of 1 to any event set register ( $E_n$ ) bits set the corresponding bits in the registers. The set event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. Writing a 0 has no effect.

The event set register operates independent of the event register (ER), and a write of 1 is always considered a valid event regardless of whether the event is enabled (the corresponding event bits are set or cleared in  $EER.E_n$ ).

Once the event is set in the event set register, it cannot be cleared by CPU writes, in other words, the event clear register (ECR) has no effect on the state of ESR. The bits will only be cleared once the transfer request corresponding to the event has been submitted to the transfer controller. The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Manually-triggered transfers via writes to ESR allow the CPU to submit DMA requests in the system, these are relevant for memory-to-memory transfer scenarios. If the  $ESR.E_n$  bit is already set and another CPU write of 1 is attempted to the same bit, then the corresponding event is latched in the event missed registers ( $EMR.E_n = 1$ ).

The ESR is shown in [Figure 15-63](#) and described in [Table 15-45](#).

**Figure 15-63. Event Set Register (ESR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-45. Event Set Register (ESR) Field Descriptions**

Bit	Field	Value	Description
31-0	$E_n$	0	Event set for event 0-31. No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

#### 15.4.2.5.4 Chained Event Register (CER)

When the OPTIONS parameter for a PaRAM entry is programmed to returned a chained completion code (ITCCHEN = 1 and/or TCCHEN = 1), then the value dictated by the TCC[5:0] (also programmed in OPT) forces the corresponding event bit to be set in the chained event register (CER). The set chained event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. This results in a chained-triggered transfer.

The chained event registers do not have any enables. The generation of a chained event is essentially enabled by the PaRAM entry that has been configured for intermediate and/or final chaining on transfer completion. The *En* bit is set (regardless of the state of EER.En) when a chained completion code is returned from one of the transfer controllers or is generated by the EDMA3CC via the early completion path. The bits in the chained event register are cleared when the corresponding events are prioritized and serviced.

If the *En* bit is already set and another chaining completion code is return for the same event, then the corresponding event is latched in the event missed register (EMR.En = 1). The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The CER is shown in [Figure 15-64](#) and described in [Table 15-46](#).

**Figure 15-64. Chained Event Register (CER)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-46. Chained Event Register (CER) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>	0	Chained event for event 0-31. No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

### 15.4.2.5.5 Event Enable Register (EER)

The EDMA3CC provides the option of selectively enabling/disabling each event in the event register (ER) by using the event enable register (EER). If an event bit in EER is set to 1 (using the event enable set register, EESR), it will enable that corresponding event. Alternatively, if an event bit in EER is cleared (using the event enable clear register, EECR), it will disable the corresponding event.

The event register latches all events that are captured by EDMA3CC, even if the events are disabled (although EDMA3CC does not process it). Enabling an event with a pending event already set in the event register enables the EDMA3CC to process the already set event like any other new event. The EER settings do not have any effect on chained events ( $CER.En = 1$ ) and manually set events ( $ESR.En = 1$ ).

The EER is shown in [Figure 15-65](#) and described in [Table 15-47](#).

**Figure 15-65. Event Enable Register (EER)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-47. Event Enable Register (EER) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>		Event enable for events 0-31.
		0	Event is not enabled. An external event latched in the event register (ER) is not evaluated by the EDMA3CC.
		1	Event is enabled. An external event latched in the event register (ER) is evaluated by the EDMA3CC.

### 15.4.2.5.6 Event Enable Clear Register (EECR)

The event enable register (EER) cannot be modified by directly writing to it. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable clear register (EECR) is used to disable events. Writes of 1 to the bits in EECR clear the corresponding event bits in EER; writes of 0 have no effect.

The EECR is shown in [Figure 15-66](#) and described in [Table 15-48](#).

**Figure 15-66. Event Enable Clear Register (EECR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-48. Event Enable Clear Register (EECR) Field Descriptions**

Bit	Field	Value	Description
31-0	$En$	0	Event enable clear for events 0-31. No effect.
		1	Event is disabled. Corresponding bit in the event enable register (EER) is cleared ( $En = 0$ ).

### 15.4.2.5.7 Event Enable Set Register (EESR)

The event enable register (EER) cannot be modified by directly writing to it. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable set register (EESR) is used to enable events. Writes of 1 to the bits in EESR set the corresponding event bits in EER; writes of 0 have no effect.

The EESR is shown in [Figure 15-67](#) and described in [Table 15-49](#).

**Figure 15-67. Event Enable Set Register (EESR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-49. Event Enable Set Register (EESR) Field Descriptions**

Bit	Field	Value	Description
31-0	$En$	0	Event enable set for events 0-31. No effect.
		1	Event is enabled. Corresponding bit in the event enable register (EER) is set ( $En = 1$ ).

### 15.4.2.5.8 Secondary Event Register (SER)

The secondary event register (SER) provides information on the state of a DMA channel or event (0 through 31). If the EDMA3CC receives a TR synchronization due to a manual-trigger, event-trigger, or chained-trigger source (ESR.En = 1, ER.En = 1, or CER.En = 1), which results in the setting of a corresponding event bit in SER (SER.En = 1), it implies that the corresponding DMA event is in the queue.

Once a bit corresponding to an event is set in SER, the EDMA3CC does not prioritize additional events on the same DMA channel. Depending on the condition that leads to the setting of the SER bits, either the EDMA3CC hardware or the software (using SECR) needs to clear the SER bits for the EDMA3CC to evaluate subsequent events and perform subsequent transfers on the same channel. Based on whether the associated TR is valid, or it is a null or dummy TR, the implications on the state of SER and the required user action in order to submit another DMA transfer might be different.

The SER is shown in [Figure 15-68](#) and described in [Table 15-50](#).

**Figure 15-68. Secondary Event Register (SER)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-50. Secondary Event Register (SER) Field Descriptions**

Bit	Field	Value	Description
31-0	En		Secondary event register. The secondary event register is used to provide information on the state of an event.
		0	Event is not currently stored in the event queue.
		1	Event is currently stored in the event queue. Event arbiter will not prioritize additional events.

### 15.4.2.5.9 Secondary Event Clear Register (SECR)

The secondary event clear register (SECR) clears the status of the secondary event registers (SER). CPU writes of 1 clear the corresponding set bits in SER. Writes of 0 have no effect.

The SECR is shown in [Figure 15-69](#) and described in [Table 15-51](#).

**Figure 15-69. Secondary Event Clear Register (SECR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-51. Secondary Event Clear Register (SECR) Field Descriptions**

Bit	Field	Value	Description
31-0	En		Secondary event clear register
		0	No effect.
		1	Corresponding bit in the secondary event register (SER) is cleared (En = 0).

### 15.4.2.6 Interrupt Registers

All DMA/QDMA channels can be set to assert an EDMA3CC completion interrupt to the CPU on transfer completion, by appropriately configuring the PaRAM entry associated with the channels. The following registers are used for the transfer completion interrupt reporting/generating by the EDMA3CC. See [Section 15.2.9](#) for more details on EDMA3CC completion interrupt generation.

#### 15.4.2.6.1 Interrupt Enable Registers (IER)

Interrupt enable register (IER) is used to enable/disable the transfer completion interrupt generation by the EDMA3CC for all DMA/QDMA channels. The IER cannot be written to directly. To set any interrupt bit in IER, a 1 must be written to the corresponding interrupt bit in the interrupt enable set registers (IESR). Similarly, to clear any interrupt bit in IER, a 1 must be written to the corresponding interrupt bit in the interrupt enable clear register (IECR).

The IER is shown in [Figure 15-70](#) and described in [Table 15-52](#).

**Figure 15-70. Interrupt Enable Register (IER)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 15-52. Interrupt Enable Register (IER) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>		Interrupt enable for channels 0-31.
		0	Interrupt is not enabled.
		1	Interrupt is enabled.

### 15.4.2.6.2 Interrupt Enable Clear Register (IECR)

The interrupt enable clear register (IECR) is used to clear interrupts. Writes of 1 to the bits in IECR clear the corresponding interrupt bits in the interrupt enable registers (IER); writes of 0 have no effect.

The IECR is shown in [Figure 15-71](#) and described in [Table 15-53](#).

**Figure 15-71. Interrupt Enable Clear Register (IECR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-53. Interrupt Enable Clear Register (IECR) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>		Interrupt enable clear for channels 0-31.
		0	No effect
		1	Corresponding bit in the interrupt enable register (IER) is cleared ( $I_n = 0$ ).

### 15.4.2.6.3 Interrupt Enable Set Register (IESR)

The interrupt enable set register (IESR) is used to enable interrupts. Writes of 1 to the bits in IESR set the corresponding interrupt bits in the interrupt enable registers (IER); writes of 0 have no effect.

The IESR is shown in [Figure 15-72](#) and described in [Table 15-54](#).

**Figure 15-72. Interrupt Enable Set Register (IESR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-54. Interrupt Enable Set Register (IESR) Field Descriptions**

Bit	Field	Value	Description
31-0	<i>En</i>		Interrupt enable set for channels 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt enable register (IER) is set ( $I_n = 1$ ).



#### 15.4.2.6.4 Interrupt Pending Register (IPR)

If the TCINTEN and/or ITCINTEN bit in the channel option parameter (OPT) is set to 1 in the PaRAM entry associated with the channel (DMA or QDMA), then the EDMA3TC (for normal completion) or the EDMA3CC (for early completion) returns a completion code on transfer or intermediate transfer completion. The value of the returned completion code is equal to the TCC bit in OPT for the PaRAM entry associated with the channel.

When an interrupt transfer completion code with  $TCC = n$  is detected by the EDMA3CC, then the corresponding bit is set in the interrupt pending register (IPR. $I_n$ , if  $n = 0$  to 31). Note that once a bit is set in the interrupt pending registers, it remains set; it is your responsibility to clear these bits. The bits set in IPR are cleared by writing a 1 to the corresponding bits in the interrupt clear registers (ICR).

The IPR is shown in [Figure 15-73](#) and described in [Table 15-55](#).

**Figure 15-73. Interrupt Pending Register (IPR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; - $n$  = value after reset

**Table 15-55. Interrupt Pending Register (IPR) Field Descriptions**

Bit	Field	Value	Description
31-0	$I_n$	0	Interrupt pending for $TCC = 0$ -31.
		0	Interrupt transfer completion code is not detected or was cleared.
		1	Interrupt transfer completion code is detected ( $I_n = 1$ , $n = EDMA3TC[5:0]$ ).

### 15.4.2.6.5 Interrupt Clear Register (ICR)

The bits in the interrupt pending register (IPR) are cleared by writing a 1 to the corresponding bits in the interrupt clear register (ICR); writes of 0 have no effect. All set bits in IPR must be cleared to allow EDMA3CC to assert additional transfer completion interrupts.

The ICR is shown in [Figure 15-74](#) and described in [Table 15-56](#).

**Figure 15-74. Interrupt Clear Register (ICR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

LEGEND: W = Write only; -n = value after reset

**Table 15-56. Interrupt Clear Register (ICR) Field Descriptions**

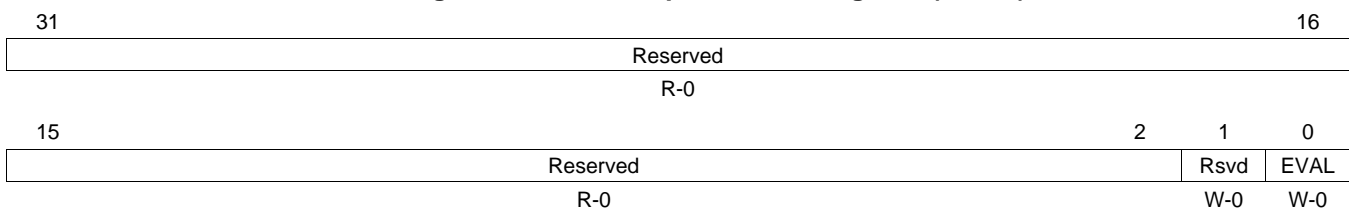
Bit	Field	Value	Description
31-0	<i>In</i>		Interrupt clear register for TCC = 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt pending register (IPR) is cleared ( <i>In</i> = 0).

### 15.4.2.6.6 Interrupt Evaluate Register (IEVAL)

The interrupt evaluate register (IEVAL) is the only register that physically exists in both the global region and the shadow regions. In other words, the read/write accessibility for the shadow region IEVAL is not affected by the DMA/QDMA region access registers (DRAEm and QRAEm). IEVAL is needed for robust ISR operations to ensure that interrupts are not missed by the CPU.

The IEVAL is shown in [Figure 15-75](#) and described in [Table 15-57](#).

**Figure 15-75. Interrupt Evaluate Register (IEVAL)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 15-57. Interrupt Evaluate Register (IEVAL) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL	0 1	Interrupt evaluate. 0 No effect. 1 Causes EDMA3CC completion interrupt to be pulsed, if any enabled (IERn = 1) interrupts are still pending (IPRn = 1). The EDMA3CC completion region interrupt that is pulsed depends on which IEVAL is being exercised. For example, writing to the EVAL bit in IEVAL0 pulses the region 0 completion interrupt, but writing to the EVAL bit in IEVAL1 pulses the region 1 completion interrupt.

### 15.4.2.7 QDMA Channel Registers

The following registers pertain to the 8 QDMA channels. The 8 QDMA channels consist of registers (with the exception of QDMAQNUM) that each have 8 bits and the bit position of each register matches the QDMA channel number.

The QDMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region is controlled by the QDMA region access registers (QRAEm). These registers are described in Section 15.4.2.3.2 and the details for shadow region/global region usage is explained in Section 15.2.7.

#### 15.4.2.7.1 QDMA Event Register (QER)

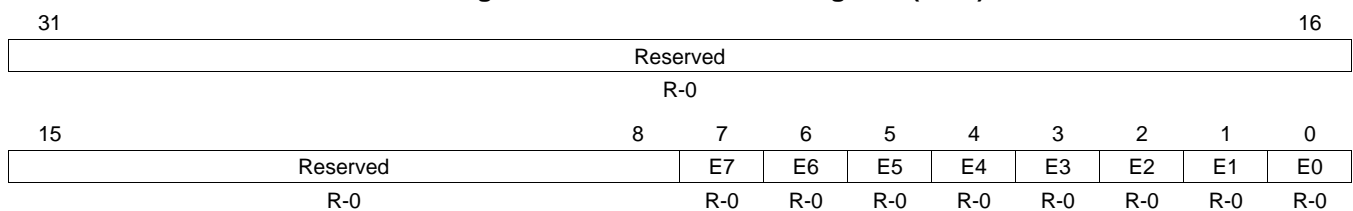
The QDMA event register (QER) channel  $n$  bit is set ( $En = 1$ ) when the CPU or any EDMA programmer (including EDMA3) performs a write to the trigger word (using the QDMA channel  $n$  mapping register (QCHMAP $n$ )) in the PaRAM entry associated with QDMA channel  $n$  (which is also programmed using QCHMAP $n$ ). The  $En$  bit is also set when the EDMA3CC performs a link update on a PaRAM address that matches the QCHMAP $n$  settings. The QDMA event is latched only if the QDMA event enable register (QEER) channel  $n$  bit is also enabled ( $QEER.En = 1$ ). Once a bit is set in QER, then the corresponding QDMA event (auto-trigger) is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers.

The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then the QDMA event missed register (QEMR) would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The set bits in QER are only cleared when the transfer request associated with the corresponding channels has been processed by the EDMA3CC and submitted to the transfer controller. If the  $En$  bit is already set and a QDMA event for the same QDMA channel occurs prior to the original being cleared, then the second missed event is latched in QEMR ( $En = 1$ ).

The QER is shown in Figure 15-76 and described in Table 15-58.

Figure 15-76. QDMA Event Register (QER)



LEGEND: R = Read only; -n = value after reset

Table 15-58. QDMA Event Register (QER) Field Descriptions

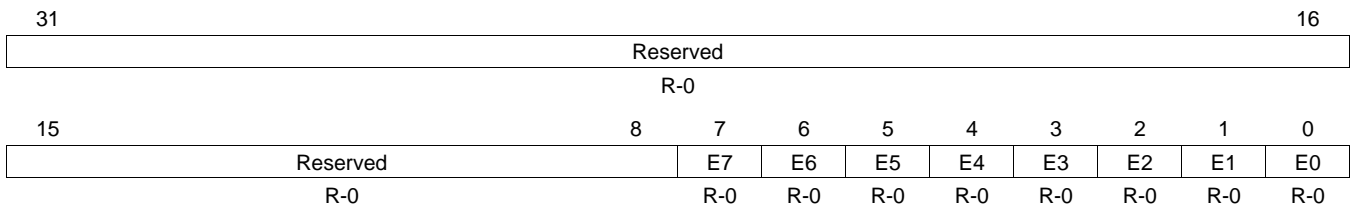
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	$En$	0	QDMA event for channels 0-7. No effect.
		1	Corresponding QDMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

### 15.4.2.7.2 QDMA Event Enable Register (QEER)

The EDMA3CC provides the option of selectively enabling/disabling each channel in the QDMA event register (QER) by using the QDMA event enable register (QEER). If any of the event bits in QEER is set to 1 (using the QDMA event enable set register, QEESR), it will enable that corresponding event. Alternatively, if any event bit in QEER is cleared (using the QDMA event enable clear register, QEECR), it will disable the corresponding QDMA channel. The QDMA event register will not latch any event for a QDMA channel, if it is not enabled via QEER.

The QEER is shown in [Figure 15-77](#) and described in [Table 15-59](#).

**Figure 15-77. QDMA Event Enable Register (QEER)**



LEGEND: R = Read only; -n = value after reset

**Table 15-59. QDMA Event Enable Register (QEER) Field Descriptions**

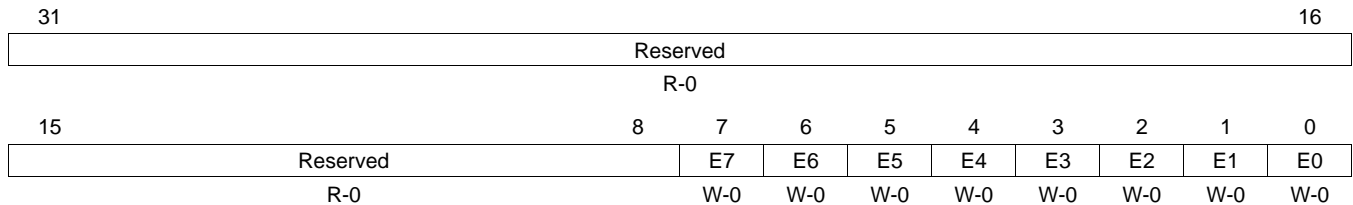
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	$E_n$	0	QDMA event enable for channels 0-7.
		0	QDMA channel $n$ is not enabled. QDMA event will not be recognized and will not latch in the QDMA event register (QER).
		1	QDMA channel $n$ is enabled. QDMA events will be recognized and will get latched in the QDMA event register (QER).

**15.4.2.7.3 QDMA Event Enable Clear Register (QEECR)**

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, in order to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable clear register (QEECR) is used to disable events. Writes of 1 to the bits in QEECR clear the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEECR is shown in [Figure 15-78](#) and described in [Table 15-60](#).

**Figure 15-78. QDMA Event Enable Clear Register (QEECR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 15-60. QDMA Event Enable Clear Register (QEECR) Field Descriptions**

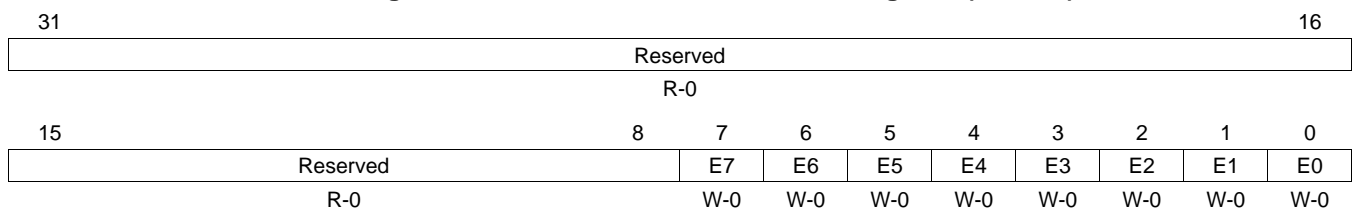
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	En		QDMA event enable clear for channels 0-7.
		0	No effect.
		1	QDMA event is disabled. Corresponding bit in the QDMA event enable register (QEER) is cleared (En = 0).

**15.4.2.7.4 QDMA Event Enable Set Register (QEESR)**

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, in order to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable set register (QEESR) is used to enable events. Writes of 1 to the bits in QEESR set the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEESR is shown in [Figure 15-79](#) and described in [Table 15-61](#).

**Figure 15-79. QDMA Event Enable Set Register (QEESR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 15-61. QDMA Event Enable Set Register (QEESR) Field Descriptions**

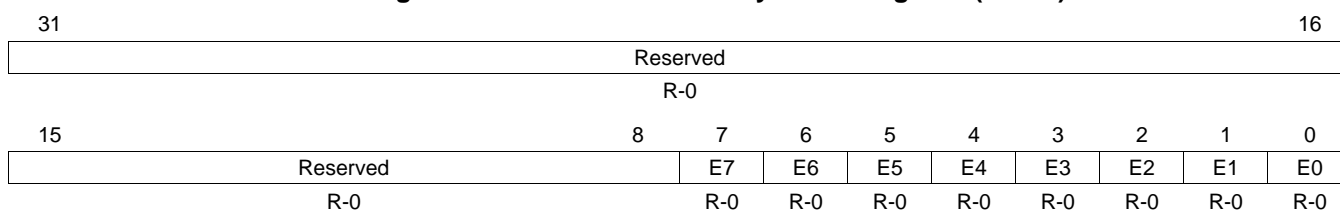
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	En		QDMA event enable set for channels 0-7.
		0	No effect.
		1	QDMA event is enabled. Corresponding bit in the QDMA event enable register (QEER) is set (En = 1).

### 15.4.2.7.5 QDMA Secondary Event Register (QSER)

The QDMA secondary event register (QSER) provides information on the state of a QDMA event. If at any time a bit corresponding to a QDMA channel is set in QSER, that implies that the corresponding QDMA event is in the queue. Once a bit corresponding to a QDMA channel is set in QSER, the EDMA3CC does not prioritize additional events on the same QDMA channel. Depending on the condition that lead to the setting of the QSER bits, either the EDMA3CC hardware or the software (using QSECR) needs to clear the QSER bits for the EDMA3CC to evaluate subsequent QDMA events on the channel. Based on whether the associated TR is valid, or it is a null or dummy TR, the implications on the state of QSER and the required user action in order to submit another QDMA transfer might be different.

The QSER is shown in [Figure 15-80](#) and described in [Table 15-62](#).

**Figure 15-80. QDMA Secondary Event Register (QSER)**



LEGEND: R = Read only; -n = value after reset

**Table 15-62. QDMA Secondary Event Register (QSER) Field Descriptions**

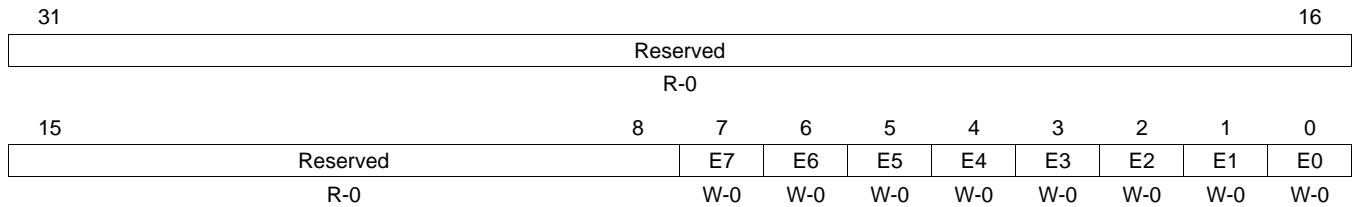
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	$E_n$	0	QDMA secondary event register for channels 0-7. QDMA event is not currently stored in the event queue.
		1	QDMA event is currently stored in event queue. EDMA3CC will not prioritize additional events.

**15.4.2.7.6 QDMA Secondary Event Clear Register (QSECR)**

The QDMA secondary event clear register (QSECR) clears the status of the QDMA secondary event register (QSER) and the QDMA event register (QER). CPU writes of 1 clear the corresponding set bits in QSER and QER. Writes of 0 have no effect. Note that this differs from the secondary event clear register (SECR) operation, which only clears the secondary event register (SER) bits and does not affect the event registers.

The QSECR is shown in [Figure 15-81](#) and described in [Table 15-63](#).

**Figure 15-81. QDMA Secondary Event Clear Register (QSECR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 15-63. QDMA Secondary Event Clear Register (QSECR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	En	0	QDMA secondary event clear register for channels 0-7. No effect.
		1	Corresponding bit in the QDMA secondary event register (QSER) and the QDMA event register (QER) is cleared (En = 0).



### 15.4.3 EDMA3 Transfer Controller (EDMA3TC) Registers

Table 15-64 lists the memory-mapped registers for the EDMA3 transfer controller (EDMA3TC). See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 15-64 should be considered as reserved locations and the register contents should not be modified.

**Table 15-64. EDMA3 Transfer Controller (EDMA3TC) Registers**

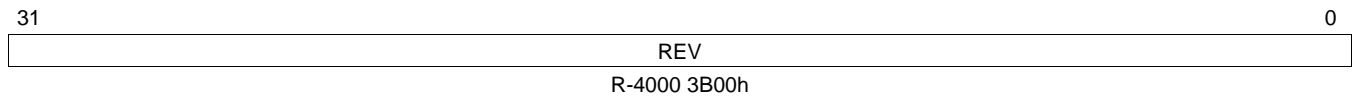
Offset	Acronym	Register Description	Section
0h	REVID	Revision Identification Register	<a href="#">Section 15.4.3.1</a>
4h	TCCFG	EDMA3TC Configuration Register	<a href="#">Section 15.4.3.2</a>
100h	TCSTAT	EDMA3TC Channel Status Register	<a href="#">Section 15.4.3.3</a>
120h	ERRSTAT	Error Status Register	<a href="#">Section 15.4.3.4.1</a>
124h	ERREN	Error Enable Register	<a href="#">Section 15.4.3.4.2</a>
128h	ERRCLR	Error Clear Register	<a href="#">Section 15.4.3.4.3</a>
12Ch	ERRDET	Error Details Register	<a href="#">Section 15.4.3.4.4</a>
130h	ERRCMD	Error Interrupt Command Register	<a href="#">Section 15.4.3.4.5</a>
140h	RDRATE	Read Command Rate Register	<a href="#">Section 15.4.3.5</a>
240h	SAOPT	Source Active Options Register	<a href="#">Section 15.4.3.6.1</a>
244h	SASRC	Source Active Source Address Register	<a href="#">Section 15.4.3.6.2</a>
248h	SACNT	Source Active Count Register	<a href="#">Section 15.4.3.6.3</a>
24Ch	SADST	Source Active Destination Address Register	<a href="#">Section 15.4.3.6.4</a>
250h	SABIDX	Source Active B-Index Register	<a href="#">Section 15.4.3.6.5</a>
254h	SAMPPRXY	Source Active Memory Protection Proxy Register	<a href="#">Section 15.4.3.6.6</a>
258h	SACNTRLD	Source Active Count Reload Register	<a href="#">Section 15.4.3.6.7</a>
25Ch	SASRCBREF	Source Active Source Address B-Reference Register	<a href="#">Section 15.4.3.6.8</a>
260h	SADSTBREF	Source Active Destination Address B-Reference Register	<a href="#">Section 15.4.3.6.9</a>
280h	DFCNTRLD	Destination FIFO Set Count Reload Register	<a href="#">Section 15.4.3.6.10</a>
284h	DFSRCBREF	Destination FIFO Set Source Address B-Reference Register	<a href="#">Section 15.4.3.6.11</a>
288h	DFDSTBREF	Destination FIFO Set Destination Address B-Reference Register	<a href="#">Section 15.4.3.6.12</a>
300h	DFOPT0	Destination FIFO Options Register 0	<a href="#">Section 15.4.3.6.13</a>
304h	DFSRC0	Destination FIFO Source Address Register 0	<a href="#">Section 15.4.3.6.14</a>
308h	DFCNT0	Destination FIFO Count Register 0	<a href="#">Section 15.4.3.6.15</a>
30Ch	DFDST0	Destination FIFO Destination Address Register 0	<a href="#">Section 15.4.3.6.16</a>
310h	DFBIDX0	Destination FIFO B-Index Register 0	<a href="#">Section 15.4.3.6.17</a>
314h	DFMPPRXY0	Destination FIFO Memory Protection Proxy Register 0	<a href="#">Section 15.4.3.6.18</a>
340h	DFOPT1	Destination FIFO Options Register 1	<a href="#">Section 15.4.3.6.13</a>
344h	DFSRC1	Destination FIFO Source Address Register 1	<a href="#">Section 15.4.3.6.14</a>
348h	DFCNT1	Destination FIFO Count Register 1	<a href="#">Section 15.4.3.6.15</a>
34Ch	DFDST1	Destination FIFO Destination Address Register 1	<a href="#">Section 15.4.3.6.16</a>
350h	DFBIDX1	Destination FIFO B-Index Register 1	<a href="#">Section 15.4.3.6.17</a>
354h	DFMPPRXY1	Destination FIFO Memory Protection Proxy Register 1	<a href="#">Section 15.4.3.6.18</a>
380h	DFOPT2	Destination FIFO Options Register 2	<a href="#">Section 15.4.3.6.13</a>
384h	DFSRC2	Destination FIFO Source Address Register 2	<a href="#">Section 15.4.3.6.14</a>
388h	DFCNT2	Destination FIFO Count Register 2	<a href="#">Section 15.4.3.6.15</a>
38Ch	DFDST2	Destination FIFO Destination Address Register 2	<a href="#">Section 15.4.3.6.16</a>
390h	DFBIDX2	Destination FIFO B-Index Register 2	<a href="#">Section 15.4.3.6.17</a>
394h	DFMPPRXY2	Destination FIFO Memory Protection Proxy Register 2	<a href="#">Section 15.4.3.6.18</a>
3C0h	DFOPT3	Destination FIFO Options Register 3	<a href="#">Section 15.4.3.6.13</a>
3C4h	DFSRC3	Destination FIFO Source Address Register 3	<a href="#">Section 15.4.3.6.14</a>
3C8h	DFCNT3	Destination FIFO Count Register 3	<a href="#">Section 15.4.3.6.15</a>

**Table 15-64. EDMA3 Transfer Controller (EDMA3TC) Registers (continued)**

Offset	Acronym	Register Description	Section
3CCh	DFDST3	Destination FIFO Destination Address Register 3	<a href="#">Section 15.4.3.6.16</a>
3D0h	DFBIDX3	Destination FIFO B-Index Register 3	<a href="#">Section 15.4.3.6.17</a>
3D4h	DFMPPRXY3	Destination FIFO Memory Protection Proxy Register 3	<a href="#">Section 15.4.3.6.18</a>

### 15.4.3.1 Revision Identification Register (REVID)

The revision identification register (REVID) is a constant register that uniquely identifies the EDMA3TC and specific revision of the EDMA3TC. The REVID is shown in [Figure 15-82](#) and described in [Table 15-65](#).

**Figure 15-82. Revision ID Register (REVID)**


LEGEND: R = Read only; -n = value after reset

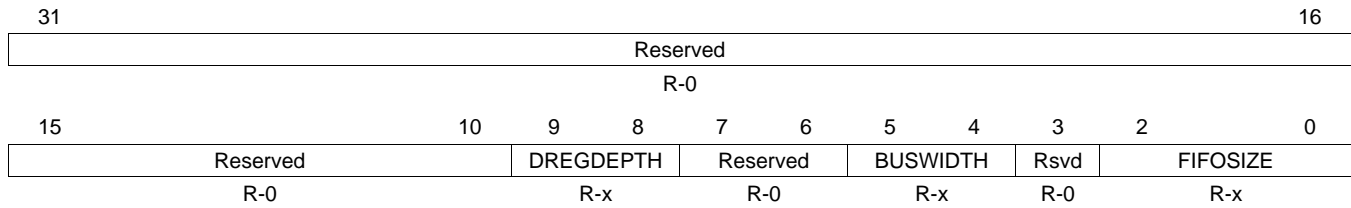
**Table 15-65. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4000 3B00h	Peripheral identifier. Uniquely identifies the EDMA3TC and the specific revision of the EDMA3TC.

### 15.4.3.2 EDMA3TC Configuration Register (TCCFG)

The EDMA3TC configuration register (TCCFG) is shown in [Figure 15-83](#) and described in [Table 15-66](#).

**Figure 15-83. EDMA3TC Configuration Register (TCCFG)**



LEGEND: R = Read only; -n = value after reset; -x = value is indeterminate after reset

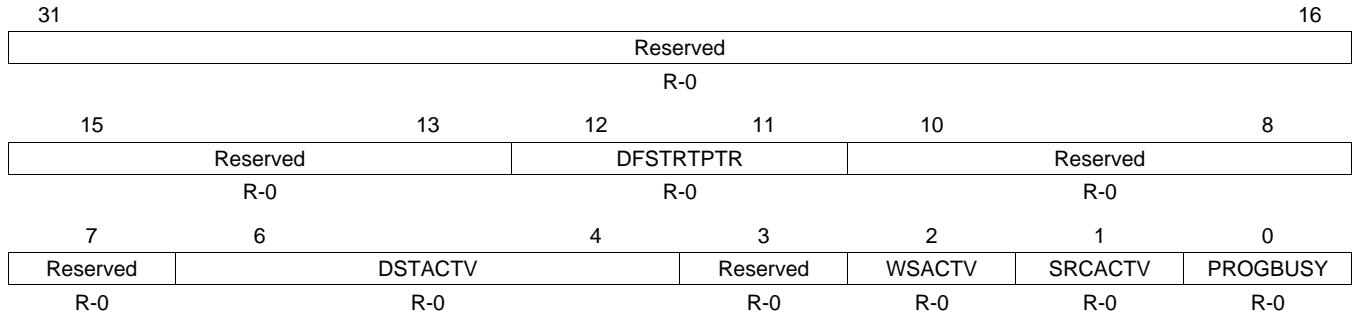
**Table 15-66. EDMA3TC Configuration Register (TCCFG) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-8	DREGDEPTH	0-3h	Destination register FIFO depth parameterization.
		0	1 entry
		1h	2 entry
		2h	4 entry (for EDMA3TC0 and EDMA3TC1)
		3h	Reserved
7-6	Reserved	0	Reserved
5-4	BUSWIDTH	0-3h	Bus width parameterization.
		0	32-bit
		1h	64-bit (for EDMA3TC0 and EDMA3TC1)
		2h-3h	Reserved
3	Reserved	0	Reserved
2-0	FIFOSIZE	0-7h	FIFO size.
		0	32-byte FIFO
		1h	64-byte FIFO
		2h	128-byte FIFO (for EDMA3TC0 and EDMA3TC1)
		3h	256-byte FIFO
4h-7h	Reserved		

### 15.4.3.3 EDMA3TC Channel Status Register (TCSTAT)

The EDMA3TC channel status register (TCSTAT) is shown in [Figure 15-84](#) and described in [Table 15-67](#).

**Figure 15-84. EDMA3TC Channel Status Register (TCSTAT)**



LEGEND: R = Read only; -n = value after reset

**Table 15-67. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions**

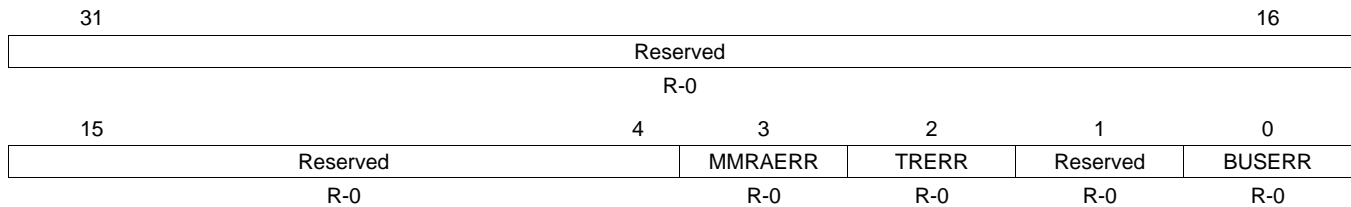
Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-11	DFSTRPTR	0-3h	Destination FIFO start pointer. The offset to the head entry of the destination register FIFO, in units of *entries*.
10-7	Reserved	0	Reserved
6-4	DSTACTV	0-7h	<p>Destination active state. Specifies the number of transfer requests (TRs) that are resident in the destination register FIFO at a given instant. This bit field can be primarily used for advanced debugging.</p> <p>0 Destination FIFO is empty.</p> <p>1h Destination FIFO contains 1 TR.</p> <p>2h Destination FIFO contains 2 TR.</p> <p>3h Destination FIFO contains 3 TR.</p> <p>4h Destination FIFO contains 4 TR. (Full if DSTREGDEPTH == 4)</p> <p>If the destination register FIFO is empty, then any TR written to Prog Set immediately transitions to the destination register FIFO. If the destination register FIFO is not empty and not full, then any TR written to Prog Set immediately transitions to the destination register FIFO set if the source active state (SRCACTV) bit is set to idle.</p> <p>If the destination register FIFO is full, then TRs cannot transition to the destination register FIFO. The destination register FIFO becomes not full when the TR at the head of the destination register FIFO is completed.</p>
5h-7h		Reserved	Reserved
3	Reserved	0	Reserved
2	WSACTV	0	Write status active.
		1	Write status is pending. Write status has not been received for all previously issued write commands.
1	SRCACTV	0	Source active state.
		1	Source active set is busy servicing a transfer request.
0	PROGBUSY	0	Program register set busy.
		1	Program set idle and is available for programming by the EDMA3CC.

### 15.4.3.4 Error Registers

#### 15.4.3.4.1 Error Status Register (ERRSTAT)

The error status register (ERRSTAT) is shown in [Figure 15-85](#) and described in [Table 15-68](#).

**Figure 15-85. Error Status Register (ERRSTAT)**



LEGEND: R = Read only; -n = value after reset

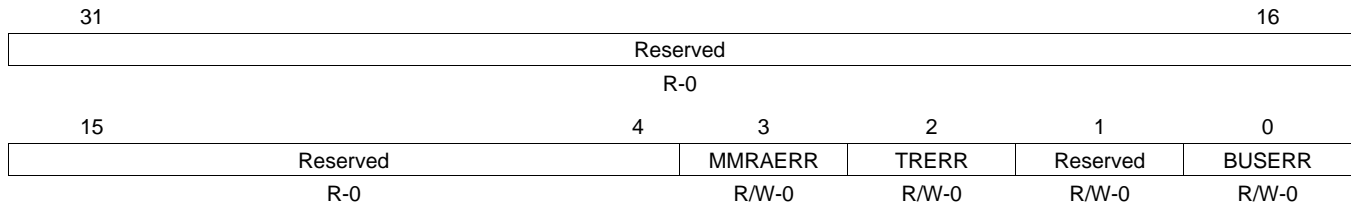
**Table 15-68. Error Status Register (ERRSTAT) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MMRAERR	0	MMR address error.
		1	MMR address error is not detected.
		1	User attempted to read or write to an invalid address in configuration memory map.
2	TRERR	0	Transfer request (TR) error event.
		1	Transfer request (TR) error is not detected.
		1	Transfer request (TR) detected that violates constant addressing mode transfer (SAM or DAM is set to 1) alignment rules or has ACNT or BCNT == 0.
1	Reserved	0	Reserved
0	BUSERR	0	Bus error event.
		1	Bus error is not detected.
		1	EDMA3TC has detected an error at source or destination address. Error information can be read from the error details register (ERRDET).

#### 15.4.3.4.2 Error Enable Register (ERREN)

The error enable register (ERREN) is shown in [Figure 15-86](#) and described in [Table 15-69](#). When any of the enable bits in ERREN is set, a bit set in the corresponding error status register (ERRSTAT) causes an assertion of the EDMA3TC interrupt.

**Figure 15-86. Error Enable Register (ERREN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

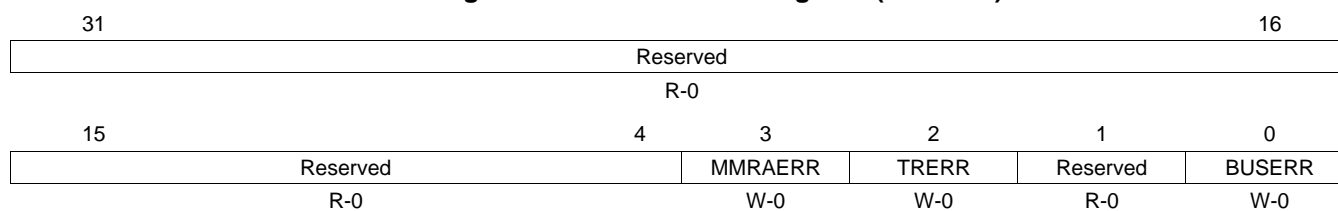
**Table 15-69. Error Enable Register (ERREN) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MMRAERR	0	Interrupt enable for MMR address error (MMRAERR). MMRAERR is disabled.
		1	MMRAERR is enabled and contributes to the state of EDMA3TC error interrupt generation
2	TRERR	0	Interrupt enable for transfer request error (TRERR). TRERR is disabled.
		1	TRERR is enabled and contributes to the state of EDMA3TC error interrupt generation.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	BUSERR	0	Interrupt enable for bus error (BUSERR). BUSERR is disabled.
		1	BUSERR is enabled and contributes to the state of EDMA3TC error interrupt generation.

### 15.4.3.4.3 Error Clear Register (ERRCLR)

The error clear register (ERRCLR) is shown in [Figure 15-87](#) and described in [Table 15-70](#).

**Figure 15-87. Error Clear Register (ERRCLR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 15-70. Error Clear Register (ERRCLR) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MMRAERR	0	Interrupt enable clear for the MMR address error (MMRAERR) bit in the error status register (ERRSTAT). No effect.
		1	Clears the MMRAERR bit in the error status register (ERRSTAT) but does not clear the error details register (ERRDET).
2	TRERR	0	Interrupt enable clear for the transfer request error (TRERR) bit in the error status register (ERRSTAT). No effect.
		1	Clears the TRERR bit in the error status register (ERRSTAT) but does not clear the error details register (ERRDET).
1	Reserved	0	Reserved
0	BUSERR	0	Interrupt clear for the bus error (BUSERR) bit in the error status register (ERRSTAT). No effect.
		1	Clears the BUSERR bit in the error status register (ERRSTAT) and clears the error details register (ERRDET).

#### 15.4.3.4.4 Error Details Register (ERRDET)

The error details register (ERRDET) is shown in [Figure 15-88](#) and described in [Table 15-71](#).

**Figure 15-88. Error Details Register (ERRDET)**

31	Reserved										18	17	16
										TCCHEN		TCINTEN	
R-0										R-0		R-0	
15	14	13					8	7	4	3	0		
Reserved			TCC				Reserved			STAT			
R-0			R-0				R-0			R-0			

LEGEND: R = Read only; -n = value after reset

**Table 15-71. Error Details Register (ERRDET) Field Descriptions**

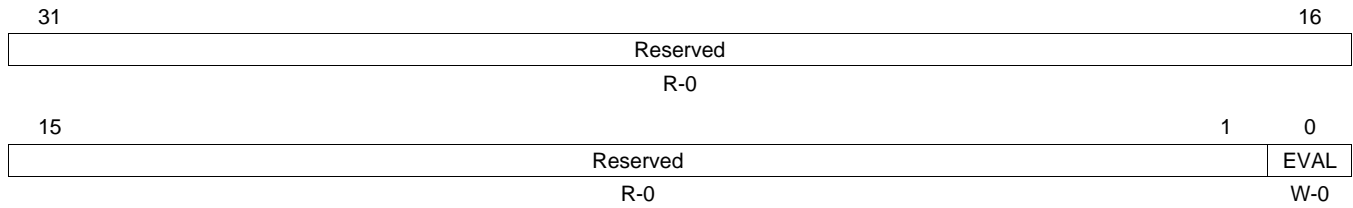
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
17	TCCHEN	0-1	Transfer completion chaining enable. Contains the TCCHEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
16	TCINTEN	0-1	Transfer completion interrupt enable. Contains the TCINTEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
15-14	Reserved	0	Reserved
13 - 8	TCC	0-3Fh	Transfer complete code. Contains the TCC value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
7-4	Reserved	0	Reserved
3-0	STAT	0-Fh	Transaction status. Stores the nonzero status/error code that was detected on the read status or write status bus. If read status and write status are returned on the same cycle, then the EDMA3TC chooses nonzero version. If both are nonzero, then the write status is treated as higher priority.
		0	No error
		1h	Read addressing error
		2h	Read privilege error
		3h	Read timeout error
		4h	Read data error
		5h-6h	Reserved
		7h	Read exclusive operation error
		8h	Reserved
		9h	Write addressing error
		Ah	Write privilege error
		Bh	Write timeout error
		Ch	Write data error
		Dh-Eh	Reserved
		Fh	Write exclusive operation error



#### 15.4.3.4.5 Error Interrupt Command Register (ERRCMD)

The error interrupt command register (ERRCMD) is shown in [Figure 15-89](#) and described in [Table 15-72](#).

**Figure 15-89. Error Interrupt Command Register (ERRCMD)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 15-72. Error Interrupt Command Register (ERRCMD) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EVAL	0	Error evaluate. No effect.
		1	EDMA3TC error line is pulsed if any of the error status register (ERRSTAT) bits are set to 1.

**15.4.3.5 Read Command Rate Register (RDRATE)**

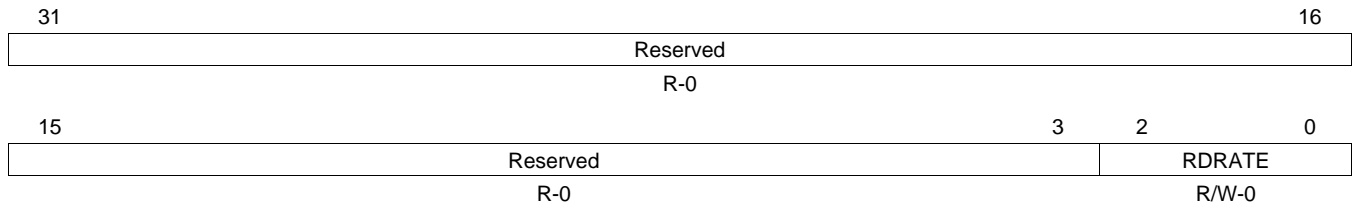
The EDMA3 transfer controller issues Read commands at a rate controlled by the Read command rate register (RDRATE). The RDRATE defines the number of idle cycles that the Read controller must wait before issuing subsequent commands. This applies both to commands within a transfer request packet (TRP) and for commands that are issued for different transfer requests (TRs). For instance, if RDRATE is set to 4 cycles between reads, there are 32 inactive cycles between reads.

RDRATE allows flexibility in transfer controller access requests to an endpoint. For an application, RDRATE can be manipulated to slow down the access rate, so that the endpoint may service requests from other masters during the inactive EDMA3TC cycles.

The RDRATE is shown in [Figure 15-90](#) and described in [Table 15-73](#).

**NOTE:** It is expected that the RDRATE value for a transfer controller is static, as it is decided based on the application requirement. It is not recommended to change this setting on the go.

**Figure 15-90. Read Command Rate Register (RDRATE)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-73. Read Command Rate Register (RDRATE) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	RDRATE	0-7h	Read rate. Controls the number of cycles between Read commands. This is a global setting that applies to all TRs for this EDMA3TC.
		0	Reads issued as fast as possible.
		1h	4 EDMA3TC cycles between reads.
		2h	8 EDMA3TC cycles between reads.
		3h	16 EDMA3TC cycles between reads.
		4h	32 EDMA3TC cycles between reads.
		5h-7h	Reserved

### 15.4.3.6 EDMA3TC Channel Registers

The EDMA3TC channel registers are split into three parts: the programming registers, the source active registers, and the destination FIFO registers. This section describes the registers and their functions. The program register set is programmed by the channel controller and is for internal use. The source active registers and the destination FIFO registers are read-only and are provided to facilitate advanced debug capabilities. The number of destination FIFO register sets depends on the destination FIFO depth. Both TC0 and TC1 have a destination FIFO depth of 4, and there are four sets of destination FIFO registers.

#### 15.4.3.6.1 Source Active Options Register (SAOPT)

The source active options register (SAOPT) is shown in [Figure 15-91](#) and described in [Table 15-74](#).

**Figure 15-91. Source Active Options Register (SAOPT)**

31							23	22	21	20	19	18	17	16	
Reserved						TCCHEN	Rsvd	TCINTEN	Reserved			TCC			
R-0						R/W-0	R-0	R/W-0	R-0			R/W-0			
15			12	11	10	8	7	6			4	3	2	1	0
TCC			Rsvd	FWID			Rsvd	PRI <sup>(1)</sup>			Reserved		DAM	SAM	
R/W-0			R-0	R/W-0			R-0	R/W-0			R-0		R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

- <sup>(1)</sup> On previous architectures, the EDMA3TC priority was controlled by the queue priority register (QUEPRI) in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module. You should use the chip-level registers and not QUEPRI to configure the TC priority.

**Table 15-74. Source Active Options Register (SAOPT) Field Descriptions**

Bit	Field	Value	Description
31-23	Reserved	0	Reserved
22	TCCHEN	0 1	Transfer complete chaining enable. 0 Transfer complete chaining is disabled. 1 Transfer complete chaining is enabled.
21	Reserved	0	Reserved
20	TCINTEN	0 1	Transfer complete interrupt enable. 0 Transfer complete interrupt is disabled. 1 Transfer complete interrupt is enabled.
19-18	Reserved	0	Reserved
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3CC.
11	Reserved	0	Reserved
10-8	FWID	0-7h 0 1h 2h 3h 4h 5h-7h	FIFO width. Applies if either SAM or DAM is set to constant addressing mode. 0 FIFO width is 8 bits. 1h FIFO width is 16 bits. 2h FIFO width is 32 bits. 3h FIFO width is 64 bits. 4h FIFO width is 128 bits. 5h-7h Reserved
7	Reserved	0	Reserved
6-4	PRI	0-7h 0 1h-6h 7h	Transfer priority. Reflects the values programmed in the queue priority register (QUEPRI) in the EDMA3CC. 0 Priority 0 - Highest priority 1h-6h Priority 1 to priority 6 7h Priority 7 - Lowest priority
3-2	Reserved	0	Reserved

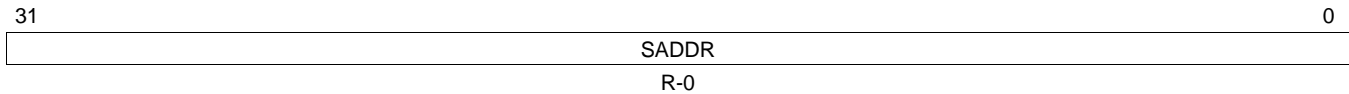
**Table 15-74. Source Active Options Register (SAOPT) Field Descriptions (continued)**

Bit	Field	Value	Description
1	DAM	0	Increment (INCR) mode. Destination addressing within an array increments.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM	0	Increment (INCR) mode. Source addressing within an array increments.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

**15.4.3.6.2 Source Active Source Address Register (SASRC)**

The source active source address register (SASRC) is shown in [Figure 15-92](#) and described in [Table 15-75](#).

**Figure 15-92. Source Active Source Address Register (SASRC)**



LEGEND: R = Read only; -n = value after reset

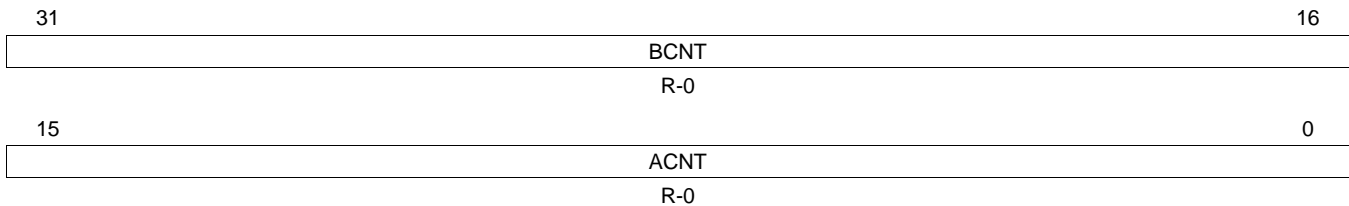
**Table 15-75. Source Active Source Address Register (SASRC) Field Descriptions**

Bit	Field	Value	Description
31-0	SADDR	0-FFFF FFFFh	Source address for program register set. EDMA3TC updates value according to source addressing mode (SAM bit in the source active options register, SAOPT) .

**15.4.3.6.3 Source Active Count Register (SACNT)**

The source active count register (SACNT) is shown in [Figure 15-93](#) and described in [Table 15-76](#).

**Figure 15-93. Source Active Count Register (SACNT)**



LEGEND: R = Read only; -n = value after reset

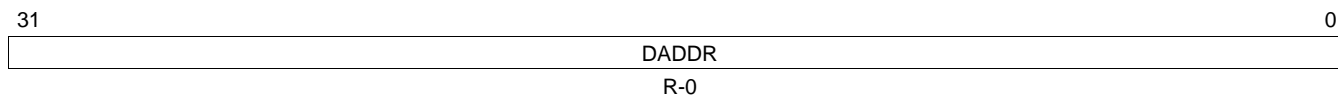
**Table 15-76. Source Active Count Register (SACNT) Field Descriptions**

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B dimension count. Number of arrays to be transferred, where each array is ACNT in length. It is decremented after each Read command appropriately. Represents the amount of data remaining to be Read. It should be 0 when transfer request (TR) is complete.
15-0	ACNT	0-FFFFh	A dimension count. Number of bytes to be transferred in first dimension. It is decremented after each Read command appropriately. Represents the amount of data remaining to be Read. It should be 0 when transfer request (TR) is complete.

#### 15.4.3.6.4 Source Active Destination Address Register (SADST)

The source active destination address register (SADST) is shown in [Figure 15-94](#) and described in [Table 15-77](#).

**Figure 15-94. Source Active Destination Address Register (SADST)**



LEGEND: R = Read only; -n = value after reset

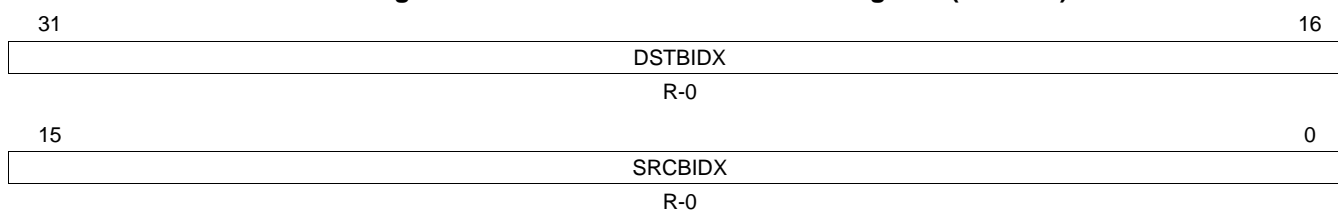
**Table 15-77. Source Active Destination Address Register (SADST) Field Descriptions**

Bit	Field	Value	Description
31-0	DADDR	0	Always reads as 0

#### 15.4.3.6.5 Source Active B-Index Register (SABIDX)

The source active B-index register (SABIDX) is shown in [Figure 15-95](#) and described in [Table 15-78](#).

**Figure 15-95. Source Active B-Index Register (SABIDX)**



LEGEND: R = Read only; -n = value after reset

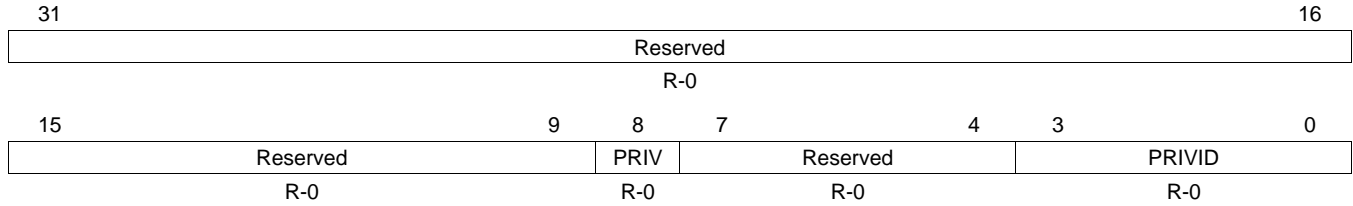
**Table 15-78. Source Active B-Index Register (SABIDX) Field Descriptions**

Bit	Field	Value	Description
31-16	DSTBIDX	0	B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. Always reads as 0.
15-0	SRCBIDX	0-FFFFh	B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array.

### 15.4.3.6.6 Source Active Memory Protection Proxy Register (SAMPPRXY)

The source active memory protection proxy register (SAMPPRXY) is shown in [Figure 15-96](#) and described in [Table 15-79](#).

**Figure 15-96. Source Active Memory Protection Proxy Register (SAMPPRXY)**



LEGEND: R = Read only; -n = value after reset

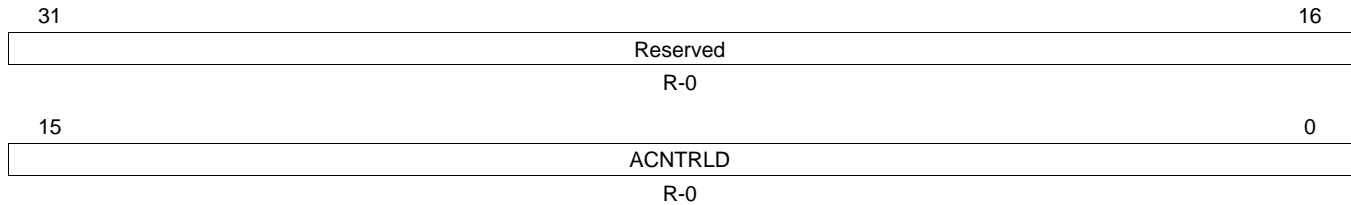
**Table 15-79. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PRIV	0 1	Privilege level. The privilege level used by the host to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.  The privilege ID is used while issuing Read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.  0 User-level privilege 1 Supervisor-level privilege
7-4	Reserved	0	Reserved
3-0	PRIVID	0-Fh 0 1	Privilege ID. This contains the privilege ID of the host that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.  This PRIVID value is used while issuing Read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.  0 For any other master that sets up the PaRAM entry. 1 If DSP sets up the PaRAM entry.

### 15.4.3.6.7 Source Active Count Reload Register (SACNTRLD)

The source active count reload register (SACNTRLD) is shown in [Figure 15-97](#) and described in [Table 15-80](#).

**Figure 15-97. Source Active Count Reload Register (SACNTRLD)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

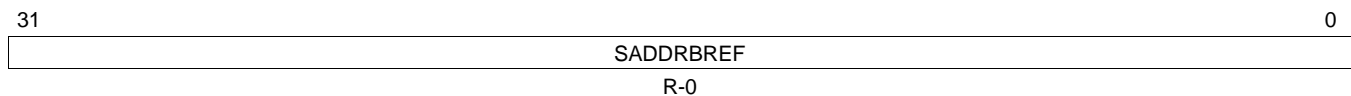
**Table 15-80. Source Active Count Reload Register (SACNTRLD) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	ACNTRLD	0-FFFFh	A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced.

### 15.4.3.6.8 Source Active Source Address B-Reference Register (SASRCBREF)

The source active source address B-reference register (SASRCBREF) is shown in [Figure 15-98](#) and described in [Table 15-81](#).

**Figure 15-98. Source Active Source Address B-Reference Register (SASRCBREF)**



LEGEND: R = Read only; -n = value after reset

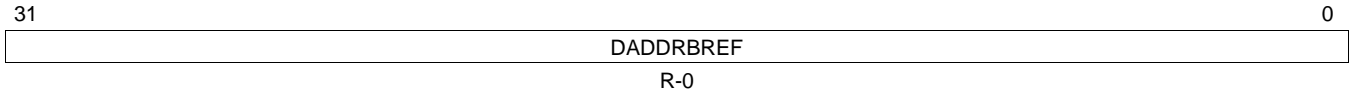
**Table 15-81. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions**

Bit	Field	Value	Description
31-0	SADDRBREF	0-FFFF FFFFh	Source address B-reference. Represents the starting address for the array currently being Read.

### 15.4.3.6.9 Source Active Destination Address B-Reference Register (SADSTBREF)

The source active destination address B-reference register (SADSTBREF) is shown in [Figure 15-99](#) and described in [Table 15-82](#).

**Figure 15-99. Source Active Destination Address B-Reference Register (SADSTBREF)**



LEGEND: R = Read only; -n = value after reset

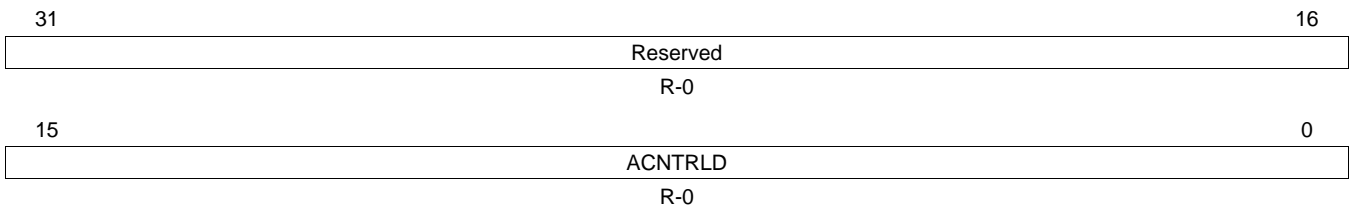
**Table 15-82. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions**

Bit	Field	Value	Description
31-0	DADDRBREF	0	Always reads as 0

### 15.4.3.6.10 Destination FIFO Set Count Reload Register (DFCNTRLD)

The destination FIFO set count reload register (DFCNTRLD) is shown in [Figure 15-100](#) and described in [Table 15-83](#).

**Figure 15-100. Destination FIFO Set Count Reload Register (DFCNTRLD)**



LEGEND: R = Read only; -n = value after reset

**Table 15-83. Destination FIFO Set Count Reload Register (DFCNTRLD) Field Descriptions**

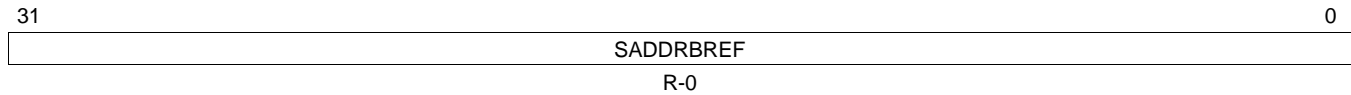
Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	ACNTRLD	0-FFFFh	A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced.



### 15.4.3.6.11 Destination FIFO Set Source Address B-Reference Register (DFSRCBREF)

The destination FIFO set source address B-reference register (DFSRCBREF) is shown in [Figure 15-101](#) and described in [Table 15-84](#).

**Figure 15-101. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF)**



LEGEND: R = Read only; -n = value after reset

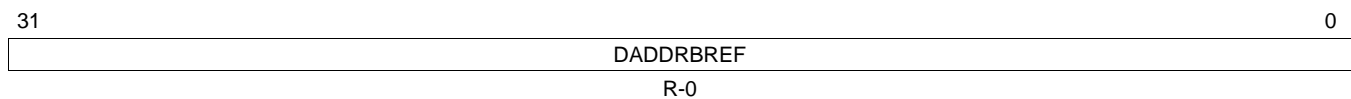
**Table 15-84. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF) Field Descriptions**

Bit	Field	Value	Description
31-0	SADDRBREF	0	Not applicable. Always Read as 0.

### 15.4.3.6.12 Destination FIFO Set Destination Address B-Reference (DFDSTBREF)

The destination FIFO set destination address B-reference register (DFDSTBREF) is shown in [Figure 15-102](#) and described in [Table 15-85](#).

**Figure 15-102. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF)**



LEGEND: R = Read only; -n = value after reset

**Table 15-85. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) Field Descriptions**

Bit	Field	Value	Description
31-0	DADDRBREF	0-FFFF FFFFh	Destination address reference for the destination FIFO register set. Represents the starting address for the array currently being written.

### 15.4.3.6.13 Destination FIFO Options Register $n$ (DFOPT $n$ )

The destination FIFO options register  $n$  (DFOPT $n$ ) is shown in [Figure 15-103](#) and described in [Table 15-86](#).

**Figure 15-103. Destination FIFO Options Register  $n$  (DFOPT $n$ )**

31	Reserved						23	22	21	20	19	18	17	16
R-0						TCCHEN	Rsvd	TCINTEN	Reserved		TCC			
R/W-0						R/W-0	R-0	R/W-0	R-0		R/W-0			
15	12	11	10	8	7	6	4		3	2	1	0		
TCC		Rsvd	FWID		Rsvd	PRI		Reserved		DAM	SAM			
R/W-0		R-0	R/W-0		R-0	R/W-0		R-0		R/W-0	R/W-0			

LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

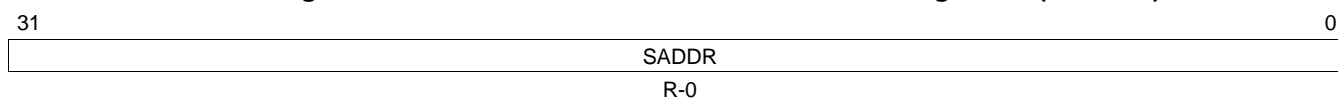
**Table 15-86. Destination FIFO Options Register  $n$  (DFOPT $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-23	Reserved	0	Reserved
22	TCCHEN	0 1	Transfer complete chaining enable. Transfer complete chaining is disabled. Transfer complete chaining is enabled.
21	Reserved	0	Reserved
20	TCINTEN	0 1	Transfer complete interrupt enable. Transfer complete interrupt is disabled. Transfer complete interrupt is enabled.
19-18	Reserved	0	Reserved
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3CC.
11	Reserved	0	Reserved
10-8	FWID	0-7h 0 1h 2h 3h 4h 5h-7h	FIFO width. Applies if either SAM or DAM is set to constant addressing mode. FIFO width is 8 bits. FIFO width is 16 bits. FIFO width is 32 bits. FIFO width is 64 bits. FIFO width is 128 bits. Reserved
7	Reserved	0	Reserved
6-4	PRI	0-7h 0 1h-6h 7h	Transfer priority. Priority 0 - Highest priority Priority 1 to priority 6 Priority 7 - Lowest priority
3-2	Reserved	0	Reserved
1	DAM	0 1	Destination address mode within an array. Increment (INCR) mode. Destination addressing within an array increments. Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM	0 1	Source address mode within an array. Increment (INCR) mode. Source addressing within an array increments. Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

#### 15.4.3.6.14 Destination FIFO Source Address Register $n$ (DFSRC $n$ )

The destination FIFO source address register  $n$  (DFSRC $n$ ) is shown in [Figure 15-104](#) and described in [Table 15-87](#).

**Figure 15-104. Destination FIFO Source Address Register  $n$  (DFSRC $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

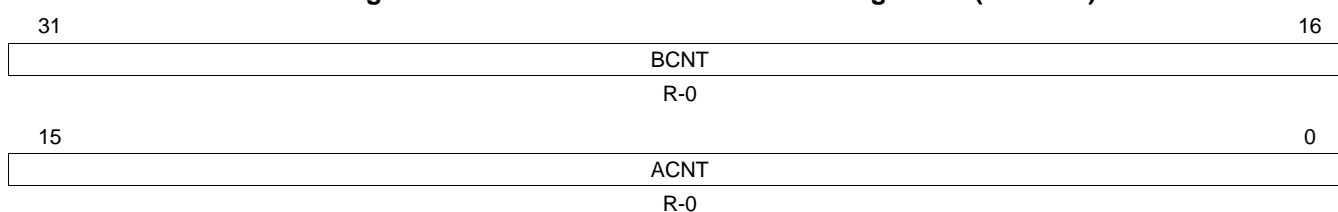
**Table 15-87. Destination FIFO Source Address Register  $n$  (DFSRC $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-0	SADDR	0	Always Read as 0.

#### 15.4.3.6.15 Destination FIFO Count Register $n$ (DFCNT $n$ )

The destination FIFO count register  $n$  (DFCNT $n$ ) is shown in [Figure 15-105](#) and described in [Table 15-88](#).

**Figure 15-105. Destination FIFO Count Register  $n$  (DFCNT $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

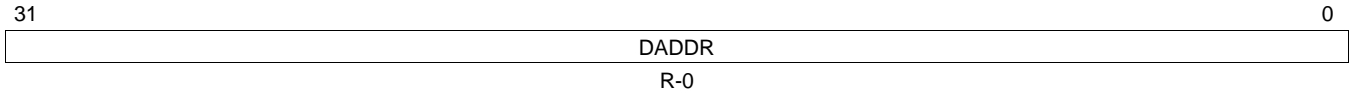
**Table 15-88. Destination FIFO Count Register  $n$  (DFCNT $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B-dimension count. Number of arrays to be transferred, where each array is ACNT in length. Count/count remaining for destination register set. Represents the amount of data remaining to be written.
15-0	ACNT	0-FFFFh	A-dimension count. Number of bytes to be transferred in first dimension count/count remaining for destination register set. Represents the amount of data remaining to be written.

### 15.4.3.6.16 Destination FIFO Destination Address Register $n$ (DFDST $n$ )

The destination FIFO destination address register  $n$  (DFDST $n$ ) is shown in [Figure 15-106](#) and described in [Table 15-89](#).

**Figure 15-106. Destination FIFO Destination Address Register  $n$  (DFDST $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

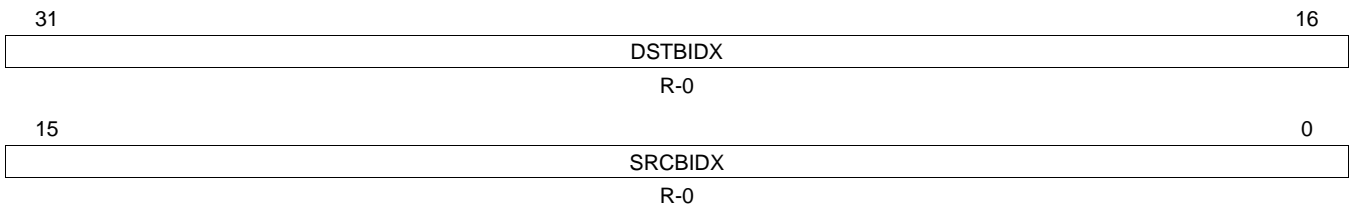
**Table 15-89. Destination FIFO Destination Address Register  $n$  (DFDST $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-0	DADDR	0	Destination address for the destination FIFO register set. When a transfer request (TR) is complete, the final value should be the address of the last write command issued.

### 15.4.3.6.17 Destination FIFO B-Index Register $n$ (DFBIDX $n$ )

The destination FIFO B-index register  $n$  (DFBIDX $n$ ) is shown in [Figure 15-107](#) and described in [Table 15-90](#).

**Figure 15-107. Destination FIFO B-Index Register  $n$  (DFBIDX $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

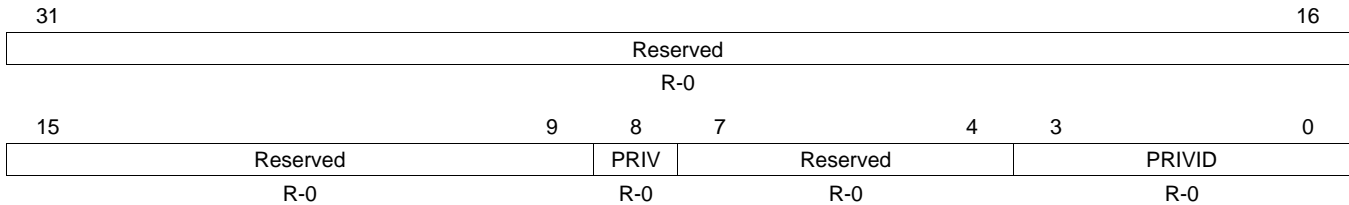
**Table 15-90. Destination FIFO B-Index Register  $n$  (DFBIDX $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-16	DSTBIDX	0-FFFFh	B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination.
15-0	SRCBIDX	0	B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array. Always Read as 0.

### 15.4.3.6.18 Destination FIFO Memory Protection Proxy Register $n$ (DFMPPRXY $n$ )

The destination FIFO memory protection proxy register  $n$  (DFMPPRXY $n$ ) is shown in [Figure 15-108](#) and described in [Table 15-91](#).

**Figure 15-108. Destination FIFO Memory Protection Proxy Register  $n$  (DFMPPRXY $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

**Table 15-91. Destination FIFO Memory Protection Proxy Register  $n$  (DFMPPRXY $n$ )  
Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PRIV	0 1	<p>Privilege level. This contains the privilege level used by the EDMA programmer to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.</p> <p>The privilege ID is used while issuing Read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.</p> <p>0 User-level privilege 1 Supervisor-level privilege</p>
7-4	Reserved	0	Reserved
3-0	PRIVID	0-Fh 0 1	<p>Privilege ID. This contains the Privilege ID of the EDMA programmer that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.</p> <p>This PRIVID value is used while issuing Read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.</p> <p>0 For any other master that sets up the PaRAM entry 1 If DSP sets up the PaRAM entry</p>

## 15.5 Tips

### 15.5.1 Debug Checklist

This section lists some tips to keep in mind while debugging applications using the EDMA3. [Table 15-92](#) provides some common issues and their probable causes and resolutions.

**Table 15-92. Debug List**

Issue	Description/Solution
The transfer associated with the channel does not happen. The channel does not get serviced.	<p>The EDMA3 channel controller (EDMA3CC) may not service a transfer request, even though the associated PaRAM set is programmed appropriately. Check for the following:</p> <ol style="list-style-type: none"> <li>1) Verify that events are enabled, that is, if an external/peripheral event is latched in the event register (ER), make sure that the event is enabled in the event enable register (EER). Similarly for QDMA channels, make sure that QDMA events are appropriately enabled in the QDMA event enable register (QEER).</li> <li>2) Verify that the DMA or QDMA secondary event register (SER) bits corresponding to the particular event or channel are not set.</li> </ol>
The secondary event register bits are set, not allowing additional transfers to occur on a channel.	<p>It is possible that a trigger event was received when the parameter set associated with the channel/event was a NULL set for a previous transfer on the channel. This is typical in two cases:</p> <ol style="list-style-type: none"> <li>1) QDMA channels: Typically if the parameter set is nonstatic and expected to be terminated by a NULL set (OPT.STATIC = 0, LINK = FFFFh), the parameter set is updated with a NULL set after submission of the last TR. Because QDMA channels are autotriggered, this update caused the generation of an event. An event generated for a NULL set causes an error condition and results in setting the bits corresponding to the QDMA channel in QEMR and QSER. This will disable further prioritization of the channel.</li> <li>2) DMA channels used in a continuous mode: The peripheral may be set up to continuously generate infinite events (for instance, in case of the McBSP, every time the data shifts out from DXR, it generates an XEVT). The parameter set may be programmed to expect only a finite number of events and to be terminated by a NULL link. After the expected number of events, the parameter set is reloaded with a NULL parameter set. Because the peripheral will generate additional events, an error condition is set in SER.En and EMR.En, preventing further event prioritization. You must ensure that the number of events received is limited to the expected number of events for which the parameter set is programmed, or you must ensure that bits corresponding to a particular channel or event are not set in the secondary event registers (SER/QSER) and the event missed registers (EMR/QEMR) before trying to perform subsequent transfers for the event/channel.</li> </ol>
Completion interrupts are not asserted, or no further interrupts are received after the first completion interrupt.	<p>You must ensure the following:</p> <ol style="list-style-type: none"> <li>1) The interrupt generation is enabled in the OPT of the associated PaRAM set (TCINTEN = 1 and/or ITCINTEN = 1).</li> <li>2) The interrupts are enabled in the EDMA3 channel controller (EDMA3CC), via the interrupt enable register (IER).</li> <li>3) The corresponding interrupts are enabled in the device interrupt controller.</li> <li>4) The set interrupts are cleared in the interrupt pending register (IPR) before exiting the transfer completion interrupt service routine (ISR). See <a href="#">Section 15.2.9.1.2</a> for details on writing EDMA3 ISRs.</li> <li>5) If working with shadow region interrupts, make sure that the DMA region access enable registers (DRAE) are set up properly, because DRAE act as secondary enables for shadow region completion interrupts, along with IER.</li> </ol> <p>If working with shadow region interrupts, make sure that the bits corresponding to the transfer completion code (TCC) value are also enabled in DRAE. For instance, if the PaRAM set associated with channel 0 returns a completion code of 31 (OPT.TCC = 31), make sure that DRAE.E31 is also set for a shadow region completion interrupt because the interrupt pending register bit set will be IPR.I31.</p>

### 15.5.2 Miscellaneous Programming/Debug Tips

1. For several registers, the setting and clearing of bits needs to be done via separate dedicated registers. For example, the event register (ER) bits can only be cleared by writing a 1 to the corresponding bits in the event clear register (ECR). Similarly, the event enable register (EER) bits can only be set with writes of 1 to the corresponding bits in the event enable set registers (EESR) and can only be cleared with writes of 1 to the corresponding bits in the event enable clear register (EECR).
2. Writes to the shadow region memory maps are governed by region access enable registers (DRAE/QRAE). If the appropriate channels are not enabled in these registers, read/write access to the shadow region memory map is not enabled.
3. When working with shadow region completion interrupts, ensure that the DMA region access enable registers (DRAE) for every region are set in a mutually exclusive way (unless it is a requirement for an application). If there is an overlap in the allocated channels and transfer completion codes (setting of interrupt pending register bits) in the region resource allocation, it results in multiple shadow region completion interrupts. For example, if DRAE0.E0 and DRAE1.E0 are both set, then on completion of a transfer that returns a TCC = 0, they will generate both shadow region 0 and 1 completion interrupts.
4. While programming a non-dummy parameter set, ensure the CCNT is not left to zero.
5. Enable the EDMA3CC error interrupt in the device controller and attach an interrupt service routine (ISR) to ensure that error conditions are not missed in an application and are appropriately addressed with the ISR.
6. Depending on the application, you may want to break large transfers into smaller transfers and use self-chaining to prevent starvation of other events in an event queue.
7. In applications where a large transfer is broken into sets of small transfers using chaining or other methods, you might choose to use the early chaining option to reduce the time between the sets of transfers and increase the throughput. However, keep in mind that with early completion, all data might have not been received at the end point when completion is reported because the EDMA3CC internally signals completion when the TR is submitted to the EDMA3TC, potentially before any data has been transferred.
8. The event queue entries can be observed to determine the last few events if there is a system failure (provided the entries were not bypassed).
9. In order to put the EDMA3CC and EDMA3TC in power-down modes, you should ensure that there is no activity with the EDMA3CC and EDMA3TC. The EDMA3CC status register (CCSTAT) and the EDMA3TC channel status register (TCSTAT) should be used.

## 15.6 Setting Up a Transfer

The following list provides a quick guide for the typical steps involved in setting up a transfer.

1. Initiating a DMA/QDMA channel:
  - (a) Determine the type of channel (QDMA or DMA) to be used.
  - (b) If using a QDMA channel, program the QDMA channel  $n$  mapping register (QCHMAP $n$ ) with the parameter set number to which the channel maps and the trigger word.
  - (c) If the channel is being used in the context of a shadow region, ensure the DMA region access enable register (DRAE) for the region is properly set up to allow read/write accesses to bits in the event register and interrupt register in the shadow region memory-map. The subsequent steps in this process should be done using the respective shadow region registers. (Shadow region descriptions and usage are provided in [Section 15.2.7.1](#).)
  - (d) Determine the type of triggering used.
    - (i) If external events are used for triggering (DMA channels), enable the respective event in EER by writing into EESR.
    - (ii) If a QDMA channel is used, enable the channel in QEER by writing into QEESR.
  - (e) Queue setup.
    - (i) If a QDMA channel is used, set up QDMAQNUM to map the channel to the respective event queue.
    - (ii) If a DMA channel is used, set up DMAQNUM to map the event to the respective event queue.
2. Parameter set setup: Program the PaRAM set number associated with the channel. Note that if it is a QDMA channel, the PaRAM entry that is configured as trigger word is written last. Alternatively, enable the QDMA channel just before the write to the trigger word.  
 See [Section 15.3](#) for parameter set field setups for different types of transfers. See the sections on chaining ([Section 15.2.8](#)) and interrupt completion ([Section 15.2.9](#)) on how to set up final/intermediate completion chaining and/or interrupts.
3. Interrupt setup:
  - (a) If working in the context of a shadow region, ensure the relevant bits in DRAE are set.
  - (b) Enable the interrupt in IER by writing into IESR.
  - (c) Ensure that the EDMA3CC completion interrupt is enabled properly in the device interrupt controller.
  - (d) Set up the interrupt controller properly to receive the expected EDMA3 interrupt.
4. Initiate transfer (this step is highly dependent on the event trigger source):
  - (a) If the source is an external event coming from a peripheral, the peripheral will be enabled to start generating relevant EDMA3 events that can be latched to the ER transfer.
  - (b) For QDMA events, writes to the trigger word will initiate the transfer.
  - (c) Manually-triggered transfers will be initiated by writes to the event set register (ESR).
  - (d) Chained-trigger events initiate when a previous transfer returns a transfer completion code equal to the chained channel number.
5. Wait for completion:
  - (a) If the interrupts are enabled as mentioned in step 3, then the EDMA3CC generates a completion interrupt to the CPU whenever transfer completion results in setting the corresponding bits in the interrupt pending register (IPR). The set bits must be cleared in IPR by writing to the corresponding bit in ICR.
  - (b) If polling for completion (interrupts not enabled in the device controller), then the application code can wait on the expected bits to be set in IPR. Again, the set bits in IPR must be manually cleared by writing to ICR before the next set of transfers is performed for the same transfer completion code values.



## External Memory Interface A (EMIFA)

---

---

This chapter describes the external memory interface A (EMIFA).

The EMIFA SDRAM interface is not supported on all devices, see your device-specific data manual to see if the EMIFA SDRAM is supported on your device.

Topic	Page
<b>16.1 Introduction</b> .....	<b>583</b>
<b>16.2 Architecture</b> .....	<b>583</b>
<b>16.3 Example Configuration</b> .....	<b>624</b>
<b>16.4 Registers</b> .....	<b>646</b>

## 16.1 Introduction

### 16.1.1 Purpose of the Peripheral

EMIFA memory controller is compliant with the JESD21-C SDR SDRAM memories utilizing 16-bit data bus of EMIFA memory controller. The purpose of this EMIFA is to provide a means for the CPU to connect to a variety of external devices including:

- Single data rate (SDR) SDRAM
- Asynchronous devices including NOR Flash, NAND Flash, and SRAM

The most common use for the EMIFA is to interface with both a flash device and an SDRAM device simultaneously. [Section 16.3](#) contains an example of operating the EMIFA in this configuration.

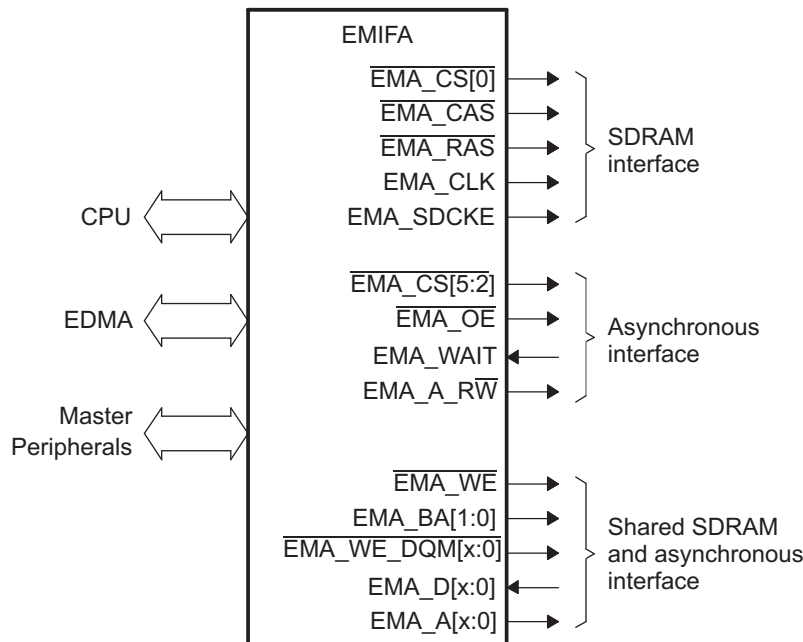
### 16.1.2 Features

The EMIFA includes many features to enhance the ease and flexibility of connecting to external SDR SDRAM and asynchronous devices. For details on features of EMIFA, see your device-specific data manual.

### 16.1.3 Functional Block Diagram

[Figure 16-1](#) illustrates the connections between the EMIFA and its internal requesters, along with the external EMIFA pins. [Section 16.2.2](#) contains a description of the entities internal to the SoC that can send requests to the EMIFA, along with their prioritization. [Section 16.2.3](#) describes the EMIFA external pins and summarizes their purpose when interfacing with SDRAM and asynchronous devices.

**Figure 16-1. EMIFA Functional Block Diagram**



## 16.2 Architecture

This section provides details about the architecture and operation of the EMIFA. Both, SDRAM and asynchronous interface are covered, along with other system-related issues such as clock control and pin multiplexing.

The EMIFA SDRAM interface is not supported on all devices, see your device-specific data manual to see if the EMIFA SDRAM is supported on your device.

### 16.2.1 Clock Control

The EMIFA clock is output on the EMA\_CLK pin and should be used when interfacing to external memories. The EMIFA clock (EMA\_CLK) does not run during device reset. When the  $\overline{\text{RESET}}$  pin is released and after the PLL controller releases the device from reset, EMA\_CLK begins to oscillate at a frequency determined by the PLL controller.

For details on clock generation and control, see the *Device Clocking* chapter.

### 16.2.2 EMIFA Requests

Different sources within the SoC can make requests to the EMIFA. These requests consist of accesses to SDRAM memory, asynchronous memory, and EMIFA registers. Because the EMIFA can process only one request at a time, a high performance crossbar switch exists within the SoC to provide prioritized requests from the different sources to the EMIFA. The sources are:

1. CPU
2. EDMA
3. Other master peripherals

If a request is submitted from two or more sources simultaneously, the crossbar switch will forward the highest priority request to the EMIFA first. Upon completion of a request, the crossbar switch again evaluates the pending requests and forwards the highest priority pending request to the EMIFA.

When the EMIFA receives a request, it may or may not be immediately processed. In some cases, the EMIFA will perform one or more auto refresh cycles before processing the request. For details on the EMIFA's internal arbitration between performing requests and performing auto refresh cycles, see [Section 16.2.12](#).

### 16.2.3 Pin Descriptions

This section describes the function of each of the EMIFA pins.

**Table 16-1. EMIFA Pins Used to Access Both SDRAM and Asynchronous Memories**

Pins(s)	I/O	Description
EMA_D[x:0]	I/O	<b>EMIFA data bus.</b> The number of available data bus pins varies among devices, see your device-specific data manual for details.
EMA_A[x:0]	O	<b>EMIFA address bus.</b> The number of available address pins varies among devices, see your device-specific data manual for details. When interfacing to an SDRAM device, these pins are primarily used to provide the row and column address to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Section 16.2.4.11</a> . EMA_A[10] is also used during the PRE command to select which banks to deactivate. When interfacing to an asynchronous device, these pins are used in conjunction with the EMA_BA pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Section 16.2.5.1</a> .
EMA_BA[1:0]	O	<b>EMIFA bank address.</b> When interfacing to an SDRAM device, these pins are used to provide the bank address inputs to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Section 16.2.4.11</a> . When interfacing to an asynchronous device, these pins are used in conjunction with the EMA_A pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in <a href="#">Section 16.2.5.1</a> .
EMA_WE_DQM[x:0]	O	<b>Active-low byte enables.</b> When interfacing to SDRAM, these pins are connected to the DQM pins of the SDRAM to individually enable/disable each of the bytes in a data access. When interfacing to an asynchronous device, these pins are connected to byte enables. See <a href="#">Section 16.2.5</a> for details.
EMA_WE	O	<b>Active-low write enable.</b> When interfacing to SDRAM, this pin is connected to the $\overline{\text{WE}}$ pin of the SDRAM and is used to send commands to the device. When interfacing to an asynchronous device, this pin provides a signal which is active-low during the strobe period of an asynchronous write access cycle.

**Table 16-2. EMIFA Pins Specific to SDRAM**

Pin(s)	I/O	Description
EMA_CS[0]	O	<b>Active-low chip enable pin for SDRAM devices.</b> This pin is connected to the chip-select pin of the attached SDRAM device and is used for enabling/disabling commands. By default, the EMIFA keeps this SDRAM chip select active, even if the EMIFA is not interfaced with an SDRAM device. This pin is deactivated when accessing the asynchronous memory bank and is reactivated on completion of the asynchronous access.
EMA_RAS	O	<b>Active-low row address strobe pin.</b> This pin is connected to the RAS pin of the attached SDRAM device and is used for sending commands to the device.
EMA_CAS	O	<b>Active-low column address strobe pin.</b> This pin is connected to the CAS pin of the attached SDRAM device and is used for sending commands to the device.
EMA_SDCKE	O	<b>Clock enable pin.</b> This pin is connected to the CKE pin of the attached SDRAM device and is used for issuing the SELF REFRESH command which places the device in self refresh mode. See <a href="#">Section 16.2.4.7</a> for details.
EMA_CLK	O	<b>SDRAM clock pin.</b> This pin is connected to the CLK pin of the attached SDRAM device. See <a href="#">Section 16.2.1</a> for details on the clock signal.

**Table 16-3. EMIFA Pins Specific to Asynchronous Memory**

Pin(s)	I/O	Description
EMA_CS[5:2]	O	<b>Active-low chip enable pins for asynchronous devices.</b> These pins are meant to be connected to the chip-select pins of the attached asynchronous device. These pins are active only during accesses to the asynchronous memory.
EMA_WAIT	I	<b>Wait input with programmable polarity / NAND Flash ready input.</b> Not all devices support both EMA_WAIT[1] and EMA_WAIT[0], see your device-specific data manual to determine support on each device. A connected asynchronous device can extend the strobe period of an access cycle by asserting the EMA_WAIT input to the EMIFA as described in <a href="#">Section 16.2.5.7</a> . To enable this functionality, the EW bit in the asynchronous <i>n</i> configuration register (CE <sub>n</sub> CFG) must be set to 1. The WP0 and WP1 bits in the asynchronous wait cycle configuration register (AWCC) must be configured to define the polarity of the EMA_WAIT pin. The CS <sub><i>n</i></sub> _WAIT bit in AWCC must also be configured to determine which EMA_WAIT[ <i>n</i> ] signal is used for memory accesses. When the CS2NAND/CS3NAND/CS4NAND/CS5NAND bit in the NAND Flash control register (NANDFCR) is set, this pin instead functions as a NAND Flash ready input.
EMA_OE	O	<b>Active-low pin enable for asynchronous devices.</b> This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle.
EMA_A_RW	O	<b>EMIFA asynchronous read/write control.</b> This pin stays high during reads and stays low during writes (same duration as CS).

### 16.2.4 SDRAM Controller and Interface

The EMIFA can gluelessly interface to most standard SDR SDRAM devices and supports such features as self refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to interface and properly configure the EMIFA to perform read and write operations to externally connected SDR SDRAM devices. Also, [Section 16.3](#) provides a detailed example of interfacing the EMIFA to a common SDRAM device.

#### 16.2.4.1 SDRAM Commands

The EMIFA supports the SDRAM commands described in [Table 16-4](#). The truth table for the SDRAM commands is shown in [Table 16-5](#) and an example timing waveform of the PRE command is shown in [Figure 16-2](#). EMA\_A[10] is pulled low in this example to deactivate only the bank specified by the EMA\_BA pins.

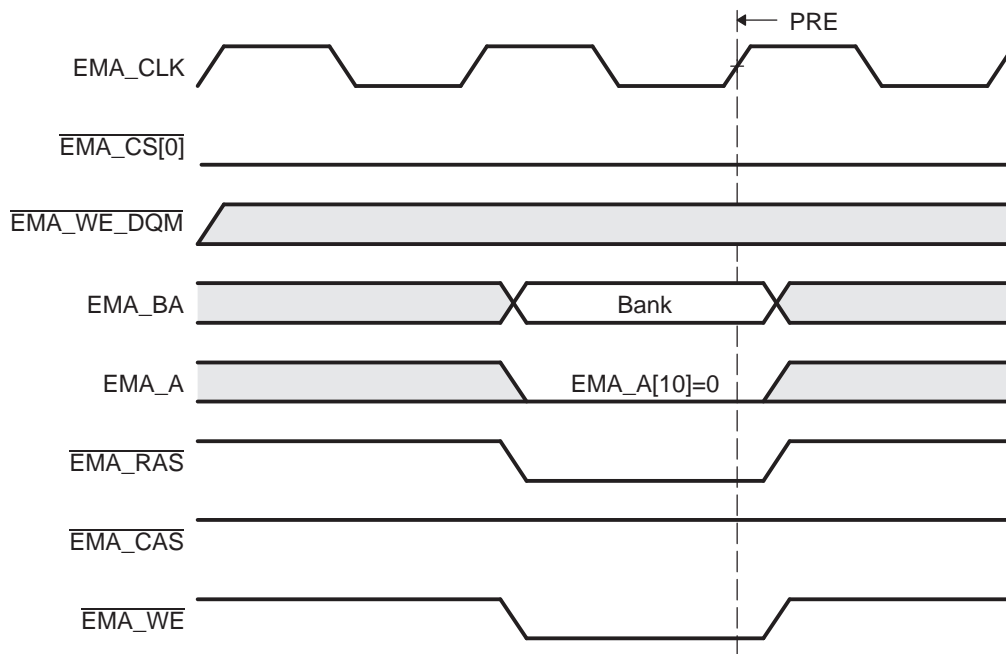
**Table 16-4. EMIFA SDRAM Commands**

Command	Function
PRE	<b>Precharge.</b> Depending on the value of EMA_A[10], the PRE command either deactivates the open row in all banks (EMA_A[10] = 1) or only the bank specified by the EMA_BA[1:0] pins (EMA_A[10] = 0).
ACTV	<b>Activate.</b> The ACTV command activates the selected row in a particular bank for the current access.
READ	<b>Read.</b> The READ command outputs the starting column address and signals the SDRAM to begin the burst read operation. Address EMA_A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
WRT	<b>Write.</b> The WRT command outputs the starting column address and signals the SDRAM to begin the burst write operation. Address EMA_A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
BT	<b>Burst terminate.</b> The BT command is used to truncate the current read or write burst request.
LMR	<b>Load mode register.</b> The LMR command sets the mode register of the attached SDRAM devices and is only issued during the SDRAM initialization sequence described in <a href="#">Section 16.2.4.4</a> .
REFR	<b>Auto refresh.</b> The REFR command signals the SDRAM to perform an auto refresh according to its internal address.
SLFR	<b>Self refresh.</b> The self refresh command places the SDRAM into self refresh mode, during which it provides its own clock signal and auto refresh cycles.
NOP	<b>No operation.</b> The NOP command is issued during all cycles in which one of the above commands is not issued.

**Table 16-5. Truth Table for SDRAM Commands**

SDRAM Pins:	CKE	CS	RAS	CAS	WE	BA[1:0]	A[12:11]	A[10]	A[9:0]
EMIFA Pins:	EMA_SDCKE	EMA_CS[0]	EMA_RAS	EMA_CAS	EMA_WE	EMA_BA[1:0]	EMA_A[12:11]	EMA_A[10]	EMA_A[9:0]
PRE	H	L	L	H	L	Bank/X	X	L/H	X
ACTV	H	L	L	H	H	Bank	Row	Row	Row
READ	H	L	H	L	H	Bank	Column	L	Column
WRT	H	L	H	L	L	Bank	Column	L	Column
BT	H	L	H	H	L	X	X	X	X
LMR	H	L	L	L	L	X	Mode	Mode	Mode
REFR	H	L	L	L	H	X	X	X	X
SLFR	L	L	L	L	H	X	X	X	X
NOP	H	L	H	H	H	X	X	X	X

**Figure 16-2. Timing Waveform of SDRAM PRE Command**

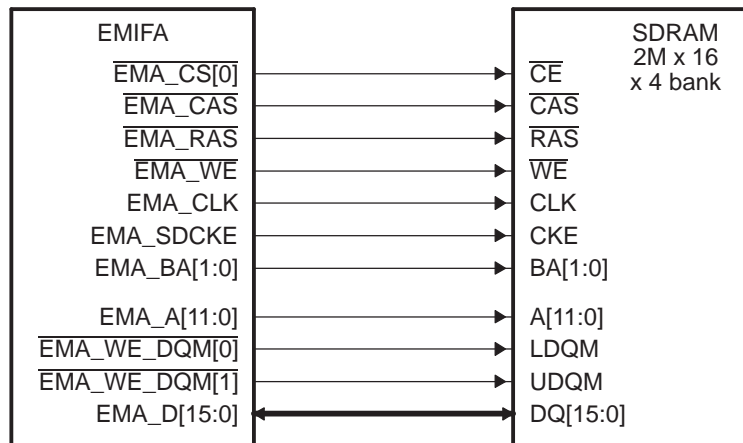
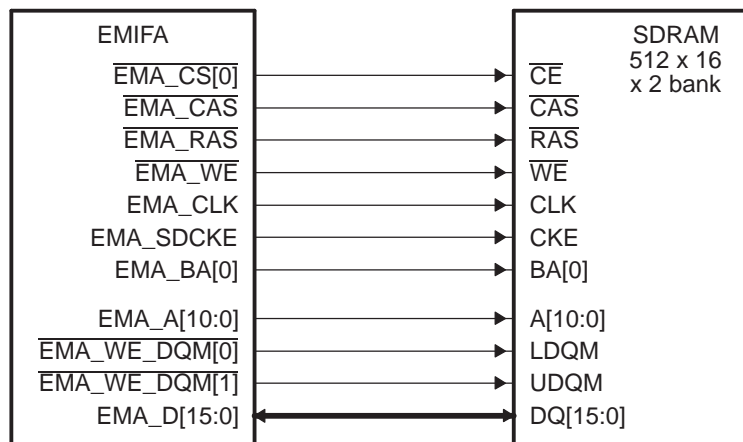


#### 16.2.4.2 Interfacing to SDRAM

The EMIFA supports a glueless interface to SDRAM devices with the following characteristics:

- Pre-charge bit is A[10]
- The number of column address bits is 8, 9, 10, or 11. See your device-specific data manual for the number of column address bits supported on your device.
- The number of row address bits is 13, 14, 15, or 16. See your device-specific data manual for the number of row address bits supported on your device.
- The number of internal banks is 1, 2, or 4. See your device-specific data manual for the number of internal banks supported on your device.

Figure 16-3 shows an interface between the EMIFA and a 2M × 16 × 4 bank SDRAM device, and Figure 16-4 shows an interface between the EMIFA and a 512K × 16 × 2 bank SDRAM device. For devices supporting 16-bit interface, refer to Table 16-6 for list of commonly-supported SDRAM devices and the required connections for the address pins.

**Figure 16-3. EMIFA to 2M x 16 x 4 bank SDRAM Interface**

**Figure 16-4. EMIFA to 512K x 16 x 2 bank SDRAM Interface**

**Table 16-6. 16-bit EMIFA Address Pin Connections**

SDRAM Size	Width	Banks	Device	Address Pins
16M bits	x16	2	SDRAM	A[10:0]
			EMIFA	EMA_A[10:0]
64M bits	x16	4	SDRAM	A[11:0]
			EMIFA	EMA_A[11:0]
128M bits	x16	4	SDRAM	A[11:0]
			EMIFA	EMA_A[11:0]
256M bits	x16	4	SDRAM	A[12:0]
			EMIFA	EMA_A[12:0]
512M bits	x16	4	SDRAM	A[12:0]
			EMIFA	EMA_A[12:0]

### 16.2.4.3 SDRAM Configuration Registers

The operation of the EMIFA's SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but [Section 16.4](#) should be referred for a more detailed description of each register, including the default registers values and bit-field positions. The following tables list the four such configuration registers, along with a description of each of their programmable fields.

**NOTE:** Writing to any of the fields: NM, CL, IBANK, and PAGESIZE in the SDRAM configuration register (SDCR) causes the EMIFA to abandon whatever it is currently doing and trigger the SDRAM initialization procedure described in [Section 16.2.4.4](#).

**Table 16-7. Description of the SDRAM Configuration Register (SDCR)**

Parameter	Description
SR	This bit controls entering and exiting of the Self-Refresh mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence.
PD	This bit controls entering and exiting of the Power down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIFA will go into Self Refresh.
PDWR	Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIFA to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. This bit should be set along with PD when entering power-down mode.
NM	<b>Narrow Mode.</b> This bit defines the width of the data bus between the EMIFA and the attached SDRAM device. When set to 1, the data bus is set to 16-bits. When set to 0, the data bus is set to 32-bits. This bit must always be set to 1.
CL	<b>CAS latency.</b> This field defines the number of clock cycles between when an SDRAM issues a READ command and when the first piece of data appears on the bus. The value in this field is sent to the attached SDRAM device via the LOAD MODE REGISTER command during the SDRAM initialization procedure as described in <a href="#">Section 16.2.4.4</a> . Only, values of 2h (CAS latency = 2) and 3h (CAS latency = 3) are supported and should be written to this field. A 1 must be simultaneously written to the BIT11_9LOCK bit field of SDCR in order to write to the CL bit field.
IBANK	<b>Number of Internal SDRAM Banks.</b> This field defines the number of banks inside the attached SDRAM devices in the following way: <ul style="list-style-type: none"> <li>When IBANK = 0, 1 internal bank is used</li> <li>When IBANK = 1h, 2 internal banks are used</li> <li>When IBANK = 2h, 4 internal banks are used</li> </ul> This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses. See <a href="#">Section 16.2.4.11</a> for details.
PAGESIZE	<b>Page Size.</b> This field defines the internal page size of the attached SDRAM devices in the following way: <ul style="list-style-type: none"> <li>When PAGESIZE = 0, 256-word pages are used</li> <li>When PAGESIZE = 1h, 512-word pages are used</li> <li>When PAGESIZE = 2h, 1024-word pages are used</li> <li>When PAGESIZE = 3h, 2048-word pages are used</li> </ul> This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses. See <a href="#">Section 16.2.4.11</a> for details.

**Table 16-8. Description of the SDRAM Refresh Control Register (SDRCR)**

Parameter	Description
RR	<b>Refresh Rate.</b> This field controls the rate at which attached SDRAM devices will be refreshed. The following equation can be used to determine the required value of RR for an SDRAM device: <ul style="list-style-type: none"> <li><math>RR = f_{EMA\_CLK} / (\text{Required SDRAM Refresh Rate})</math></li> </ul> More information about the operation of the SDRAM refresh controller can be found in <a href="#">Section 16.2.4.6</a> .



**Table 16-9. Description of the SDRAM Timing Register (SDTIMR)**

Parameter	Description
T_RFC	<b>SDRAM Timing Parameters.</b> These fields configure the EMIFA to comply with the AC timing requirements of the attached SDRAM devices. This allows the EMIFA to avoid violating SDRAM timing constraints and to more efficiently schedule its operations. More details about each of these parameters can be found in the register description in <a href="#">Section 16.4.6</a> . These parameters should be set to satisfy the corresponding timing requirements found in the SDRAM's datasheet.
T_RP	
T_RCD	
T_WR	
T_RAS	
T_RC	
T_RRD	

**Table 16-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR)**

Parameter	Description
T_XS	<b>Self Refresh Exit Parameter.</b> The T_XS field of this register informs the EMIFA about the minimum number of EMA_CLK cycles required between exiting Self Refresh and issuing any command. This parameter should be set to satisfy the $t_{XSR}$ value for the attached SDRAM device.

#### 16.2.4.4 SDRAM Auto-Initialization Sequence

The EMIFA automatically performs an SDRAM initialization sequence, regardless of whether it is interfaced to an SDRAM device, when either of the following two events occur:

- The EMIFA comes out of reset. No memory accesses to the SDRAM and Asynchronous interfaces are performed until this auto-initialization is complete.
- A write is performed to any of the three least significant bytes of the SDRAM configuration register (SDCR)

An SDRAM initialization sequence consists of the following steps:

1. If the initialization sequence is activated by a write to SDCR, and if any of the SDRAM banks are open, the EMIFA issues a PRE command with EMA\_A[10] held high to indicate all banks. This is done so that the maximum ACTV to PRE timing for an SDRAM is not violated.
2. The EMIFA drives EMA\_SDCKE high and begins continuously issuing NOP commands until eight SDRAM refresh intervals have elapsed. An SDRAM refresh interval is equal to the value of the RR field of SDRAM refresh control register (SDRCR), divided by the frequency of EMA\_CLK ( $RR/f_{EMA\_CLK}$ ). This step is used to avoid violating the Power-up constraint of most SDRAM devices that requires 200  $\mu$ s (sometimes 100  $\mu$ s) between receiving stable Vdd and CLK and the issuing of a PRE command. Depending on the frequency of EMA\_CLK, this step may or may not be sufficient to avoid violating the SDRAM constraint. See [Section 16.2.4.5](#) for more information.
3. After the refresh intervals have elapsed, the EMIFA issues a PRE command with EMA\_A[10] held high to indicate all banks.
4. The EMIFA issues eight AUTO REFRESH commands.
5. The EMIFA issues the LMR command with the EMA\_A[9:0] pins set as described in [Table 16-11](#).
6. Finally, the EMIFA performs a refresh cycle, which consists of the following steps:
  - (a) Issuing a PRE command with EMA\_A[10] held high if any banks are open
  - (b) Issuing an REF command

**Table 16-11. SDRAM LOAD MODE REGISTER Command**

EMA_A[9:7]	EMA_A[6:4]	EMA_A[3]	EMA_A[2:0]
0 (Write bursts are of the programmed burst length in EMA_A[2:0])	These bits control the CAS latency of the SDRAM and are set according to CL field in the SDRAM configuration register (SDCR) as follows: <ul style="list-style-type: none"> <li>If CL = 2, EMA_A[6:4] = 2h (CAS latency = 2)</li> <li>If CL = 3, EMA_A[6:4] = 3h (CAS latency = 3)</li> </ul>	0 (Sequential Burst Type. Interleaved Burst Type not supported)	These bits control the burst length of the SDRAM and are set according to the NM field in the SDRAM configuration register (SDCR) as follows: <ul style="list-style-type: none"> <li>If NM = 0, EMA_A[2:0] = 2h (Burst Length = 4)</li> <li>If NM = 1, EMA_A[2:0] = 3h (Burst Length = 8)</li> </ul>

#### 16.2.4.5 SDRAM Configuration Procedure

There are two different SDRAM configuration procedures. Although EMIFA automatically performs the SDRAM initialization sequence described in [Section 16.2.4.4](#) when coming out of reset, it is recommended to follow one of the procedures listed below before performing any EMIFA memory requests. Procedure A should be followed if it is determined that the SDRAM Power-up constraint was not violated during the SDRAM Auto-Initialization Sequence detailed in [Section 16.2.4.4](#) on coming out of Reset. The SDRAM Power-up constraint specifies that 200  $\mu$ s (sometimes 100  $\mu$ s) should elapse between receiving stable V<sub>dd</sub> and CLK and the issuing of a PRE command. Procedure B should be followed if the SDRAM Power-up constraint was violated. The 200  $\mu$ s (100  $\mu$ s) SDRAM Power-up constraint will be violated if the frequency of EMA\_CLK is greater than 50 MHz (100 MHz for 100  $\mu$ s SDRAM power-up constraint) during SDRAM Auto-Initialization Sequence. Procedure B should be followed if there is any doubt that the Power-up constraint was met.

**Procedure A** — Following is the procedure to be followed if the SDRAM Power-up constraint was NOT violated:

1. Place the SDRAM into Self-Refresh Mode by setting the SR bit of SDCR to 1. A byte-write to the upper byte of SDCR should be used to avoid restarting the SDRAM Auto-Initialization Sequence described in [Section 16.2.4.4](#). The SDRAM should be placed into Self-Refresh mode when changing the frequency of EMA\_CLK to avoid incurring the 200  $\mu$ s Power-up constraint again.
2. Program the CPU's PLL Controller to provide the desired EMA\_CLK clock frequency. Refer to the device Data Manual for details on programming the PLL Controller. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device Data Manual.
3. Remove the SDRAM from Self-Refresh Mode by clearing the SR bit of SDCR to 0. A byte-write to the upper byte of SDCR should be used to avoid restarting the SDRAM Auto-Initialization Sequence described in [Section 16.2.4.4](#).
4. Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.
5. Program the RR field of SDCR to match that of the attached device's refresh interval. See [Section 16.2.4.6.1](#) details on determining the appropriate value.
6. Program SDCR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in [Section 16.2.4.4](#) to be re-run. This second initialization generally takes much less time due to the increased frequency of EMA\_CLK.

**Procedure B** — Following is the procedure to be followed if the SDRAM Power-up constraint was violated:

1. Program the CPU's PLL Controller to provide the desired EMA\_CLK clock frequency. Refer to the device Data Manual for details on programming the PLL Controller. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device Data Manual.
2. Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.

3. Program the RR field of SDRCR such that the following equation is satisfied:  $(RR \times 8)/(f_{EMA\_CLK}) > 200 \mu s$  (sometimes  $100 \mu s$ ). For example, an EMA\_CLK frequency of 100 MHz would require setting RR to 2501 (9C5h) or higher to meet a 200  $\mu s$  constraint.
4. Program SDCR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in [Section 16.2.4.4](#) to be re-run with the new value of RR.
5. Perform a read from the SDRAM to assure that step 5 of this procedure will occur after the initialization process has completed. Alternatively, wait for 200  $\mu s$  instead of performing a read.
6. Finally, program the RR field to match that of the attached device's refresh interval. See [Section 16.2.4.6.1](#) details on determining the appropriate value.

After following the above procedure, the EMIFA is ready to perform accesses to the attached SDRAM device. See [Section 16.3](#) for an example of configuring the SDRAM interface.

#### 16.2.4.6 EMIFA Refresh Controller

An SDRAM device requires that each of its rows be refreshed at a minimum required rate. The EMIFA can meet this constraint by performing auto refresh cycles at or above this required rate. An auto refresh cycle consists of issuing a PRE command to all banks of the SDRAM device followed by issuing a REFR command. To inform the EMIFA of the required rate for performing auto refresh cycles, the RR field of the SDRAM refresh control register (SDRCR) must be programmed. The EMIFA will use this value along with two internal counters to automatically perform auto refresh cycles at the required rate. The auto refresh cycles cannot be disabled, even if the EMIFA is not interfaced with an SDRAM. The remainder of this section details the EMIFA's refresh scheme and provides an example for determining the appropriate value to place in the RR field of SDRCR.

The two counters used to perform auto-refresh cycles are a 13-bit refresh interval counter and a 4-bit refresh backlog counter. At reset and upon writing to the RR field, the refresh interval counter is loaded with the value from RR field and begins decrementing, by one, each EMIFA clock cycle. When the refresh interval counter reaches zero, the following actions occur:

- The refresh interval counter is reloaded with the value from the RR field and restarts decrementing.
- The 4-bit refresh backlog counter increments unless it has already reached its maximum value.

The refresh backlog counter records the number of auto refresh cycles that the EMIFA currently has outstanding. This counter is decremented by one each time an auto refresh cycle is performed and incremented by one each time the refresh interval counter expires. The refresh backlog counter saturates at the values of 0000b and 1111b. The EMIFA uses the refresh backlog counter to determine the urgency with which an auto refresh cycle should be performed. The four levels of urgency are described in [Table 16-12](#). This refresh scheme allows the required refreshes to be performed with minimal impact on access requests.

**Table 16-12. Refresh Urgency Levels**

Urgency Level	Refresh Backlog Counter Range	Action Taken
Refresh May	1-3	An auto-refresh cycle is performed only if the EMIFA has no requests pending and none of the SDRAM banks are open.
Refresh Release	4-7	An auto-refresh cycle is performed if the EMIFA has no requests pending, regardless of whether any SDRAM banks are open.
Refresh Need	8-11	An auto-refresh cycle is performed at the completion of the current access unless there are read requests pending.
Refresh Must	12-15	Multiple auto-refresh cycles are performed at the completion of the current access until the Refresh Release urgency level is reached. At that point, the EMIFA can begin servicing any new read or write requests.

### 16.2.4.6.1 Determining the Appropriate Value for the RR Field

The value that should be programmed into the RR field of SDCR can be calculated by using the frequency of the EMA\_CLK signal ( $f_{\text{EMA\_CLK}}$ ) and the required refresh rate of the SDRAM ( $f_{\text{Refresh}}$ ). The following formula can be used:

$$\text{RR} = f_{\text{EMA\_CLK}} / f_{\text{Refresh}}$$

The SDRAM datasheet often communicates the required SDRAM Refresh Rate in terms of the number of REFR commands required in a given time interval. The required SDRAM Refresh Rate in the formula above can therefore be calculated by dividing the number of required cycles per time interval ( $n_{\text{cycles}}$ ) by the time interval given in the datasheet ( $t_{\text{Refresh Period}}$ ):

$$f_{\text{Refresh}} = n_{\text{cycles}} / t_{\text{Refresh Period}}$$

Combining these formulas, the value that should be programmed into the RR field can be computed as:

$$\text{RR} = f_{\text{EMA\_CLK}} \times t_{\text{Refresh Period}} / n_{\text{cycles}}$$

The following example illustrates calculating the value of RR. Given that:

- $f_{\text{EMA\_CLK}} = 100 \text{ MHz}$  (frequency of the EMIFA clock)
- $t_{\text{Refresh Period}} = 64 \text{ ms}$  (required refresh interval of the SDRAM)
- $n_{\text{cycles}} = 8192$  (number of cycles in a refresh interval for the SDRAM)

RR can be calculated as:

$$\text{RR} = 100 \text{ MHz} \times 64 \text{ ms} / 8192$$

$$\text{RR} = 781.25$$

$$\text{RR} = 782 \text{ cycles} = 30\text{Eh cycles}$$

### 16.2.4.7 Self-Refresh Mode

The EMIFA can be programmed to enter the self-refresh state by setting the SR bit of SDCR to 1. This will cause the EMIFA to issue the SLFR command after completing any outstanding SDRAM access requests and clearing the refresh backlog counter by performing one or more auto refresh cycles. This places the attached SDRAM device into self-refresh mode in which it consumes a minimal amount of power while performing its own refresh cycles. The SR bit should be set and cleared using a byte-write to the upper byte of the SDRAM configuration register (SDCR) to avoid triggering the SDRAM initialization sequence.

While in the self-refresh state, the EMIFA continues to service asynchronous bank requests and register accesses as normal, with one caveat. The EMIFA will not park the data bus following a read to asynchronous memory while in the self-refresh state. Instead, the EMIFA tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIFA is in the self-refresh state, in order to prevent floating inputs on the data bus. More information about data bus parking can be found in [Section 16.2.6](#).

The EMIFA will exit from the self-refresh state if either of the following events occur:

- The SR bit of SDCR is cleared to 0.
- An SDRAM accesses is requested.

The EMIFA exits from the self-refresh state by driving EMA\_SDCKE high and performing an auto refresh cycle.

The attached SDRAM device should also be placed into Self-Refresh Mode when changing the frequency of EMA\_CLK using the PLL Controller. If the frequency of EMA\_CLK changes while the SDRAM is not in Self-Refresh Mode, Procedure B in [Section 16.2.4.5](#) should be followed to reinitialize the device.

### 16.2.4.8 Power Down Mode

To support low-power modes, the EMIFA can be requested to issue a POWER DOWN command to the SDRAM by setting the PD bit in the SDRAM configuration register (SDCR). When this bit is set, the EMIFA will continue normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog (if there is one) has been cleared. At this point the EMIFA will enter the power-down state. Upon entering this state, the EMIFA will issue a POWER DOWN command (same as a NOP command but driving EMA\_SDCKE low on the same cycle). The EMIFA then maintains EMA\_SDCKE low until it exits the power-down state.

Since the EMIFA services the refresh backlog before it enters the power-down state, all internal banks of the SDRAM are closed (precharged) prior to issuing the POWER DOWN command. Therefore, the EMIFA only supports Precharge Power Down. The EMIFA does not support Active Power Down, where internal banks of the SDRAM are open (active) before the POWER DOWN command is issued.

During the power-down state, the EMIFA services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power-down state upon completion.

The PDWR bit in SDCR indicates whether the EMIFA should perform refreshes in power-down state. If the PDWR bit is set, the EMIFA exits the power-down state every time the Refresh Must level is set, performs AUTO REFRESH commands to the SDRAM, and returns back to the power-down state. This evenly distributes the refreshes to the SDRAM in power-down state. If the PDWR bit is not set, the EMIFA does not perform any refreshes to the SDRAM. Therefore, the data integrity of the SDRAM is not assured upon power down exit if the PDWR bit is not set.

If the PD bit is cleared while in the power-down state, the EMIFA will come out of the power-down state. The EMIFA:

- Drives EMA\_SDCKE high.
- Enters its idle state.

### 16.2.4.9 SDRAM Read Operation

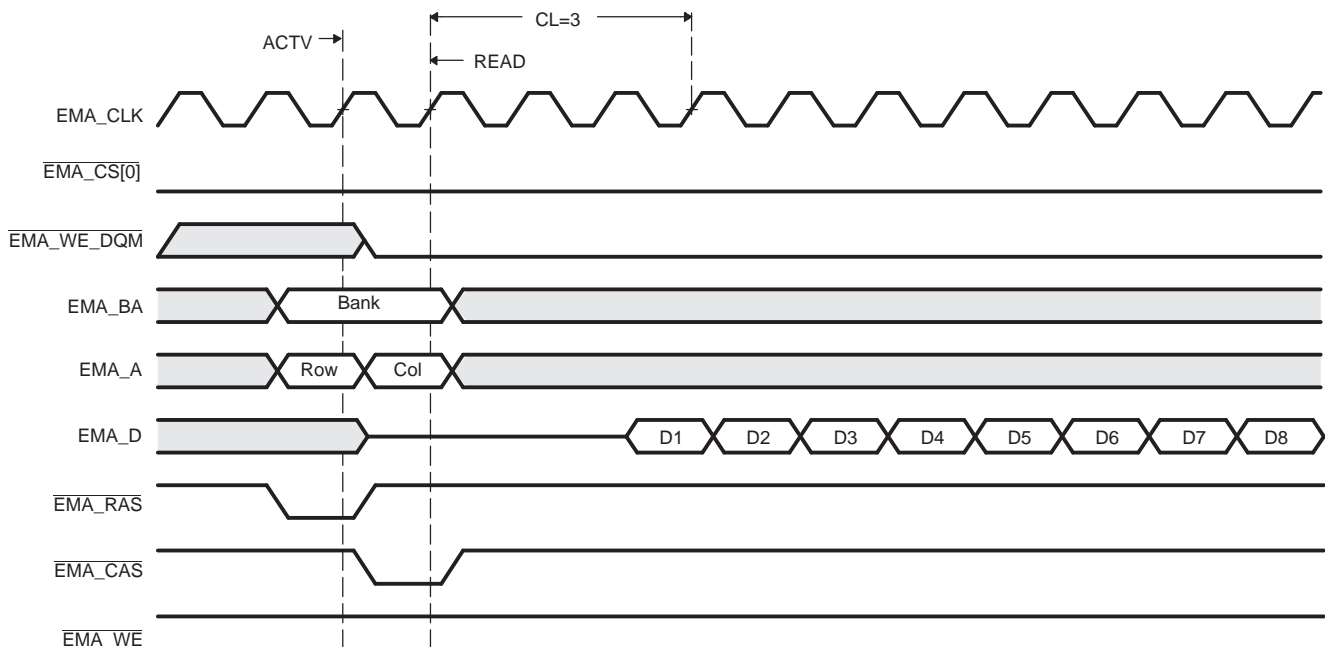
When the EMIFA receives a read request to SDRAM from one of the requesters listed in [Section 16.2.2](#), it performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIFA proceeds to issue a READ command while specifying the desired bank and column address. EMA\_A[10] is held low during the READ command to avoid auto-precharging. The READ command signals the SDRAM device to start bursting data from the specified address while the EMIFA issues NOP commands. Following a READ command, the CL field of the SDRAM configuration register (SDCR) defines how many delay cycles will be present before the read data appears on the data bus. This is referred to as the CAS latency.

[Figure 16-5](#) shows the signal waveforms for a basic SDRAM read operation in which a burst of data is read from a single page. When the EMIFA SDRAM interface is configured to 16 bit by setting the NM bit of the SDRAM configuration register (SDCR) to 1, a burst size of eight is used. [Figure 16-5](#) shows a burst size of eight.

The EMIFA will truncate a series of bursting data if the remaining addresses of the burst are not required to complete the request. The EMIFA can truncate the burst in three ways:

- By issuing another READ to the same page in the same bank.
- By issuing a PRE command in order to prepare for accessing a different page of the same bank.
- By issuing a BT command in order to prepare for accessing a page in a different bank.

**Figure 16-5. Timing Waveform for Basic SDRAM Read Operation**



Several other pins are also active during a read access. The  $\overline{\text{EMA\_WE\_DQM}}[1:0]$  pins are driven low during the READ commands and are kept low during the NOP commands that correspond to the burst request. The state of the other EMIFA pins during each command can be found in [Table 16-5](#).

The EMIFA schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIFA uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands between various commands during an access. Refer to the register description of SDTIMR in [Section 16.4.6](#) for more details on the various timing parameters.

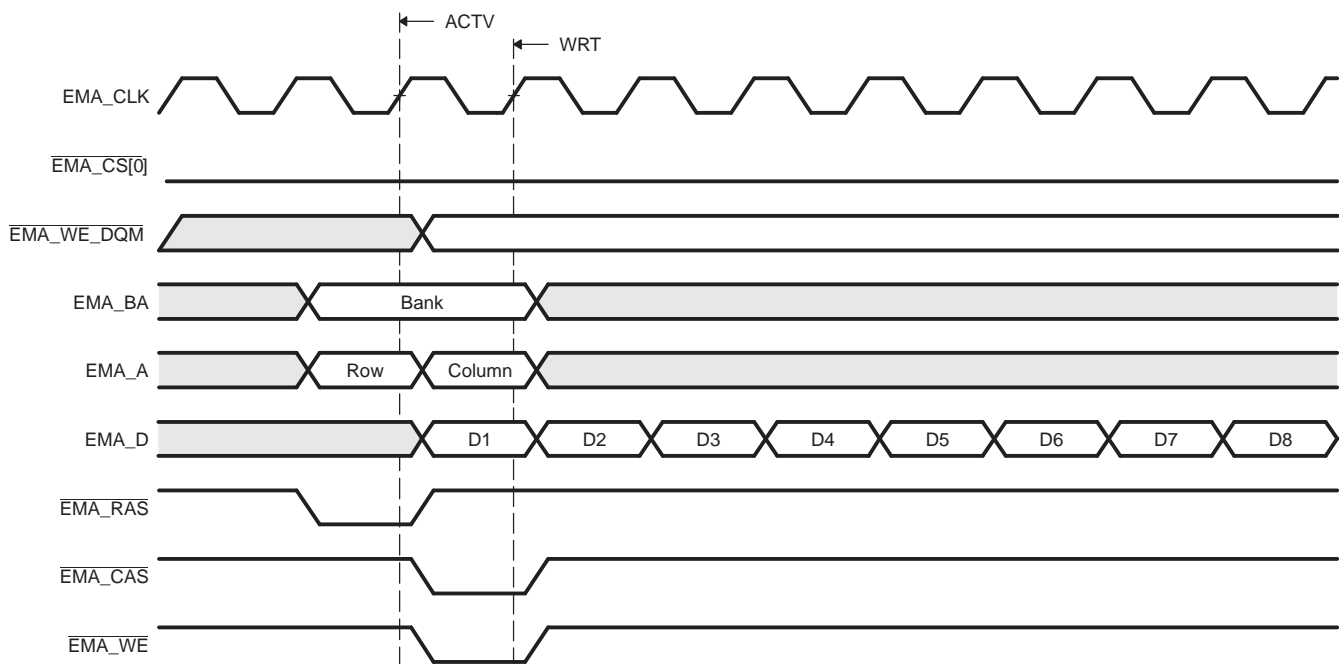


### 16.2.4.10 SDRAM Write Operations

When the EMIFA receives a write request to SDRAM from one of the requesters listed in [Section 16.2.2](#), it performs one or more write-access cycles. A write-access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIFA proceeds to issue a WRT command while specifying the desired bank and column address. EMA\_A[10] is held low during the WRT command to avoid auto-precharging. The WRT command signals the SDRAM device to start writing a burst of data to the specified address while the EMIFA issues NOP commands. The associated write data will be placed on the data bus in the cycle concurrent with the WRT command and with subsequent burst continuation NOP commands.

[Figure 16-6](#) shows the signal waveforms for a basic SDRAM write operation in which a burst of data is read from a single page. When the EMIFA SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDCR) to 1, a burst size of eight is used. [Figure 16-6](#) shows a burst size of eight.

**Figure 16-6. Timing Waveform for Basic SDRAM Write Operation**



The EMIFA will truncate a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIFA can truncate the burst in three ways:

- By issuing another WRT to the same page
- By issuing a PRE command in order to prepare for accessing a different page of the same bank
- By issuing a BT command in order to prepare for accessing a page in a different bank

Several other pins are also active during a write access. The EMA\_WE\_DQM[1:0] pins are driven to select which bytes of the data word will be written to the SDRAM device. They are also used to mask out entire undesired data words during a burst access. The state of the other EMIFA pins during each command can be found in [Table 16-5](#).

The EMIFA schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIFA uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands during various cycles of an access. Refer to the register description of SDTIMR in [Section 16.4.6](#) for more details on the various timing parameters.

### 16.2.4.11 Mapping from Logical Address to EMIFA Pins

When the EMIFA receives an SDRAM access request, it must convert the address of the access into the appropriate signals to send to the SDRAM device. The details of this address mapping are shown in [Table 16-13](#) for 16-bit operation. Using the settings of the IBANK and PAGESIZE fields of the SDRAM configuration register (SDCR), the EMIFA determines which bits of the logical address are mapped to the SDRAM row, column, and bank addresses.

As the logical address is incremented by one halfword (16-bit operation), the column address is likewise incremented by one until a page boundary is reached. When the logical address increments across a page boundary, the EMIFA moves into the same page in the next bank of the attached device by incrementing the bank address EMA\_BA and resetting the column address. The page in the previous bank is left open until it is necessary to close it. This method of traversal through the SDRAM banks helps maximize the number of open banks inside of the SDRAM and results in an efficient use of the device. There is no limitation on the number of banks that can be open at one time, but only one page within a bank can be open at a time.

The EMIFA uses the EMA\_WE\_DQM pins during a WRT command to mask out selected bytes or entire words. The EMA\_WE\_DQM pins are always low during a READ command.

**Table 16-13. Mapping from Logical Address to EMIFA Pins for 16-bit SDRAM**

IBANK	PAGESIZE	Logical Address														
		31:27	26	25	24	23	22	21:14	13	12	11	10	9	8:1	0	
0	0	-						Row Address						Col Address	EMA_WE_DQM[0]	
1	0	-						Row Address						EMA_BA[0]	Col Address	EMA_WE_DQM[0]
2	0	-						Row Address						EMA_BA[1:0]	Col Address	EMA_WE_DQM[0]
0	1	-						Row Address						Column Address		EMA_WE_DQM[0]
1	1	-						Row Address						EMA_BA[0]	Column Address	EMA_WE_DQM[0]
2	1	-						Row Address						EMA_BA[1:0]	Column Address	EMA_WE_DQM[0]
0	2	-						Row Address						Column Address		EMA_WE_DQM[0]
1	2	-						Row Address						EMA_BA[0]	Column Address	EMA_WE_DQM[0]
2	2	-						Row Address						EMA_BA[1:0]	Column Address	EMA_WE_DQM[0]
0	3	-						Row Address						Column Address		EMA_WE_DQM[0]
1	3	-						Row Address						EMA_BA[0]	Column Address	EMA_WE_DQM[0]
2	3	-						Row Address						EMA_BA[1:0]	Column Address	EMA_WE_DQM[0]

---

**NOTE:** The upper bit of the Row Address is used only when addressing 256-Mbit and 512-Mbit SDRAM memories.

---



## 16.2.5 Asynchronous Controller and Interface

The EMIFA easily interfaces to a variety of asynchronous devices including NOR Flash, NAND Flash, and SRAM. It can be operated in two major modes (see [Table 16-14](#)):

- Normal Mode
- Select Strobe Mode

**Table 16-14. Normal Mode vs. Select Strobe Mode**

Mode	Function of $\overline{\text{EMA\_WE\_DQM}}$ pins	Operation of $\overline{\text{EMA\_CS}}[5:2]$
Normal Mode	Byte enables	Active during the entire asynchronous access cycle
Select Strobe Mode	Byte enables	Active only during the strobe period of an access cycle

The first mode of operation is Normal Mode, in which the  $\overline{\text{EMA\_WE\_DQM}}$  pins of the EMIFA function as byte enables. In this mode, the  $\overline{\text{EMA\_CS}}[5:2]$  pins behaves as typical chip select signals, remaining active for the duration of the asynchronous access. See [Section 16.2.5.1](#) for an example interface with multiple 8-bit devices.

The second mode of operation is Select Strobe Mode, in which the  $\overline{\text{EMA\_CS}}[5:2]$  pins act as a strobe, active only during the strobe period of an access. In this mode, the  $\overline{\text{EMA\_WE\_DQM}}$  pins of the EMIFA function as standard byte enables for reads and writes. A summary of the differences between the two modes of operation are shown in [Table 16-14](#). Refer to [Section 16.2.5.4](#) for the details of asynchronous operations in Normal Mode, and to [Section 16.2.5.5](#) for the details of asynchronous operations in Select Strobe Mode. The EMIFA hardware defaults to Normal Mode, but can be manually switched to Select Strobe Mode by setting the SS bit in the asynchronous  $m$  ( $m = 1, 2, 3, \text{ or } 4$ ) configuration register (CE $n$ CFG) ( $n = 2, 3, 4, \text{ or } 5$ ). Throughout the chapter,  $m$  can hold the values 1, 2, 3 or 4; and  $n$  can hold the values 2, 3, 4, or 5.

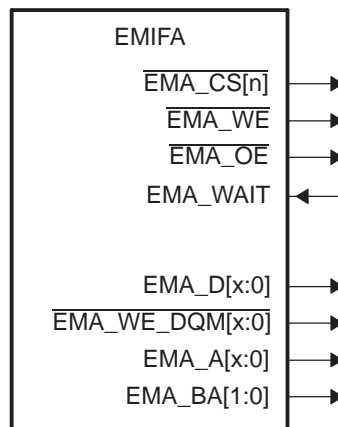
In both Normal Mode and Select Strobe Mode, the EMIFA can be configured to operate in a sub-mode called NAND Flash Mode. In NAND Flash Mode, the EMIFA is able to calculate an error correction code (ECC) for transfers up to 518 bytes.

The EMIFA also provides configurable cycle timing parameters and an Extended Wait Mode that allows the connected device to extend the strobe period of an access cycle. The following sections describe the features related to interfacing with external asynchronous devices.

### 16.2.5.1 Interfacing to Asynchronous Memory

[Figure 16-7](#) shows the EMIFA's external pins used in interfacing with an asynchronous device. In  $\overline{\text{EMA\_CS}}[n]$ ,  $n = 2, 3, 4, \text{ or } 5$ .

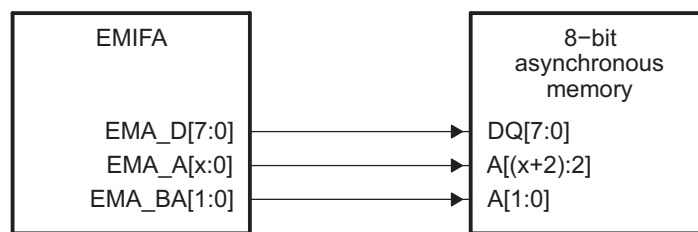
**Figure 16-7. EMIFA Asynchronous Interface**



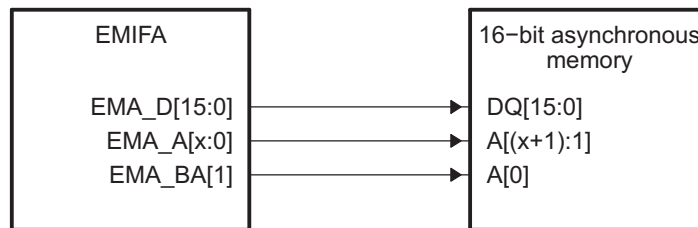
Of special note is the connection between the EMIFA and the external device's address bus. The EMIFA address pin EMA\_A[0] always provides the least significant bit of a 32-bit word address. Therefore, when interfacing to a 16-bit or 8-bit asynchronous device, the EMA\_BA[1] and EMA\_BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Additionally, when the EMIFA interfaces to a 16-bit asynchronous device, the EMA\_BA[0] pin can serve as the upper address line EMA\_A[22]. Note that the width of the address bus varies with devices; therefore, see your device-specific data manual for the EMA\_A bus width supported. Figure 16-8 and Figure 16-9 show the mapping between the EMIFA and the connected device's data and address pins for various programmed data bus widths. The data bus width may be configured in the asynchronous *n* configuration register (CE<sub>n</sub>CFG).

Figure 16-9 shows a common interface between the EMIFA and external asynchronous memory. Figure 16-9 shows an interface between the EMIFA and an external memory with byte enables. The EMIFA should be operated in either Normal Mode or Select Strobe Mode when using this interface, so that the EMA\_WE\_DQM signals operate as byte enables.

**Figure 16-8. EMIFA to 8-bit/16-bit Memory Interface**

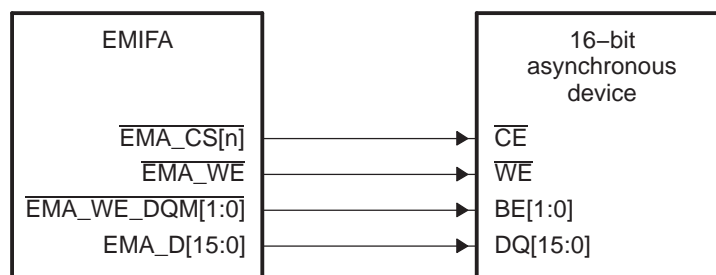


a) EMIF to 8-bit memory interface



b) EMIF to 16-bit memory interface

**Figure 16-9. Common Asynchronous Interface**



### 16.2.5.2 Accessing Larger Asynchronous Memories

The device has a limited number of dedicated EMIFA address pins, enough to interface directly to an SDRAM. If a device such as an asynchronous flash needs to be attached to the EMIFA, then GPIO pins may be used to control the flash device's upper address lines. This is sufficient to boot from the flash. Normally, code stored in flash is copied into SDRAM or internal memory before executing because these memories have much faster access times. For details on which device pins are GPIO capable, see your device-specific data manual.

The ROM bootloader can load a secondary bootloader from an attached asynchronous device. The ROM bootloader assumes that any GPIO pins used to control the upper address lines of the boot flash will be pulled to 0 after reset. This means that normally the GPIO pins selected for this function will be either spare or used as outputs only by the application, and therefore can be pulled to 0 at reset with an external pulldown resistor. The GPIO pins chosen should be tri-stated by default on device reset. For details on which GPIO-capable pins are tri-stated on device reset, see your device-specific data manual.

When booting from flash, the ROM bootloader copies a board-specific secondary bootloader from the lower portion of the flash, so it does not need to manipulate the upper address lines. Only the secondary bootloader, which is board-specific and is stored in the external flash, needs to know which GPIO pins have been assigned to the function of upper address lines. Therefore, the secondary bootloader can perform the task of configuring the selected pins as GPIO and loading the remainder of the code from the upper flash memory.

### 16.2.5.3 Configuring the EMIFA for Asynchronous Accesses

The operation of the EMIFA's asynchronous interface can be configured by programming the appropriate register fields. The reset value and bit position for each register field can be found in [Section 16.4](#), but the Boot ROM documentation should be consulted to determine if the fields are programmed during boot. The following tables list the register fields that can be programmed and describe the purpose of each field. These registers can be programmed prior to accessing the external memory, and the transfer following a write to these registers will use the new configuration.

**Table 16-15. Description of the Asynchronous *m* Configuration Register (CE<sub>n</sub>CFG)**

Parameter	Description
SS	<p><b>Select Strobe mode.</b> This bit selects the EMIFA's mode of operation in the following way:</p> <ul style="list-style-type: none"> <li>• SS = 0 selects Normal Mode                             <ul style="list-style-type: none"> <li>– EMA_WE_DQM pins function as byte enables</li> <li>– EMA_CS[5:2] active for duration of access</li> </ul> </li> <li>• SS = 1 selects Select Strobe Mode                             <ul style="list-style-type: none"> <li>– EMA_WE_DQM pins function as byte enables</li> <li>– EMA_CS[5:2] acts as a strobe.</li> </ul> </li> </ul>
EW	<p><b>Extended Wait Mode enable.</b></p> <ul style="list-style-type: none"> <li>• EW = 0 disables Extended Wait Mode</li> <li>• EW = 1 enables Extended Wait Mode</li> </ul> <p>When set to 1, the EMIFA enables its Extended Wait Mode in which the strobe width of an access cycle can be extended in response to the assertion of the EMA_WAIT pin<sup>(1)</sup>. The WP<sub>n</sub> bit in the asynchronous wait cycle configuration register (AWCC) controls to polarity of EMA_WAIT pin. Extended Wait Mode should not be used while in NAND Flash Mode. See <a href="#">Section 16.2.5.7</a> for more details on this mode of operation.</p>
W_SETUP/R_SETUP	<p><b>Read/Write setup widths.</b></p> <p>These fields define the number (n) of EMIFA clock cycles of setup time for the address pins (EMA_A and EMA_BA), byte enables (EMA_WE_DQM), and asynchronous chip enable (EMA_CS[5:2]) before the read strobe pin (EMA_OE) or write strobe pin (EMA_WE) falls. This value should be encoded as n - 1, where n is the number of EMIFA clock cycles. For example, when W_SETUP = 2, then write setup width = 3 EMA_CLK cycles. For writes, the W_SETUP field also defines the setup time for the data pins (EMA_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>

<sup>(1)</sup> The EMA\_WAIT pin is not available on all devices; therefore, this field is reserved on those devices.

**Table 16-15. Description of the Asynchronous *m* Configuration Register (CE<sub>*n*</sub>CFG) (continued)**

Parameter	Description
W_STROBE/R_STROBE	<p><b>Read/Write strobe widths.</b></p> <p>These fields define the number (<i>n</i>) of EMIFA clock cycles between the falling and rising edge of the read strobe pin (EMA_OE) or write strobe pin (EMA_WE). This value should be encoded as <i>n</i> - 1, where <i>n</i> is the number of EMIFA clock cycles. For example, when W_SETUP = 2, then write setup width = 3 EMA_CLK cycles. If Extended Wait Mode is enabled by setting the EW field in the asynchronous <i>n</i> configuration register (CE<sub><i>n</i></sub>CFG), these fields must be set to a value greater than zero. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>
W_HOLD/R_HOLD	<p><b>Read/Write hold widths.</b> <sup>(2)</sup></p> <p>These fields define the number (<i>n</i>) of EMIFA clock cycles of hold time for the address pins (EMA_A and EMA_BA), byte enables (EMA_WE_DQM), and asynchronous chip enable (EMA_CS[5:2]) after the read strobe pin (EMA_OE) or write strobe pin (EMA_WE) rises. This value should be encoded as <i>n</i> - 1, where <i>n</i> is the number of EMIFA clock cycles. For example, when W_SETUP = 2, then write setup width = 3 EMA_CLK cycles. For writes, the W_HOLD field also defines the hold time for the data pins (EMA_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>
TA	<p><b>Minimum turnaround time.</b></p> <p>This field defines the minimum number of EMIFA clock cycles between asynchronous reads and writes, minus one cycle. The purpose of this feature is to avoid contention on the bus. The value written to this field also determines the number of cycles that will be inserted between asynchronous accesses and SDRAM accesses. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field. If more turnaround cycles are required than can be programmed into the TA field, additional cycles can be added to the R_HOLD field to compensate.</p>
ASIZE	<p><b>Asynchronous Device Bus Width.</b></p> <p>This field determines the data bus width of the asynchronous interface in the following way:</p> <ul style="list-style-type: none"> <li>ASIZE = 0 selects an 8-bit bus</li> <li>ASIZE = 1 selects a 16-bit bus</li> </ul> <p>The configuration of ASIZE determines the function of the EMA_A and EMA_BA pins as described in Section 16.2.5.1. This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in Section 16.2.2. For example, a request for a 32-bit word would require four external access when ASIZE = 0. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.</p>

<sup>(2)</sup> When using a 16 bit NAND device with ECC calculation support enabled via the EMIFA controller, the EMIFA ECC engine requires the read data to be stable for 2 clock cycles to calculate ECC. The ECC latches 8 bits of data at a time, therefore it requiring additional 2 clock cycles to complete the calculation for 16-bit data. For reliable 16-Bit NAND operations R\_HOLD must be set to 1 to ensure 2 clock cycles for ECC calculation.

**Table 16-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC)<sup>(1)</sup>**

Parameter	Description
WP <sub><i>n</i></sub>	<p><b>EMA_WAIT Polarity.</b></p> <ul style="list-style-type: none"> <li>WP<sub><i>n</i></sub> = 0 selects active-low polarity</li> <li>WP<sub><i>n</i></sub> = 1 selects active-high polarity</li> </ul> <p>When set to 1, the EMIFA will wait if the EMA_WAIT pin is high. When cleared to 0, the EMIFA will wait if the EMA_WAIT pin is low. The EMIFA must have the Extended Wait Mode enabled for the EMA_WAIT pin to affect the width of the strobe period. The polarity of the EMA_WAIT signal is not programmable in NAND Flash Mode.</p>

<sup>(1)</sup> The EMA\_WAIT pin is not available on all devices; therefore, this register is reserved on those devices.

**Table 16-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC)<sup>(1)</sup>**  
(continued)

Parameter	Description
MAX_EXT_WAIT	<p><b>Maximum Extended Wait Cycles.</b></p> <p>This field configures the number of EMIFA clock cycles the EMIFA will wait for the EMA_WAIT pin to be deactivated during the strobe period of an access cycle. The maximum number of EMIFA clock cycles it will wait is determined by the following formula:            Maximum Extended Wait Cycles = (MAX_EXT_WAIT + 1) × 16</p> <p>If the EMA_WAIT pin is not deactivated within the time specified by this field, the EMIFA resumes the access cycle, registering whatever data is on the bus and proceeding to the hold period of the access cycle. This situation is referred to as an Asynchronous Timeout. An Asynchronous Timeout generates an interrupt, if it has been enabled in the EMIFA interrupt mask set register (INTMSKSET). Refer to <a href="#">Section 16.2.8.1</a> for more information about the EMIFA interrupts. Extended Wait Mode should not be used while in NAND Flash Mode.</p>

**Table 16-17. Description of the EMIFA Interrupt Mask Set Register (INTMSKSET)**

Parameter	Description
WR_MASK_SET	<b>Wait Rise Mask Set.</b> Writing a 1 enables an interrupt to be generated when a rising edge on EMA_WAIT <sup>(1)</sup> occurs while in NAND Flash Mode
AT_MASK_SET	<b>Asynchronous Timeout Mask Set.</b> Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

<sup>(1)</sup> The EMA\_WAIT pin is not available on all devices; therefore, this field is reserved on those devices.

**Table 16-18. Description of the EMIFA Interrupt Mast Clear Register (INTMSKCLR)**

Parameter	Description
WR_MASK_CLR	<b>Wait Rise Mask Clear.</b> Writing a 1 to this bit disables the interrupt, clearing the WR_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).
AT_MASK_CLR	<b>Asynchronous Timeout Mask Clear.</b> Writing a 1 to this bit prevents an interrupt from being generated when an Asynchronous Timeout occurs.

#### 16.2.5.4 Read and Write Operations in Normal Mode

Normal Mode is the asynchronous interface's default mode of operation. It is selected when the SS bit in the asynchronous *n* configuration register (CE<sub>n</sub>CFG) is cleared to 0. In this mode, the EMA\_WE\_DQM pins operate as byte enables. [Section 16.2.5.4.1](#) and [Section 16.2.5.4.2](#) explain the details of read and write operations while in Normal Mode.

##### 16.2.5.4.1 Asynchronous Read Operations (Normal Mode)

**NOTE:** During the entirety of an asynchronous read operation, the EMA\_WE pin is driven high.

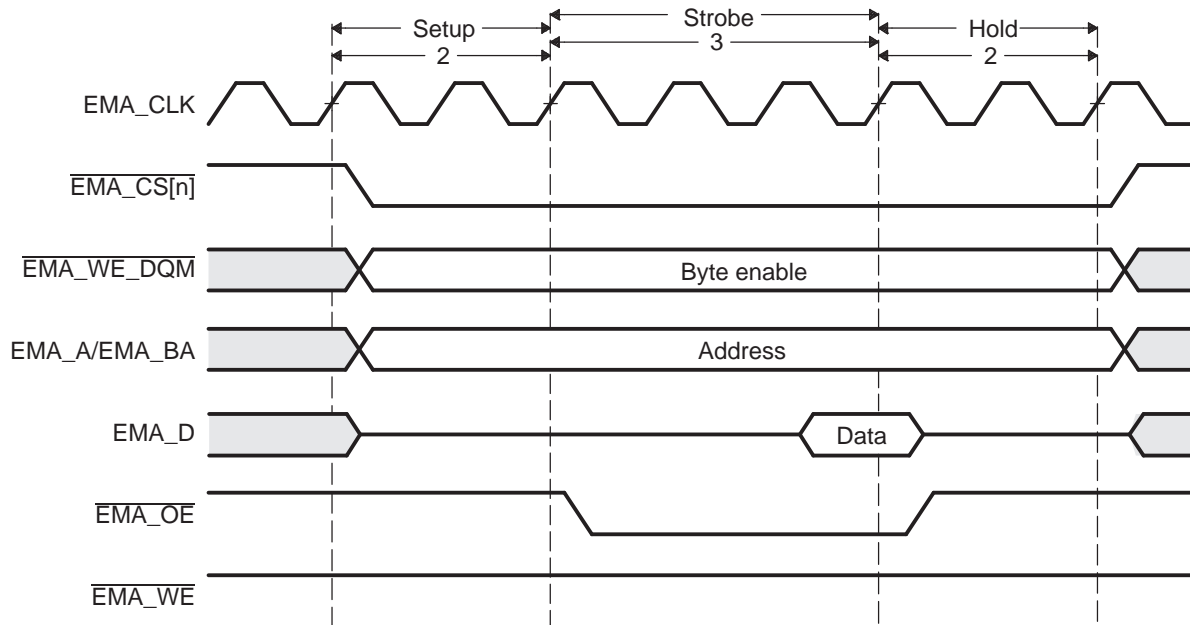
An asynchronous read is performed when any of the requesters mentioned in [Section 16.2.2](#) request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in [Section 16.2.12](#). In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous read operation in Normal Mode are described in [Table 16-19](#). Also, [Figure 16-10](#) shows an example timing diagram of a basic read operation.

**Table 16-19. Asynchronous Read Operation in Normal Mode**

Time Interval	Pin Activity in Normal Mode
Turnaround period	Once the read operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (CE <sub>n</sub> CFG). There are two exceptions to this rule: <ul style="list-style-type: none"> <li>If the current read operation was directly preceded by another read operation to the same chip select, no turnaround cycles are inserted.</li> </ul> After the EMIFA has waited for the turnaround cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.

**Table 16-19. Asynchronous Read Operation in Normal Mode (continued)**

Time Interval	Pin Activity in Normal Mode
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> <li>The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in CE<sub>n</sub>CFG.</li> <li>The address pins EMA_A and EMA_BA become valid and carry the values described in <a href="#">Section 16.2.5.1</a>.</li> <li>EMA_CS[5:2] falls to enable the external device (if not already low from a previous operation)</li> </ul>
Strobe period	<p>The following actions occur during the strobe period of a read operation:</p> <ol style="list-style-type: none"> <li>EMA_OE falls at the start of the strobe period</li> <li>On the rising edge of the clock which is concurrent with the end of the strobe period:                             <ul style="list-style-type: none"> <li>EMA_OE rises</li> <li>The data on the EMA_D bus is sampled by the EMIFA.</li> </ul> </li> </ol> <p>In <a href="#">Figure 16-10</a>, EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. <a href="#">Section 16.2.5.7</a> contains more details on using the EMA_WAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> <li>The address pins EMA_A and EMA_BA become invalid</li> <li>EMA_CS[5:2] rises (if no more operations are required to complete the current request)</li> </ul> <p>EMIFA may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turnaround period for the pending read or write operation.</p>

**Figure 16-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode**


### 16.2.5.4.2 Asynchronous Write Operations (Normal Mode)

**NOTE:** During the entirety of an asynchronous write operation, the  $\overline{\text{EMA\_OE}}$  pin is driven high.

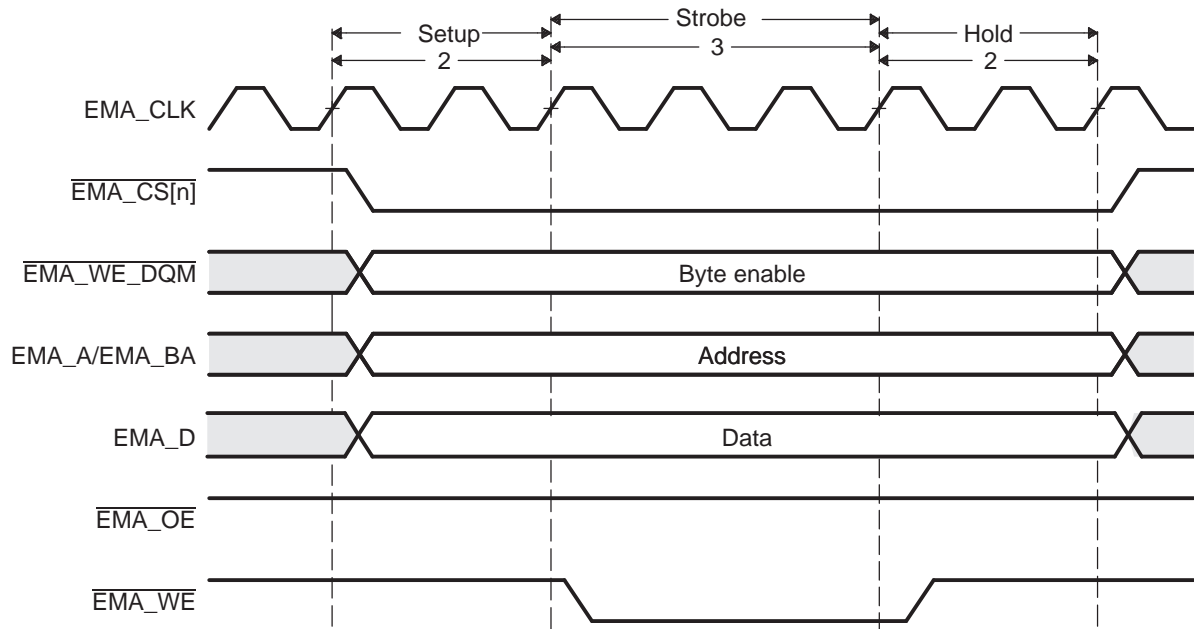
An asynchronous write is performed when any of the requesters mentioned in [Section 16.2.2](#) request a write to memory in the asynchronous bank of the EMIFA. After the request is received, a write operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in [Section 16.2.12](#). In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous write operation in Normal Mode are described in [Table 16-20](#). Also, [Figure 16-11](#) shows an example timing diagram of a basic write operation.

**Table 16-20. Asynchronous Write Operation in Normal Mode**

Time Interval	Pin Activity in Normal Mode
Turnaround period	Once the write operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (CE <sub>n</sub> CFG). There are two exceptions to this rule: <ul style="list-style-type: none"> <li>If the current write operation was directly preceded by another write operation to the same chip select, no turn-around cycles are inserted.</li> </ul> After the EMIFA has waited for the turn-around cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: <ul style="list-style-type: none"> <li>The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in CE<sub>n</sub>CFG.</li> <li>The address pins EMA_A and EMA_BA and the data pins EMA_D become valid. The EMA_A and EMA_BA pins carry the values described in <a href="#">Section 16.2.5.1</a>.</li> <li><math>\overline{\text{EMA\_CS}}[5:2]</math> falls to enable the external device (if not already low from a previous operation).</li> </ul>
Strobe period	The following actions occur at the start of the strobe period of a write operation: <ol style="list-style-type: none"> <li><math>\overline{\text{EMA\_WE}}</math> falls</li> <li>The <math>\overline{\text{EMA\_WE\_DQM}}</math> pins become valid as byte enables.</li> </ol> The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: <ol style="list-style-type: none"> <li><math>\overline{\text{EMA\_WE}}</math> rises</li> <li>The <math>\overline{\text{EMA\_WE\_DQM}}</math> pins deactivate</li> </ol> In <a href="#">Figure 16-11</a> , EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. <a href="#">Section 16.2.5.7</a> contains more details on using the EMA_WAIT pin.
End of the hold period	At the end of the hold period: <ul style="list-style-type: none"> <li>The address pins EMA_A and EMA_BA become invalid</li> <li>The data pins become invalid</li> <li><math>\overline{\text{EMA\_CS}}[n]</math> (<i>n</i> = 2, 3, 4, or 5) rises (if no more operations are required to complete the current request)</li> </ul> The EMIFA may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turnaround period for the pending read or write operation.



**Figure 16-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode**



### 16.2.5.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIFA's second mode of operation. It is selected when the SS bit of the asynchronous  $n$  configuration register (CE $n$ CFG) is set to 1. In this mode, the EMA\_WE\_DQM pins operate as byte enables and the EMA\_CS[n] ( $n = 2, 3, 4, \text{ or } 5$ ) pin is only active during the strobe period of an access cycle. [Section 16.2.5.4.1](#) and [Section 16.2.5.4.2](#) explain the details of read and write operations while in Select Strobe Mode.

#### 16.2.5.5.1 Asynchronous Read Operations (Select Strobe Mode)

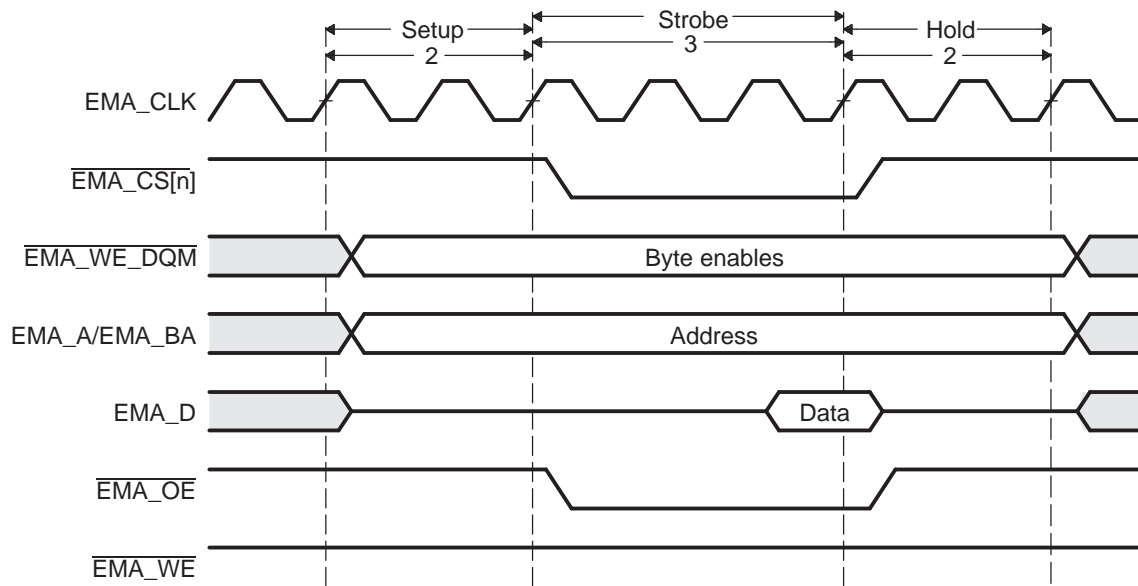
**NOTE:** During the entirety of an asynchronous read operation, the EMA\_WE pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in [Section 16.2.2](#) request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in [Section 16.2.12](#). In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous read operation in Select Strobe Mode are described in [Table 16-21](#). Also, [Figure 16-12](#) shows an example timing diagram of a basic read operation.

**Table 16-21. Asynchronous Read Operation in Select Strobe Mode**

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	<p>Once the read operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <math>n</math> configuration register (CE<math>n</math>CFG). There are two exceptions to this rule:</p> <ul style="list-style-type: none"> <li>If the current read operation was directly preceded by another read operation to the same chip select, no turn-around cycles are inserted.</li> </ul> <p>After the EMIFA has waited for the turn-around cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.</p>
Start of the setup period	<p>The following actions occur at the start of the setup period:</p> <ul style="list-style-type: none"> <li>The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in CE<math>n</math>CFG.</li> <li>The address pins EMA_A and EMA_BA become valid and carry the values described in <a href="#">Section 16.2.5.1</a>.</li> <li>The EMA_WE_DQM pins become valid as byte enables.</li> </ul>
Strobe period	<p>The following actions occur during the strobe period of a read operation:</p> <ol style="list-style-type: none"> <li>EMA_CS[n] (<math>n = 2, 3, 4, \text{ or } 5</math>) and EMA_OE fall at the start of the strobe period</li> <li>On the rising edge of the clock which is concurrent with the end of the strobe period: <ul style="list-style-type: none"> <li>EMA_CS[n] (<math>n = 2, 3, 4, \text{ or } 5</math>) and EMA_OE rise</li> <li>The data on the EMA_D bus is sampled by the EMIFA.</li> </ul> </li> </ol> <p>In <a href="#">Figure 16-12</a>, EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. <a href="#">Section 16.2.5.7</a> contains more details on using the EMA_WAIT pin.</p>
End of the hold period	<p>At the end of the hold period:</p> <ul style="list-style-type: none"> <li>The address pins EMA_A and EMA_BA become invalid</li> <li>The EMA_WE_DQM pins become invalid</li> </ul> <p>The EMIFA may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turnaround period for the pending read or write operation.</p>

**Figure 16-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode**



### 16.2.5.5.2 Asynchronous Write Operations (Select Strobe Mode)

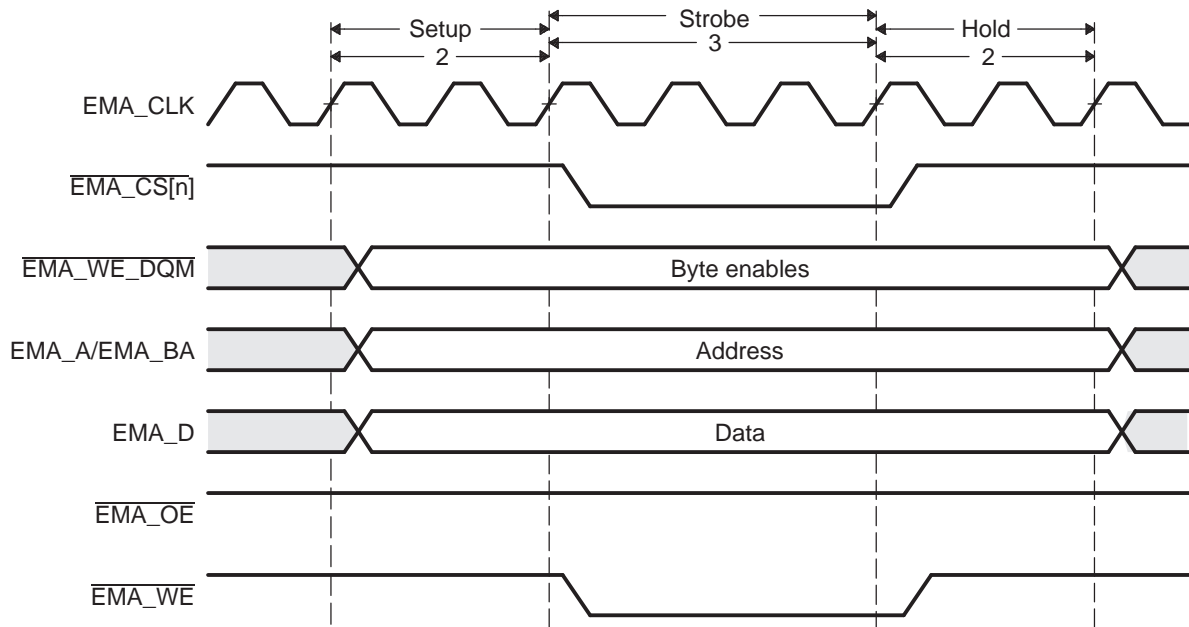
**NOTE:** During the entirety of an asynchronous write operation, the  $\overline{\text{EMA\_OE}}$  pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in [Section 16.2.2](#) request a write to memory in the asynchronous bank of the EMIFA. After the request is received, a write operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in [Section 16.2.12](#). In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous write operation in Select Strobe Mode are described in [Table 16-22](#). Also, [Figure 16-13](#) shows an example timing diagram of a basic write operation.

**Table 16-22. Asynchronous Write Operation in Select Strobe Mode**

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	Once the write operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous <i>n</i> configuration register (CE <sub>n</sub> CFG). There are two exceptions to this rule: <ul style="list-style-type: none"> <li>If the current write operation was directly preceded by another write operation to the same chip select, no turn-around cycles are inserted.</li> </ul> After the EMIFA has waited for the turnaround cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: <ul style="list-style-type: none"> <li>The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in CE<sub>n</sub>CFG.</li> <li>The address pins EMA_A and EMA_BA and the data pins EMA_D become valid. The EMA_A and EMA_BA pins carry the values described in <a href="#">Section 16.2.5.1</a>.</li> <li>The <math>\overline{\text{EMA\_WE\_DQM}}</math> pins become active as byte enables.</li> </ul>
Strobe period	The following actions occur at the start of the strobe period of a write operation: <ul style="list-style-type: none"> <li><math>\overline{\text{EMA\_CS}}[n]</math> (<i>n</i> = 2, 3, 4, or 5) and <math>\overline{\text{EMA\_WE}}</math> fall</li> </ul> The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: <ul style="list-style-type: none"> <li><math>\overline{\text{EMA\_CS}}[n]</math> (<i>n</i> = 2, 3, 4, or 5) and <math>\overline{\text{EMA\_WE}}</math> rise</li> </ul> In <a href="#">Figure 16-13</a> , EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. <a href="#">Section 16.2.5.7</a> contains more details on using the EMA_WAIT pin.
End of the hold period	At the end of the hold period: <ul style="list-style-type: none"> <li>The address pins EMA_A and EMA_BA become invalid</li> <li>The data pins become invalid</li> <li>The <math>\overline{\text{EMA\_WE\_DQM}}</math> pins become invalid</li> </ul> The EMIFA may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turn-around period for the pending read or write operation.

**Figure 16-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode**



### 16.2.5.6 NAND Flash Mode

NAND Flash Mode is a submode of both Normal Mode and Select Strobe Mode. Chip select  $\overline{\text{EMA\_CS}}[n]$  ( $n = 2, 3, 4, \text{ or } 5$ ) may be placed in NAND Flash mode by setting the  $\text{CS}_n\text{NAND}$  ( $n = 2, 3, 4, \text{ or } 5$ ) bit in the NAND Flash control register (NANDFCR). [Table 16-23](#) displays the bit fields present in NANDFCR and briefly describes their use.

When a chip select space is configured to operate in NAND Flash mode, the EMIFA hardware can calculate the error correction code (ECC) for each 518 byte data transfer to that chip select space. The EMIFA hardware will not generate the NAND access cycle, which includes the command, address, and data phases, necessary to complete a transfer to NAND Flash. All NAND Flash operations can be divided into single asynchronous cycles, and with the help of software the EMIFA can execute a complete NAND access cycle.

**Table 16-23. Description of the NAND Flash Control Register (NANDFCR)**

Parameter	Description
CS5ECC	<b>NAND Flash ECC state for <math>\overline{\text{EMA\_CS}}[5]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to start an ECC calculation for <math>\overline{\text{EMA\_CS}}[5]</math></li> <li>Cleared to 0 when NAND Flash 4 ECC register (NANDF4ECC) is read.</li> </ul>
CS5NAND	<b>NAND Flash mode for <math>\overline{\text{EMA\_CS}}[5]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to enable NAND Flash mode for <math>\overline{\text{EMA\_CS}}[5]</math></li> </ul>
CS4ECC	<b>NAND Flash ECC state for <math>\overline{\text{EMA\_CS}}[4]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to start an ECC calculation for <math>\overline{\text{EMA\_CS}}[4]</math></li> <li>Cleared to 0 when NAND Flash 3 ECC register (NANDF3ECC) is read.</li> </ul>
CS4NAND	<b>NAND Flash mode for <math>\overline{\text{EMA\_CS}}[4]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to enable NAND Flash mode for <math>\overline{\text{EMA\_CS}}[4]</math></li> </ul>
CS3ECC	<b>NAND Flash ECC state for <math>\overline{\text{EMA\_CS}}[3]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to start an ECC calculation for <math>\overline{\text{EMA\_CS}}[3]</math></li> <li>Cleared to 0 when NAND Flash 2ECC register (NANDF2ECC) is read.</li> </ul>
CS3NAND	<b>NAND Flash mode for <math>\overline{\text{EMA\_CS}}[3]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to enable NAND Flash mode for <math>\overline{\text{EMA\_CS}}[3]</math></li> </ul>
CS2ECC	<b>NAND Flash ECC state for <math>\overline{\text{EMA\_CS}}[2]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to start an ECC calculation for <math>\overline{\text{EMA\_CS}}[2]</math></li> <li>Cleared to 0 when NAND Flash 1 ECC register (NANDF1ECC) is read.</li> </ul>
CS2NAND	<b>NAND Flash mode for <math>\overline{\text{EMA\_CS}}[2]</math>.</b> <ul style="list-style-type: none"> <li>Set to 1 to enable NAND Flash mode for <math>\overline{\text{EMA\_CS}}[2]</math></li> </ul>

#### 16.2.5.6.1 Configuring for NAND Flash Mode

Similar to the asynchronous accesses previously described, the EMIFA's memory-mapped registers must be programmed appropriately to interface to a NAND Flash device. In addition to the fields listed in [Table 16-15](#), the  $\text{CS}_n\text{NAND}$  ( $n = 2, 3, 4, \text{ or } 5$ ) bit of the NAND Flash control register (NANDFCR) should be set to 1 to enter NAND Flash Mode. Note that the EW bit of  $\text{CE}_n\text{CFG}$  should be cleared to avoid enabling the wait feature while in NAND Flash Mode.

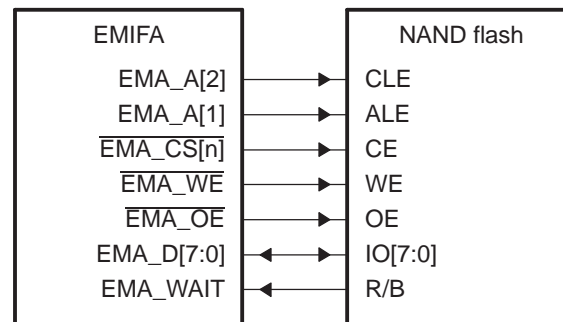
#### 16.2.5.6.2 Connecting to NAND Flash

[Figure 16-14](#) shows the EMIFA external pins used to interface with a NAND Flash device. EMIFA address lines are used to drive the NAND Flash device's command latch enable (CLE) and address latch enable (ALE) signals. Any EMIFA address lines may be used to drive the CLE and ALE signals of the NAND Flash.

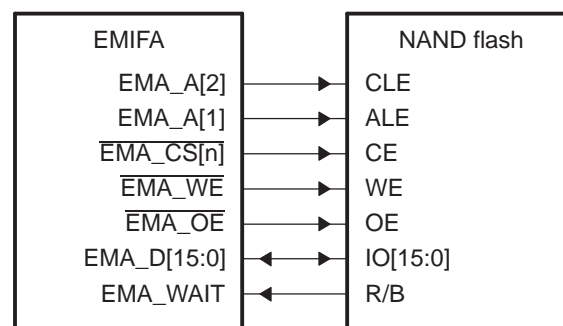
---

**NOTE:** The EMIFA will not control the NAND Flash device's write protect pin. The write protect pin must be controlled outside of the EMIFA.

---

**Figure 16-14. EMIFA to NAND Flash Interface**


a) Connection to 8-bit NAND device



b) Connection to 16-bit NAND device

### 16.2.5.6.3 Driving CLE and ALE

As stated in [Section 16.2.5.1](#), the EMIFA always drives the least significant bit of a 32-bit word address on EMA\_A[0]. This functionality must be considered when attempting to drive the offset lines connected to CLE and ALE to the appropriate state.

For example, if using EMA\_A[2] and EMA\_A[1] to connect to CLE and ALE, respectively, the following offsets should be added to EMIFA base address:

- 0000 0000h to drive CLE and ALE low
- 0000 0010h to drive CLE high and ALE low
- 0000 0008h to drive CLE low and ALE high

### 16.2.5.6.4 NAND Read and Program Operations

A NAND Flash access cycle is composed of a command, address, and data phase. The EMIFA will not automatically generate these three phases to complete a NAND access with one transfer request. To complete a NAND access cycle, multiple single asynchronous access cycles must be completed by the EMIFA. Software must be used to request the appropriate asynchronous accesses to complete a NAND Flash access cycle. This software must be developed to the specification of the chosen NAND Flash device.

Since NAND operations are divided into single asynchronous access cycles, the chip select signal will not remain activated for the duration of the NAND operation. Instead, the chip select signal will deactivate between each asynchronous access cycle. For this reason, the EMIFA does not support NAND Flash devices that require the chip select signal to remain low during the  $t_r$  time for a read. See [Section 16.2.5.6.8](#) for workaround.

Care must be taken when performing a NAND read or write operation via the EDMA controller. See [Section 16.2.5.6.5](#) for more details.

---

**NOTE:** The EMIFA does not support NAND Flash devices that require the chip select signal to remain low during the  $t_r$  time for a read. See [Section 16.2.5.6.8](#) for workaround.

---

#### 16.2.5.6.5 NAND Data Read and Write via EDMA Controller

When performing NAND accesses, the EDMA controller is most efficiently used for the data phase of the access. The command and address phases of the NAND access require only a few words of data to be transferred and therefore do not take advantage of the EDMA controller's ability to transfer larger quantities of data with a single request. In this section we will focus on using the EDMA controller for the data phase of a NAND access.

There are two conditions that require care to be taken when performing NAND reads and writes via the EDMA controller. These are:

- The address lines used to drive CLE and ALE signals must be driven low
- The EMIFA does not support constant addressing mode

Since the EMIFA does not support a constant addressing mode, when programming the EDMA, a linear incrementing address mode must be used. When using a linear incrementing address mode, if the CLE and ALE are driven by EMA\_A[2] and EMA\_A[1], respectively, care must be taken not to increase the address into a range that drives CLE and/or ALE high. To prevent the address from incrementing into a range that drives CLE and/or ALE high, the EDMA ACNT, BCNT, SIDX, DIDX, and synchronization type must be programmed appropriately. Following is an example configuration of EDMA controller when EMA\_A[2] is connected to CLE and EMA\_A[1] is connected to ALE.

EDMA setup for a NAND Flash data read:

- ACNT  $\leq$  8 bytes (this can also be set to less than or equal to the external data bus width)
- BCNT = transfer size in bytes/ACNT
- SIDX (source index) = 0
- DIDX (destination index) = ACNT
- AB synchronized

EDMA setup for a NAND Flash data write:

- ACNT  $\leq$  8 bytes (this can also be set to less than or equal to the external data bus width)
- BCNT = transfer size in bytes/ACNT
- SIDX (source index) = ACNT
- DIDX (destination index) = 0
- AB synchronized

#### 16.2.5.6.6 ECC Generation

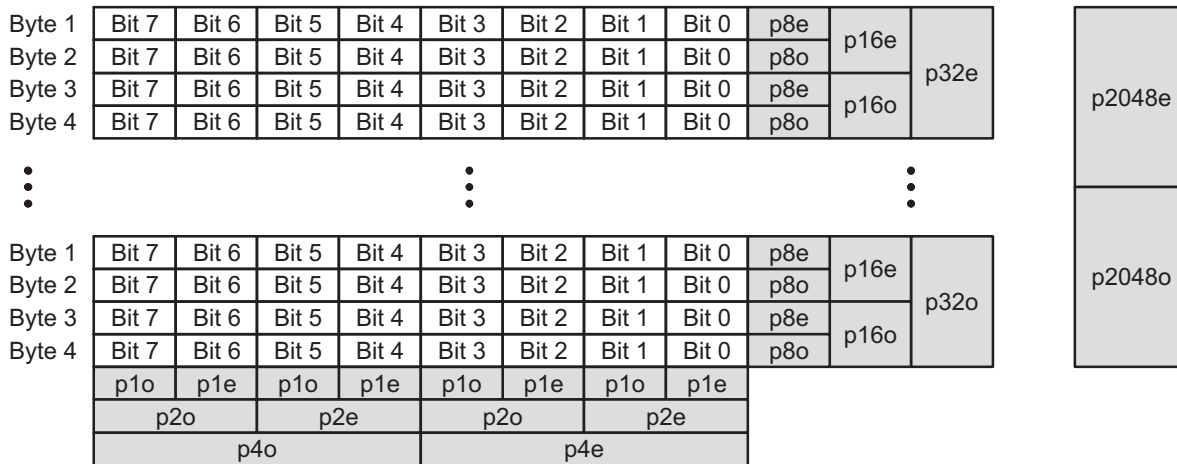
##### 16.2.5.6.6.1 1-Bit ECC

If the CS $n$ NAND ( $n = 2, 3, 4,$  or  $5$ ) bit in the NAND Flash control register (NANDFCR) is set to 1, the EMIFA supports 1-bit ECC calculation for up to 512 bytes for the corresponding chip select. To perform the ECC calculation, the CS $n$ ECC ( $n = 2, 3, 4,$  or  $5$ ) bit in NANDFCR must be set to 1. It is the responsibility of the software to start the ECC calculation by writing to the CS $n$ ECC ( $n = 2, 3, 4,$  or  $5$ ) bit prior to issuing a write or read to NAND Flash. It is also the responsibility of the software to read the calculated ECC from the NAND Flash  $m$  ECC register (NANDF $m$ ECC) ( $m = 1, 2, 3,$  or  $4$ ) once the transfer to NAND Flash has completed. If the software writes or reads more than 512 bytes, the ECC will be incorrect. Reading the NAND $m$ ECC ( $m = 1, 2, 3,$  or  $4$ ) clears the CS $n$ ECC ( $n = 2, 3, 4,$  or  $5$ ) bit in NANDFCR. The NANDF $m$ ECC ( $m = 1, 2, 3,$  or  $4$ ) is cleared upon writing a 1 to the CS $n$ ECC ( $n = 2, 3, 4,$  or  $5$ ) bit. [Figure 16-15](#) shows the algorithm used to calculate the ECC value for an 8-bit NAND Flash.



For an 8-bit NAND Flash p1o through p4e are column parities and p8e through p2048o are row parities. Similarly, the algorithm can be extended to a 16-bit NAND Flash. For a 16-bit NAND Flash p1o through p8e are column parities and p16e through p2048o are row parities. The software must ignore the unwanted parity bits if ECC is desired for less than 512 bytes of data. For example, p2048e and p2048o are not required for ECC on 256 bytes of data. Similarly, p1024e, p1024o, p2048e, and p2048o are not required for ECC on 128 bytes of data.

**Figure 16-15. ECC Value for 8-Bit NAND Flash**



**16.2.5.6.6.2 4-Bit ECC**

The EMIFA supports 4-bit ECC on 8-bit/16-bit NAND Flash. In NAND mode, if the NAND Flash 4-bit ECC start bit (4BITECC\_START) in the in the NAND Flash control register (NANDFCR) is set, the EMIFA calculates 4-bit ECC for the selected chip select. Only one chip select can be selected for the 4-bit ECC calculation at one time. The selection of the chip select is done by programming the 4-bit ECC CS select bit field (4BITECCSEL) in the NAND Flash control register (NANDFCR). The calculated parity (for writes) and syndrome (for reads) can be read from the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]). The 4-bit ECC start bit (4BITECC\_START) is cleared upon reading any of the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]). The NAND Flash 4-bit ECC 1-4 registers are cleared upon writing one to the 4-bit ECC start bit (4BITECC\_START).

The 4-bit ECC algorithm works on a 10-bit data bus, but only the lower eight bits of the data bus actually contain data. When the EMIFA is used in 16-bit mode, the lower and upper 8-bits of the 16-bit data read from the data bus are fed into the ECC engine one at a time, in that order. In all cases, since only 8-bits of data are fed to the ECC engine, the upper two bits of the 10-bit data bus that feeds the ECC engine are always zero. However, the parity and the syndrome value read from the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]) are 10 bits wide. It is the responsibility of software to convert 10-bit parity values to 8 bits before writing to the spare location of the NAND Flash after a write operation. Similarly, it is the responsibility of the software to convert the 8-bit parity values read from the spare location of the NAND Flash after a read operation, to 10 bits before writing the NAND Flash 4-bit ECC load register (NAND4BITECCLOAD).

The 4-bit ECC employed in the EMIFA interface is a Reed-Solomon error correcting code. The symbol size is ten bits (two bits are always zero and eight bits contain data as described above). With eight 10-bit parity words, up to four symbols can be corrected per block read. Though the ECC operation is called 4-bit, it is important to note that correction can actually happen on up to four 10-bit symbols. Only the lower eight bits of each 10-bit symbol actually contain data (see above), so correction can happen on up to four bytes. When bit errors are randomly distributed through the block of data read from the NAND, those errors are not likely to fall into the same bytes of data, so 4-bits of correction is an apt description. Technically speaking, however, more than four bits of error can be corrected if multiple bit errors are confined to four or fewer bytes of the data. If bit errors fall into more than four bytes, the ECC engine will report that there are too many errors to correct.

At the end of the syndrome calculation after read, the error address and the error value can be calculated by setting the address and error value calculation start bit (4BITECC\_ADD\_CALC\_START) in the NAND Flash control register (NANDFCR). The end of address calculation is flagged by the 4-bit ECC correction state field (ECC\_STATE) in the NAND Flash status register (NANDFSR). The number of errors can be read from the 4-bit number of errors field (ECC\_ERRNUM) in the NAND Flash status register (NANDFSR). The error address value can be read from the NAND Flash error address 1-2 registers (NANDERRADD[2:1]). The error value can be read from the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]). The address and error value start bit (4BITECC\_ADD\_CALC\_START) is cleared upon reading any of the NAND Flash error address 1-2 registers (NANDERRADD[2:1]) or the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]). The EMIFA registers the syndrome value internally before the error address and error value calculation. Therefore, a new read operation can be performed simultaneously with the error address calculation.

The EMIFA supports 4-bit ECC calculation up to 518 bytes. The software needs to follow the following procedure for 4-bit ECC calculation:

For writes:

1. Set the 4BITECC\_START bit in the NAND Flash control register (NANDFCR) to 1.
2. Write 518 bytes of data to the NAND Flash.
3. Read the parity from the NAND Flash 4-Bit ECC 1-4 registers (NAND4BITECC[4:1]).
4. Convert the 10-bit parity values to 8-bits. All 10-bit parity values can be concatenated together with ECC value 1 (4BITECCVAL1) as LSB and ECC value 8 (4BITECCVAL8) as MSB. Then the concatenated value can be broken down into ten 8-bit values.
5. Store the parity to spare location in the NAND Flash.

For reads:

1. Set the 4BITECC\_START bit in the NAND Flash control register (NANDFCR) to 1.
2. Read 518 bytes of data from the NAND Flash.
3. Clear the 4BITECC\_START bit in NANDFCR by reading any of the NAND Flash 4-bit ECC registers.
4. Read the parity stored in the spare location in the NAND Flash.
5. Convert the 8-bit parity values to 10-bits. Reverse of the conversion that was done during writes.
6. Write the parity values in the NAND Flash 4-bit ECC load register (NAND4BITECCLOAD). Write each parity value one at a time starting from 4BITECCVAL8 down to 4BITECCVAL1.
7. Perform a dummy read to the NAND Flash status register (NANDFSR). This is only required to ensure time for syndrome calculation after writing the ECC values in step 6.
8. Read the syndrome from the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]). A syndrome value of 0 means no bit errors. If the syndrome is non-zero, continue with step 9.
9. Set the 4BITECC\_ADD\_CALC\_START bit in the NAND Flash control register (NANDFCR) to 1.
10. Perform a dummy read to any EMIFA registers except the NAND Flash error address 1-2 registers (NANDERRADD[2:1]) or the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]).
11. Start another read from NAND, if required (a new thread from step 1).
12. Wait for the 4-bit ECC correction state field (ECC\_STATE) in the NAND Flash status register (NANDFSR) to be equal to 1, 2h, or 3h.
13. The number of errors can be read from the 4-bit number of errors field (ECC\_ERRNUM) in the NAND Flash status register (NANDFSR).
14. Read the error address from the NAND Flash error address 1-2 registers (NANDERRADD[2:1]). Address for the error word is equal to (total\_words\_read + 7 - address\_value). For 518 bytes, the address will be equal to (525 - address\_value).
15. Read the error value from the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]). Errors can be corrected by XORing the error word with the error value from the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]).

### 16.2.5.6.7 NAND Flash Status Register (NANDFSR)

The NAND Flash status register (NANDFSR) indicates the raw status of the EMA\_WAIT pin while in NAND Flash Mode. The EMA\_WAIT pin should be connected to the NAND Flash device's R/ $\bar{B}$  signal, so that it indicates whether or not the NAND Flash device is busy. During a read, the R/ $\bar{B}$  signal will transition and remain low while the NAND Flash retrieves the data requested. Once the R/ $\bar{B}$  signal transitions high, the requested data is ready and should be read by the EMIFA. During a write/program operation, the R/ $\bar{B}$  signal transitions and remains low while the NAND Flash is programming the Flash with the data it has received from the EMIFA. Once the R/ $\bar{B}$  signal transitions high, the data has been written to the Flash and the next phase of the transaction may be performed. From this explanation, you can see that the NAND Flash status register is useful to the software for indicating the status of the NAND Flash device and determining when to proceed to the next phase of a NAND Flash operation.

When a rising edge occurs on the EMA\_WAIT pin, the EMIFA sets the WR (Wait Rise) bit in the EMIFA interrupt raw register (INTRAW). Therefore, the EMIFA Wait Rise interrupt may be used to indicate the status of the NAND Flash device. The WP $n$  bit in the asynchronous wait cycle configuration register (AWCC) does not affect the NAND Flash status register (NANDFSR) or the WR bit in INTRAW. See [Section 16.2.8](#) for more a detailed description of the wait rise interrupt.

### 16.2.5.6.8 Interfacing to a Non-CE Don't Care NAND Flash

As explained in [Section 16.2.5.6.4](#), the EMIFA does not support NAND Flash devices that require the chip select signal to remain low during the  $t_r$  time for a read. One way to work around this limitation is to use a GPIO pin to drive the  $\bar{C}E$  signal of the NAND Flash device. If this work around is implemented, software will configure the selected GPIO to be low, then begin the NAND Flash operation, starting with the command phase. Once the NAND Flash operation has completed the software can then configure the selected GPIO to be high.

### 16.2.5.7 Extended Wait Mode and the EMA\_WAIT Pin

The EMIFA supports the Extend Wait Mode. This is a mode in which the external asynchronous device may assert control over the length of the strobe period. The Extended Wait Mode can be entered by setting the EW bit in the asynchronous  $n$  configuration register (CE $n$ CFG) ( $n = 2, 3, 4, \text{ or } 5$ ). When this bit is set, the EMIFA monitors the EMA\_WAIT pin to determine if the attached device wishes to extend the strobe period of the current access cycle beyond the programmed number of clock cycles.

When the EMIFA detects that the EMA\_WAIT pin has been asserted, it will begin inserting extra strobe cycles into the operation until the EMA\_WAIT pin is deactivated by the external device. The EMIFA will then return to the last cycle of the programmed strobe period and the operation will proceed as usual from this point. Please refer to the device data manual for details on the timing requirements of the EMA\_WAIT signal.

The EMA\_WAIT pin cannot be used to extend the strobe period indefinitely. The programmable MAX\_EXT\_WAIT field in the asynchronous wait cycle configuration register (AWCC) determines the maximum number of EMA\_CLK cycles the strobe period may be extended beyond the programmed length. When the counter expires, the EMIFA proceeds to the hold period of the operation regardless of the state of the EMA\_WAIT pin. The EMIFA can also generate an interrupt upon expiration of this counter. See [Section 16.2.8.1](#) for details on enabling this interrupt.

For the EMIFA to function properly in the Extended Wait mode, the WP $n$  bit of AWCC must be programmed to match the polarity of the EMA\_WAIT pin. In its reset state of 1, the EMIFA will insert wait cycles when the EMA\_WAIT pin is sampled high. When set to 0, the EMIFA will insert wait cycles only when EMA\_WAIT is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

Finally, a restriction is placed on the strobe period timing parameters when operating in Extended Wait mode. Specifically, the sum of the W\_SETUP and W\_STROBE fields must be greater than 4, and the sum of the R\_SETUP and R\_STROBE fields must be greater than 4 for the EMIFA to recognize the EMA\_WAIT pin has been asserted. The W\_SETUP, W\_STROBE, R\_SETUP, and R\_STROBE fields are in CE $n$ CFG.

### 16.2.6 Data Bus Parking

The EMIFA always drives the data bus to the previous write data value when it is idle. This feature is called data bus parking. Only when the EMIFA issues a read command to the external memory does it stop driving the data bus. The data bus is released (tri-stated) when the chip enable ( $\overline{EMA\_CS[n]}$ ) is asserted by EMIFA for the read access. After the read operation is completed, the data bus is driven again by the bus parking feature at the end of the turnaround time. At all other times that the EMIF is enabled but not actively transferring data, the bus parking feature drives the data bus to the last written value.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIFA is in the self-refresh state. In this situation, the read operation is not followed by the EMIFA parking the data bus. Instead, the EMIFA tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIFA is in the self-refresh state, in order to prevent floating inputs on the data bus. External pull-ups, such as 10kΩ resistors, should be placed on the 16 EMIFA data bus pins (which do not have internal pull-ups) if it is required to perform reads in this situation. The precise resistor value should be chosen so that the worst case combined off-state leakage currents do not cause the voltage levels on the associated pins to drop below the high-level input voltage requirement.

For information about the self-refresh state, see [Section 16.2.4.7](#).

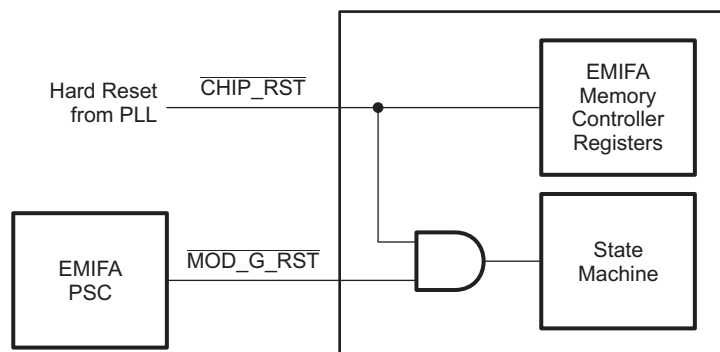
### 16.2.7 Reset and Initialization Considerations

The EMIFA memory controller has two reset signals,  $\overline{CHIP\_RST}$  and  $\overline{MOD\_G\_RST}$ . The  $\overline{CHIP\_RST}$  is a module-level reset that resets both the state machine as well as the EMIFA memory controller's memory-mapped registers. The  $\overline{MOD\_G\_RST}$  resets the state machine only. If the EMIFA memory controller is reset independently of other peripherals, the user's software should not perform memory, as well as register accesses, while  $\overline{CHIP\_RST}$  or  $\overline{MOD\_G\_RST}$  are asserted. If memory or register accesses are performed while the EMIFA memory controller is in the reset state, other masters may hang. Following the rising edge of  $\overline{CHIP\_RST}$  or  $\overline{MOD\_G\_RST}$ , the EMIFA memory controller immediately begins its initialization sequence. Command and data stored in the EMIFA memory controller FIFOs are lost. [Table 16-24](#) describes the different methods for asserting each reset signal. [Figure 16-16](#) shows the EMIFA memory controller reset diagram.

**Table 16-24. Reset Sources**

Reset Signal	Reset Source
$\overline{CHIP\_RST}$	Hardware/ Device Reset
$\overline{MOD\_G\_RST}$	Power and Sleep Controller

**Figure 16-16. EMIFA Reset Block Diagram**



The EMIFA and its registers are reset when any of the following events occur:

1. The  $\overline{\text{RESET}}$  pin on the device is asserted
2. An emulator reset is initiated through the Code Composer Studio™ integrated development environment

In the first case, the EMIFA will exit the reset state when  $\overline{\text{RESET}}$  is released and after the PLL controller releases the entire device from reset. In the second case, the EMIFA will exit the reset state immediately after the emulator reset is complete.

In both cases, the EMIFA automatically begins running the SDRAM initialization sequence described in [Section 16.2.4.4](#) after coming out of reset. Even though the initialization procedure is automatic, a special procedure, found in [Section 16.2.4.5](#) must still be followed.

## 16.2.8 Interrupt Support

The EMIFA supports a single interrupt to the CPU. [Section 16.2.8.1](#) details the generation and internal masking of EMIFA interrupts, and [Section 16.2.8.2](#) describes how the EMIFA interrupts are sent to the CPU.

### 16.2.8.1 Interrupt Events

There are three conditions that may cause the EMIFA to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EMA\_WAIT signal (wait rise interrupt)
- An asynchronous time out
- Usage of unsupported addressing mode (line trap interrupt)

The wait rise interrupt occurs when a rising edge is detected on EMA\_WAIT signal. This interrupt generation is not affected by the  $WP_n$  bit in the asynchronous wait cycle configuration register (AWCC). The asynchronous time out interrupt condition occurs when the attached asynchronous device fails to deassert the EMA\_WAIT pin within the number of cycles defined by the MAX\_EXT\_WAIT bit in AWCC (this happens only in extended wait mode). EMIFA supports only linear incrementing and cache line wrap addressing modes. If an access request for an unsupported addressing mode is received, the EMIFA will set the LT bit in the EMIFA interrupt raw register (INTRAW) and treat the request as a linear incrementing request.

Only when the interrupt is enabled by setting the appropriate bit (WR\_MASK\_SET/AT\_MASK\_SET/LT\_MASK\_SET) in the EMIFA interrupt mask set register (INTMSKSET) to 1, will the interrupt be sent to the CPU. Once enabled, the interrupt may be disabled by writing a 1 to the corresponding bit in the EMIFA interrupt mask clear register (INTMSKCLR). The bit fields in both the INTMSKSET and INTMSKCLR may be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the INTMSKSET and INTMSKCLR will have a value of 1; when the interrupt is disabled, the corresponding bit field will have a value of 0.

The EMIFA interrupt raw register (INTRAW) and the EMIFA interrupt mask register (INTMSK) indicate the status of each interrupt. The appropriate bit (WR/AT/LT) in INTRAW is set when the interrupt condition occurs, whether or not the interrupt has been enabled. However, the appropriate bit (WR\_MASKED/AT\_MASKED/LT\_MASKED) in INTMSK is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in INTRAW clears the INTRAW bit as well as the corresponding bit in INTMSK. [Table 16-25](#) contains a brief summary of the interrupt status and control bit fields. See [Section 16.4](#) for complete details on the register fields.

**Table 16-25. Interrupt Monitor and Control Bit Fields**

Register Name	Bit Name	Description
EMIFA interrupt raw register (INTRAW)	WR	This bit is set when an rising edge on the EMA_WAIT signal occurs. Writing a 1 clears the WR bit as well as the WR_MASKED bit in INTMSK.
	AT	This bit is set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the AT_MASKED bit in INTMSK.
	LT	This bit is set when an unsupported addressing mode is used. Writing a 1 clears LT bit as well as the LT_MASKED bit in INTMSK.
EMIFA interrupt mask register (INTMSK)	WR_MASKED	This bit is set only when a rising edge on the EMA_WAIT signal occurs and the interrupt has been enabled by writing a 1 to the WR_MASK_SET bit in INTMSKSET.
	AT_MASKED	This bit is set only when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the AT_MASK_SET bit in INTMSKSET.
	LT_MASKED	This bit is set only when line trap interrupt occurs and the interrupt has been enabled by writing a 1 to the LT_MASK_SET bit in INTMSKSET.
EMIFA interrupt mask set register (INTMSKSET)	WR_MASK_SET	Writing a 1 to this bit enables the wait rise interrupt.
	AT_MASK_SET	Writing a 1 to this bit enables the asynchronous timeout interrupt.
	LT_MASK_SET	Writing a 1 to this bit enables the line trap interrupt.
EMIFA interrupt mask clear register (INTMSKCLR)	WR_MASK_CLR	Writing a 1 to this bit disables the wait rise interrupt.
	AT_MASK_CLR	Writing a 1 to this bit disables the asynchronous timeout interrupt.
	LT_MASK_CLR	Writing a 1 to this bit disables the line trap interrupt.

### 16.2.8.2 Interrupt Multiplexing

For details on EMIFA interrupt multiplexing, see your device-specific data manual.

### 16.2.8.3 Interrupt Processing

For details on EMIFA interrupt processing, see the .

### 16.2.9 EDMA Event Support

EMIFA memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests may be made directly, by masters and the DMA.

### 16.2.10 Pin Multiplexing

For details on EMIFA pin multiplexing, see your device-specific data manual.

### 16.2.11 Memory Map

For information describing the device memory-map, see your device-specific data manual.



### 16.2.12 Priority and Arbitration

[Section 16.2.2](#) describes the external prioritization and arbitration among requests from different sources within the SoC. The result of this external arbitration is that only one request is presented to the EMIFA at a time. Once the EMIFA completes a request, the external arbiter then provides the EMIFA with the next pending request.

Internally, the EMIFA undertakes memory device transactions according to a strict priority scheme. The highest priority events are:

- A device reset.
- A write to any of the three least significant bytes of the SDRAM configuration register (SDCR).

Either of these events will cause the EMIFA to immediately commence its initialization sequence as described in [Section 16.2.4.4](#).

Once the EMIFA has completed its initialization sequence, it performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIFA's backlog refresh counter is at the Refresh Must urgency level, the EMIFA performs multiple SDRAM auto refresh cycles until the Refresh Release urgency level is reached.
2. If an SDRAM or asynchronous read has been requested, the EMIFA performs a read operation.
3. If the EMIFA's backlog refresh counter is at the Refresh Need urgency level, the EMIFA performs an SDRAM auto refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIFA performs a write operation.
5. If the EMIFA's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIFA performs an SDRAM auto refresh cycle.
6. If the value of the SR bit in SDCR has been set to 1, the EMIFA will enter the self-refresh state as described in [Section 16.2.4.7](#).

After taking one of the actions listed above, the EMIFA then returns to the top of the priority list to determine its next action.

Because the EMIFA does not issue auto-refresh cycles when in the self-refresh state, the above priority scheme does not apply when in this state. See [Section 16.2.4.7](#) for details on the operation of the EMIFA when in the self-refresh state.

### 16.2.13 System Considerations

This section describes various system considerations to keep in mind when operating the EMIFA.

#### 16.2.13.1 Asynchronous Request Times

In a system that interfaces to both SDRAM and asynchronous memory, the asynchronous requests must not take longer than the smaller of the following two values:

- $t_{RAS}$  (typically 120  $\mu$ s) - to avoid violating the maximum time allowed between issuing an ACTV and PRE command to the SDRAM.
- $t_{Refresh\ Rate} \times 11$  (typically 15.7  $\mu$ s  $\times$  11 = 172.7  $\mu$ s) - to avoid refresh violations on the SDRAM.

The length of an asynchronous request is controlled by multiple factors, the primary factor being the number of access cycles required to complete the request. For example, an asynchronous request for 4 bytes will require four access cycles using an 8-bit data bus and only two access cycle using a 16-bit data bus. The maximum request size that the EMIFA can be sent is 16 words, therefore the maximum number of access cycles per memory request is 64 when the EMIFA is configured with an 8-bit data bus. The length of the individual access cycles that make up the asynchronous request is determined by the programmed setup, strobe, hold, and turnaround values, but can also be extended with the assertion of the EMA\_WAIT input signal up to a programmed maximum limit. It is up to the user to make sure that an entire asynchronous request does not exceed the timing values listed above when also interfacing to an SDRAM device. This can be done by limiting the asynchronous timing parameters.

#### 16.2.13.2 Cache Fill Requests

The CPU can run code from either internal or external memory. When running code from external memory, the CPU's program cache is periodically filled with eight words (32-bytes) through a dedicated port to the EMIFA. Two system level concerns arise when filling the program cache from the EMIFA.

First, the program cache fills have the possibility of being locked out from accessing the EMIFA by a stream of higher priority requests. Therefore, care should be taken when issuing persistent requests to the EMIFA from a source such which is a high priority requester.

Second, requests to the EMIFA from the other sources risk missing their deadlines while a program cache fill from the EMIFA is in progress. This is because all other EMIFA accesses are held pending while the program cache is filled. The worst-case scenario that can arise is when a requester submits a request immediately after a program cache fill request has begun. The system should be analyzed to make sure that this worst-case request delay is acceptable.



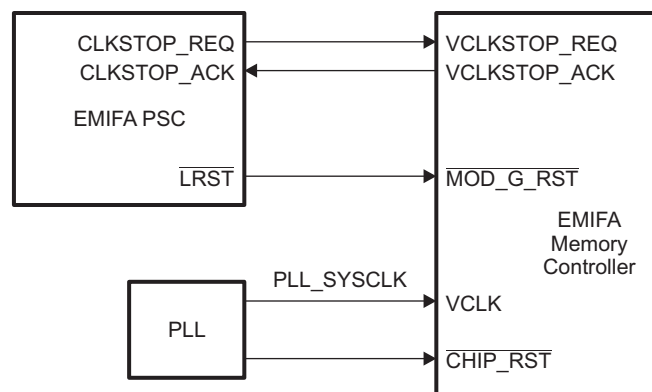
### 16.2.14 Power Management

Power dissipation from the EMIFA memory controller may be managed by following methods:

- Self-refresh mode
- Power-down mode
- Gating input clocks to the module off

Gating input clocks off to the EMIFA memory controller achieves higher power savings when compared to the power savings of self-refresh or power down mode. The input clocks are turned off outside of the EMIFA memory controller through the use of the Power and Sleep Controller (PSC) and the PLL controller. [Figure 16-17](#) shows the connections between the EMIFA memory controller, PSC, and PLL. Before gating clocks off, the EMIFA memory controller must place the SDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the clock provided by the PLL must not be turned off because this may result in data corruption. See the following subsections for the proper procedures to follow when stopping the EMIFA memory controller clocks.

**Figure 16-17. EMIFA PSC Block Diagram**



#### 16.2.14.1 Power Management Using Self-Refresh Mode

The EMIFA can be placed into a self-refresh state in order to place the attached SDRAM devices into self-refresh mode, which consumes less power for most SDRAM devices. In this state, the attached SDRAM device uses an internal clock to perform its own auto refresh cycles. This maintains the validity of the data in the SDRAM without the need for any external commands. Refer to [Section 16.2.4.7](#) for more details on placing the EMIFA into the self-refresh state.

#### 16.2.14.2 Power Management Using Power Down Mode

In case of power down, to lower the power consumption, EMIFA drives EMA\_SDCKE low. EMA\_SDCKE goes high when there is a need to send refresh (REFR) commands, after which EMA\_SDCKE is again driven low. EMA\_SDCKE remains low until any request arrives. Refer to [Section 16.2.4.8](#) for more details on placing EMIFA in power down mode.

### 16.2.14.3 Power Management Using Clock Stop

The LPSC of the memory controller can be programmed to be in one of the following states:

- Enable
- Auto Sleep
- Auto Wake
- Sync Reset

After the EMIFA clock is enabled, by default it is in the enable state. EMIFA can be put to auto sleep state, when the clock is to be gated off. Auto Wake brings back EMIFA to the enable state from the auto sleep state.

#### 16.2.14.3.1 Auto Sleep and Auto Wake

To achieve maximum power savings EMIFA core clock should be gated off. EMIFA memory controller can make use of auto sleep and auto wake to achieve clock gating. Following describes the procedure to be followed to put EMIFA memory controller in auto sleep state:

- EMIFA should be put to self-refresh mode before stopping the clock. Refer to [Section 16.2.4.7](#) for details on self-refresh mode. The EMIFA memory controller will complete any outstanding accesses and backlogged refresh cycles and then place the EMIFA memory in self-refresh mode.
- Then, program the LPSC of EMIFA for auto sleep, to gate off the clocks.

Register and memory access requests are honored while EMIFA is in auto sleep state. When EMIFA sees a request while it is in auto sleep state, it automatically returns to enable state, processes the request, and returns back to auto sleep state until further requests come.

On frequent requests, EMIFA switches between auto sleep and enable states. To bring EMIFA back to the enable state, auto wake can be used. Following procedure is followed for performing auto wake.

- Program the LPSC of EMIFA for auto wake.
- Bring EMIFA out of self-refresh. Refer to [Section 16.2.4.7](#) for details on self-refresh mode.

After auto wake, EMIFA is in enable state and clocks run continuously.

#### 16.2.14.3.2 Sync Reset and Enable

Sync reset of EMIFA through the LPSC does not reset the EMIFA registers or memory. Thus EMIFA LPSC sync reset behavior is similar to EMIFA LPSC auto sleep, except that register or memory requests are not honored by EMIFA. Following is the procedure to put EMIFA in sync reset state:

- EMIFA should be put to self-refresh mode before stopping the clock. Refer to [Section 16.2.4.7](#) for details on self-refresh mode. The EMIFA memory controller will complete any outstanding accesses and backlogged refresh cycles and then place the EMIFA memory in self-refresh mode.
- Then, program the LPSC of EMIFA to Sync-Reset state.

On sync reset, requests to EMIFA are not honored. To bring EMIFA back to the enable state, use the following enable procedure:

- Program the LPSC of EMIFA to enter enable state.
- Bring EMIFA out of self-refresh. Refer to [Section 16.2.4.7](#) for details on self-refresh mode.

Now EMIFA memory controller is in the enable state and continues with normal operation.

### 16.2.15 Emulation Considerations

EMIFA memory controller will remain fully functional during emulation halts, to allow emulation access to external memory.

## 16.3 Example Configuration

This section presents an example of interfacing the EMIFA to both an SDR SDRAM device and an asynchronous flash device.

### 16.3.1 Hardware Interface

Figure 16-18 shows the hardware interface between the EMIFA, a Samsung K4S641632H-TC(L)70 64Mb SDRAM device, and two SHARP LH28F800BJE-PTTL90 8Mb Flash memory. The connection between the EMIFA and the SDRAM is straightforward, but the connection between the EMIFA and the flash deserves a detailed look.

The address inputs for the flash are provided by three sources. The A[12:0] address inputs are provided by a combination of the EMA\_A and EMA\_BA pins according to Section 16.2.5.1. The upper address inputs A[18:13] are provided by GPIO pins. The six GPIO pins are connected to the upper address bits of the flash memory and attached to pulldown resistors so that their value is 0 after reset and before configuring the pins as GPIO. This is necessary if the ROM bootloader is copying the secondary bootloader from the flash. More details on using GPIO pins as upper address pins can be found in Section 16.2.5.2. RD/ $\overline{\text{BY}}$  signal from one flash is connected to EMA\_WAIT pin of EMIFA. A GPIO pin can be made use of to receive the RD/ $\overline{\text{BY}}$  signal coming from the second flash, as shown in Figure 16-18

Finally, this example configuration connects the  $\overline{\text{EMA\_WE}}$  pin to the  $\overline{\text{WE}}$  input of the flash and operates the EMIFA in Normal Mode.

### 16.3.2 Software Configuration

The following sections describe how to interface the EMIFA to SDRAM, Asynchronous SRAM (ASRAM), or a NAND Flash device.

#### 16.3.2.1 Configuring the SDRAM Interface

This section describes how to configure the EMIFA to interface with the Samsung K4S641632H-TC(L)70 SDRAM with a clock frequency of  $f_{\text{EMA\_CLK}} = 100$  MHz. Procedure A described in Section 16.2.4.5 is followed which assumes that the SDRAM power-up timing constraint were met during the SDRAM Auto-Initialization sequence after Reset.

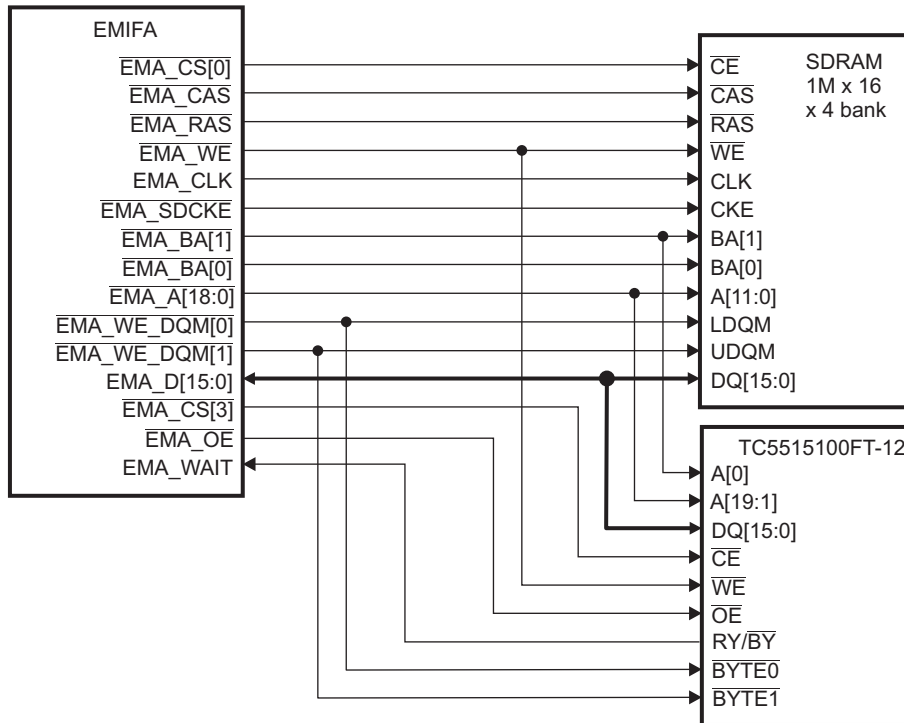
##### 16.3.2.1.1 PLL Programming for the EMIFA to K4S641632H-TC(L)70 Interface

The device PLL Controller should first be programmed to select the desired EMA\_CLK frequency. Before doing this, the SDRAM should be placed in Self-Refresh Mode by setting the SR bit in the SDRAM configuration register (SDCR). The SR bit should be set using a byte-write to the upper byte of the SDCR to avoid triggering the SDRAM Initialization Sequence. The EMA\_CLK frequency can now be adjusted to the desired value by programming the appropriate SYSCLK domain of the PLL Controller. Once the PLL has been reprogrammed, remove the SDRAM from Self-Refresh by clearing the SR bit in SDCR, again with a byte-write.

**Table 16-26. SR Field Value For the EMIFA to K4S641632H-TC(L)70 Interface**

Field	Value	Purpose
SR	1 then 0	To place the EMIFA into the self refresh state

Figure 16-18. Example Configuration Interface



### 16.3.2.1.2 SDRAM Timing Register (SDTIMR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface

The fields of the SDRAM timing register (SDTIMR) should be programmed first as described in [Table 16-27](#) to satisfy the required timing parameters for the K4S641632H-TC(L)70. Based on these calculations, a value of 6111 4610h should be written to SDTIMR. [Figure 16-19](#) shows a graphical description of how SDTIMR should be programmed.

**Table 16-27. SDTIMR Field Calculations for the EMIFA to K4S641632H-TC(L)70 Interface**

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_RFC	$T\_RFC \geq (t_{RFC} \times f_{EMA\_CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}^{(1)}$	6
T_RP	$T\_RP \geq (t_{RP} \times f_{EMA\_CLK}) - 1$	$t_{RP} = 20 \text{ ns (min)}$	1
T_RCD	$T\_RCD \geq (t_{RCD} \times f_{EMA\_CLK}) - 1$	$t_{RCD} = 20 \text{ ns (min)}$	1
T_WR	$T\_WR \geq (t_{WR} \times f_{EMA\_CLK}) - 1$	$t_{RDL} = 2 \text{ CLK} = 20 \text{ ns (min)}^{(2)}$	1
T_RAS	$T\_RAS \geq (t_{RAS} \times f_{EMA\_CLK}) - 1$	$t_{RAS} = 49 \text{ ns (min)}$	4
T_RC	$T\_RC \geq (t_{RC} \times f_{EMA\_CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}$	6
T_RRD	$T\_RRD \geq (t_{RRD} \times f_{EMA\_CLK}) - 1$	$t_{RRD} = 14 \text{ ns (min)}$	1

<sup>(1)</sup> The Samsung datasheet does not specify a  $t_{RFC}$  value. Instead, Samsung specifies  $t_{RC}$  as the minimum auto refresh period.

<sup>(2)</sup> The Samsung datasheet does not specify a  $t_{WR}$  value. Instead, Samsung specifies  $t_{RDL}$  as last data in to row precharge minimum delay.

**Figure 16-19. SDRAM Timing Register (SDTIMR)**

31	27	26	24	23	22	20	19	18	16
0 0110		001		0	001		0	001	
T_RFC		T_RP		Rsvd	T_RCD		Rsvd	T_WR	
15	12	11	8	7	6	4	3	0	
0100		0110		0	001		0000		
T_RAS		T_RC		Rsvd	T_RRD		Reserved		

**16.3.2.1.3 SDRAM Self Refresh Exit Timing Register (SDSRETR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface**

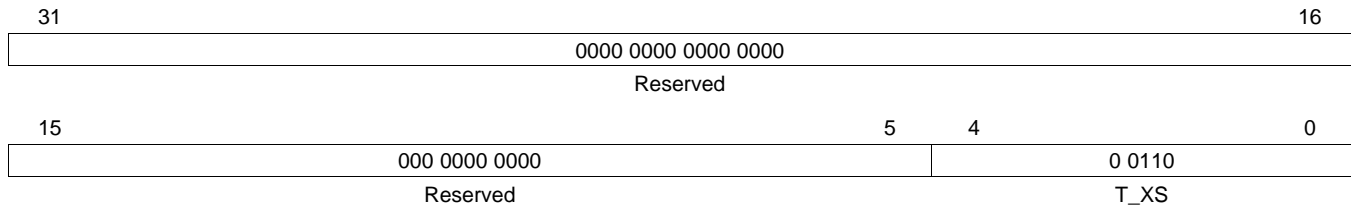
The SDRAM self refresh exit timing register (SDSRETR) should be programmed second to satisfy the  $t_{XSR}$  timing requirement from the K4S641632H-TC(L)70 datasheet. Table 16-28 shows the calculation of the proper value to program into the T\_XS field of this register. Based on this calculation, a value of 6h should be written to SDSRETR. Figure 16-20 shows how SDSRETR should be programmed.

**Table 16-28. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface**

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_XS	$T\_XS \geq (t_{XSR} \times f_{EMA\_CLK}) - 1$	$t_{RC} = 68 \text{ ns (min)}^{(1)}$	6

<sup>(1)</sup> The Samsung datasheet does not specify a  $t_{XSR}$  value. Instead, Samsung specifies  $t_{RC}$  as the minimum required time after CKE going high to complete self refresh exit.

**Figure 16-20. SDRAM Self Refresh Exit Timing Register (SDSRETR)**



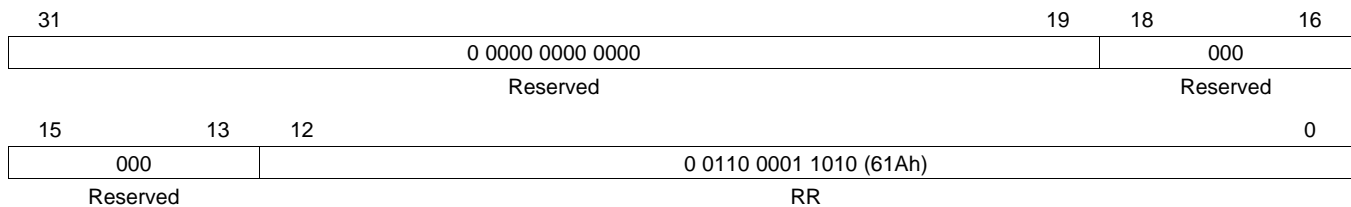
**16.3.2.1.4 SDRAM Refresh Control Register (SDRCR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface**

The SDRAM refresh control register (SDRCR) should next be programmed to satisfy the required refresh rate of the K4S641632H-TC(L)70. Table 16-29 shows the calculation of the proper value to program into the RR field of this register. Based on this calculation, a value of 61Ah should be written to SDRCR. Figure 16-21 shows how SDRCR should be programmed.

**Table 16-29. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface**

Field Name	Formula	Values	Value Calculated for Field
RR	$RR \leq \frac{f_{EMA\_CLK} \times t_{Refresh\ Period}}{n_{cycles}}$	From SDRAM datasheet: $t_{Refresh\ Period} = 64 \text{ ms}$ ; $n_{cycles} = 4096$ EMIFA clock rate: $f_{EMA\_CLK} = 100 \text{ MHz}$	$RR = 1562 \text{ cycles} = 61Ah \text{ cycles}$

**Figure 16-21. SDRAM Refresh Control Register (SDRCR)**



### 16.3.2.1.5 SDRAM Configuration Register (SDCR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface

Finally, the fields of the SDRAM configuration register (SDCR) should be programmed as described in [Table 16-30](#) to properly interface with the K4S641632H-TC(L)70 device. Based on these settings, a value of 4720h should be written to SDCR. [Figure 16-22](#) shows how SDCR should be programmed. The EMIFA is now ready to perform read and write accesses to the SDRAM.

**Table 16-30. SDCR Field Values For the EMIFA to K4S641632H-TC(L)70 Interface**

Field	Value	Purpose
SR	0	To avoid placing the EMIFA into the self refresh state
NM	1	To configure the EMIFA for a 16-bit data bus
CL	011b	To select a CAS latency of 3
BIT11_9LOCK	1	To allow the CL field to be written
IBANK	010b	To select 4 internal SDRAM banks
PAGESIZE	0	To select a page size of 256 words

**Figure 16-22. SDRAM Configuration Register (SDCR)**

31	30	29	28	24
0	0	0	0 0000	
SR	Reserved	Reserved	Reserved	
23	00 0000		18	17
Reserved		Reserved		Reserved
15	14	13	12	11
0	1	0	0	011
Reserved	NM	Reserved	Reserved	CL
7	6	4	3	2
0	010		0	000
Reserved	IBANK		Reserved	PAGESIZE
8	0			
BIT11_9LOCK				

### 16.3.2.2 Interfacing to Asynchronous SRAM (ASRAM)

The following example describes how to interface the EMIFA to the Toshiba TC55V16100FT-12 device.

#### 16.3.2.2.1 Meeting AC Timing Requirements for ASRAM

When configuring the EMIFA to interface to ASRAM, you must consider the AC timing requirements of the ASRAM as well as the AC timing requirements of the EMIFA. These can be found in the data sheet for each respective device. The read and write asynchronous cycles are programmed separately in the asynchronous configuration register (CE<sub>n</sub>CFG).

For a read access, [Table 16-31](#) to [Table 16-33](#) list the AC timing specifications that must be considered.

**Table 16-31. EMIFA Input Timing Requirements**

Parameter	Description
t <sub>SU</sub>	Data Setup time, data valid before EMA_OE high
t <sub>H</sub>	Data Hold time, data valid after EMA_OE high

**Table 16-32. ASRAM Output Timing Characteristics**

Parameter	Description
t <sub>ACC</sub>	Address Access time
t <sub>OH</sub>	Output data Hold time for address change
t <sub>COD</sub>	Output Disable time from chip enable

**Table 16-33. ASRAM Input Timing Requirement for a Read**

Parameter	Description
t <sub>RC</sub>	Read Cycle time

[Figure 16-23](#) shows an asynchronous read access and describes how the EMIFA and ASRAM AC timing requirements work together to define the values for R\_SETUP, R\_STROBE, and R\_HOLD.

From [Figure 16-23](#), the following equations may be derived. t<sub>cyc</sub> is the period at which the EMIFA operates. The R\_SETUP, R\_STROBE, and R\_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nanoseconds. This explains the presence of t<sub>cyc</sub> in the denominator of the following equations. A minus 1 is included in the equations because each field in CE<sub>n</sub>CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, R\_SETUP is equal to R\_SETUP width in EMIFA clock cycles minus 1 cycle.

$$R\_SETUP + R\_STROBE \geq \frac{t_{ACC}(m) + t_{SU}}{t_{cyc}} - 2$$

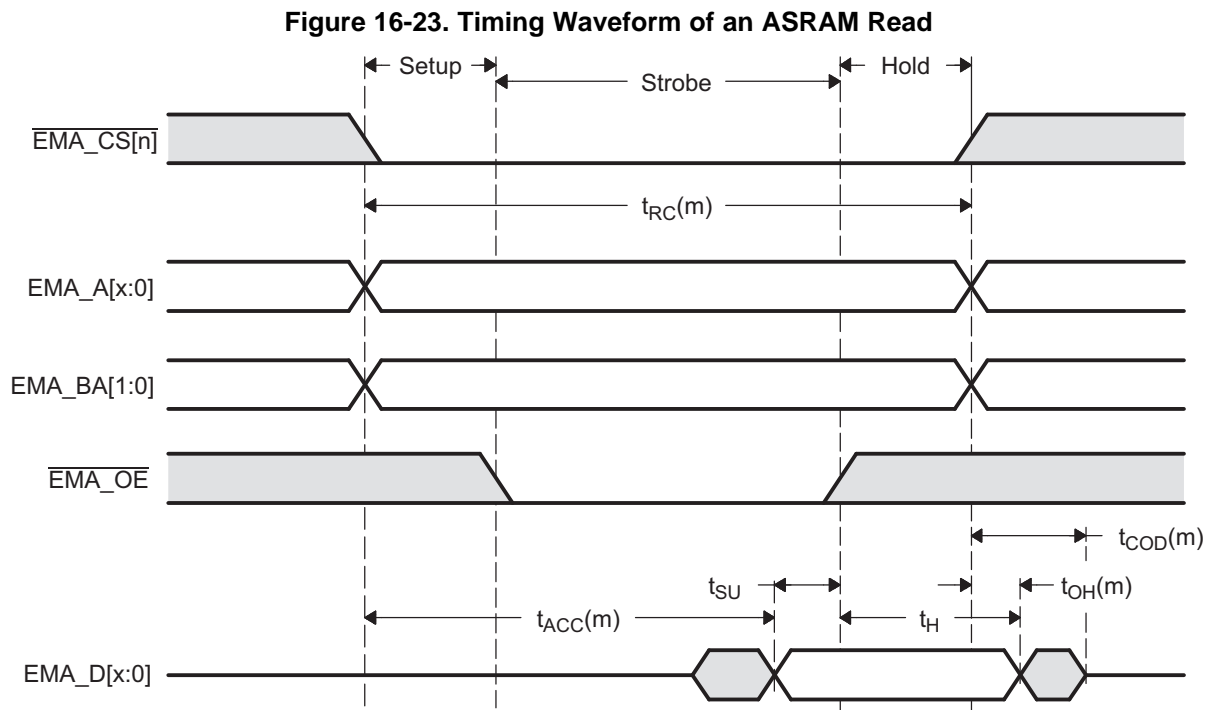
$$R\_SETUP + R\_STROBE + R\_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

$$R\_HOLD \geq \frac{(t_H - t_{OH}(m))}{t_{cyc}} - 1$$



The EMIFA offers an additional parameter, TA, that defines the turnaround time between read and write cycles. This parameter protects against the situation when the output turn-off time of the memory is longer than the time it takes to start the next write cycle. If this is the case, the EMIFA will drive data at the same time as the memory, causing contention on the bus. By examining [Figure 16-23](#), the equation for TA can be derived as:

$$TA \geq \frac{t_{\text{COD}}(m)}{t_{\text{cyc}}} - 1$$



For a write access, [Table 16-34](#) lists the AC timing specifications that must be satisfied.

**Table 16-34. ASRAM Input Timing Requirements for a Write**

Parameter	Description
t <sub>WP</sub>	Write Pulse width
t <sub>AW</sub>	Address valid to end of Write
t <sub>DS</sub>	Data Setup time
t <sub>WR</sub>	Write Recovery time
t <sub>DH</sub>	Data Hold time
t <sub>WC</sub>	Write Cycle time

Figure 16-24 shows an asynchronous write access and describes how the EMIFA and ASRAM AC timing requirements work together to define values for W\_SETUP, W\_STROBE, and W\_HOLD.

From Figure 16-24, the following equations may be derived.  $t_{cyc}$  is the period at which the EMIFA operates. The W\_SETUP, W\_STROBE, and W\_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nano seconds. This is explains the presence of  $t_{cyc}$  in the denominator of the following equations. A minus 1 is included in the equations because each field in CENCFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, W\_SETUP is equal to W\_SETUP width in EMIFA clock cycles minus 1 cycle.≥

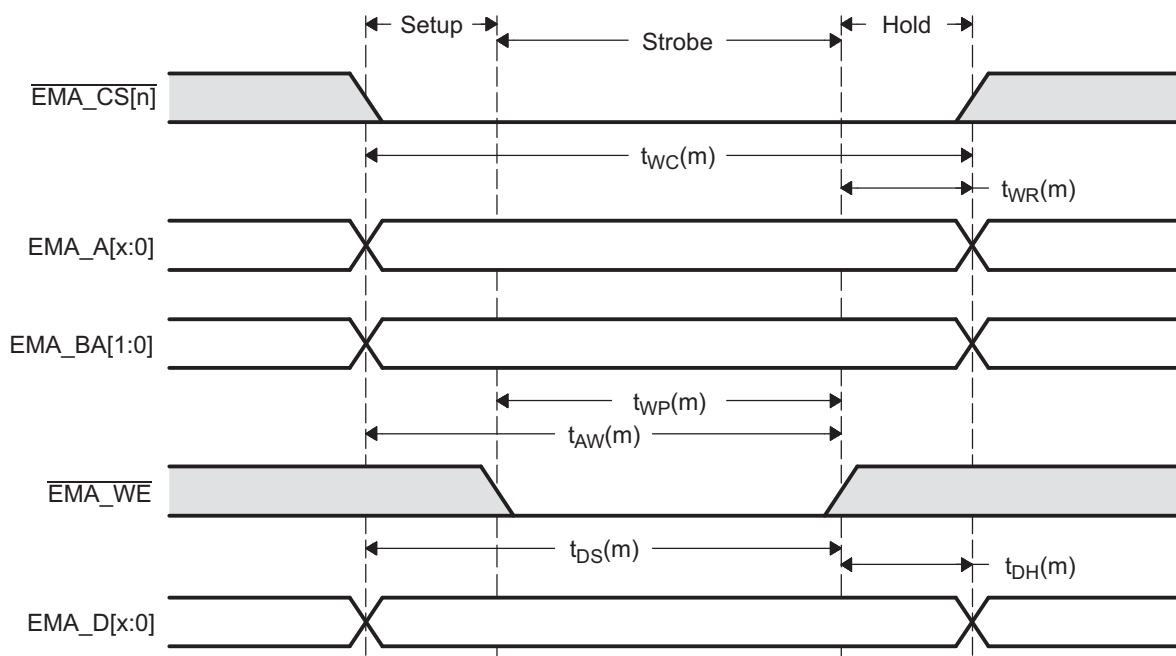
$$W\_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$W\_SETUP + W\_STROBE \geq \max\left(\frac{t_{AW}(m)}{t_{cyc}}, \frac{t_{DS}(m)}{t_{cyc}}\right) - 2$$

$$W\_HOLD \geq \max\left(\frac{t_{WR}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Figure 16-24. Timing Waveform of an ASRAM Write



### 16.3.2.2.2 Taking Into Account PCB Delays

The equations described in [Section 16.3.2.2.1](#) are for the ideal case, when board design does not contribute delays. Board characteristics, such as impedance, loading, length, number of nodes, etc., affect how the device driver behaves. Signals driven by the EMIFA will be delayed when they reach the ASRAM and conversely. [Table 16-35](#) lists the delays shown in [Figure 16-25](#) and [Figure 16-26](#) due to PCB affects. The PCB delays are board specific and must be estimated or determined through the use of IBIS modeling. The signals denoted (ASRAM) are the signals seen at the ASRAM. For example,  $\overline{EMA\_CS}$  represents the signal at the EMIFA and  $\overline{EMA\_CS}$  (ASRAM) represents the delayed signal seen at the ASRAM.

**Table 16-35. ASRAM Timing Requirements With PCB Delays**

Parameter	Description
<b>Read Access</b>	
$t_{EM\_CS}$	Delay on $\overline{EMA\_CS}$ from EMIFA to ASRAM. $\overline{EMA\_CS}$ is driven by EMIF.
$t_{EM\_A}$	Delay on EMA_A from EMIFA to ASRAM. EMA_A is driven by EMIF.
$t_{EM\_OE}$	Delay on $\overline{EMA\_OE}$ from EMIFA to ASRAM. $\overline{EMA\_OE}$ is driven by EMIF.
$t_{EM\_D}$	Delay on EMA_D from ASRAM to EMIFA. EMA_D is driven by ASRAM.
<b>Write Access</b>	
$t_{EM\_CS}$	Delay on $\overline{EMA\_CS}$ from EMIFA to ASRAM. $\overline{EMA\_CS}$ is driven by EMIF.
$t_{EM\_A}$	Delay on EMA_A from EMIFA to ASRAM. EMA_A is driven by EMIF.
$t_{EM\_WE}$	Delay on $\overline{EMA\_WE}$ from EMIFA to ASRAM. $\overline{EMA\_WE}$ is driven by EMIF.
$t_{EM\_D}$	Delay on EMA_D from EMIFA to ASRAM. EMA_D is driven by EMIF.

From [Figure 16-25](#), the following equations may be derived.  $t_{cyc}$  is the period at which the EMIFA operates. The R\_SETUP, R\_STROBE, and R\_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nano seconds. This explains the presence of  $t_{cyc}$  in the denominator of the following equations. A minus 1 is included in the equations because each field in CENCFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, R\_SETUP is equal to R\_SETUP width in EMIFA clock cycles minus 1 cycle.

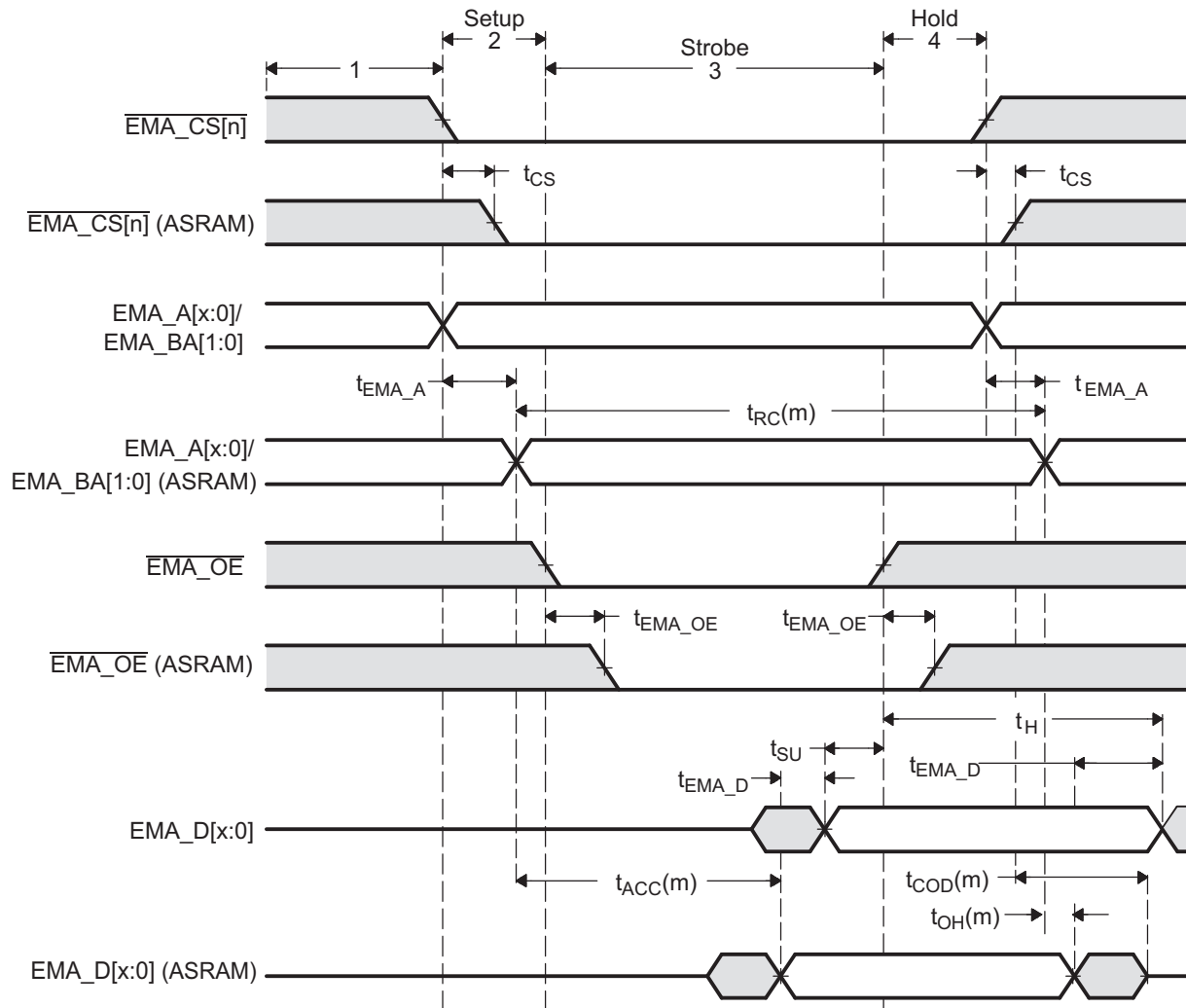
$$R\_SETUP + R\_STROBE \geq \frac{(t_{EM\_A} + t_{ACC}(m) + t_{SU} + t_{EM\_D})}{t_{cyc}} - 2$$

$$R\_SETUP + R\_STROBE + R\_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

$$R\_HOLD \geq \frac{(t_H - t_{EM\_D} - t_{OH}(m) - t_{EM\_A})}{t_{cyc}} - 1$$

$$TA \geq \frac{(t_{EM\_CS} + t_{COD}(m) + t_{EM\_D})}{t_{cyc}} - 1$$

Figure 16-25. Timing Waveform of an ASRAM Read with PCB Delays



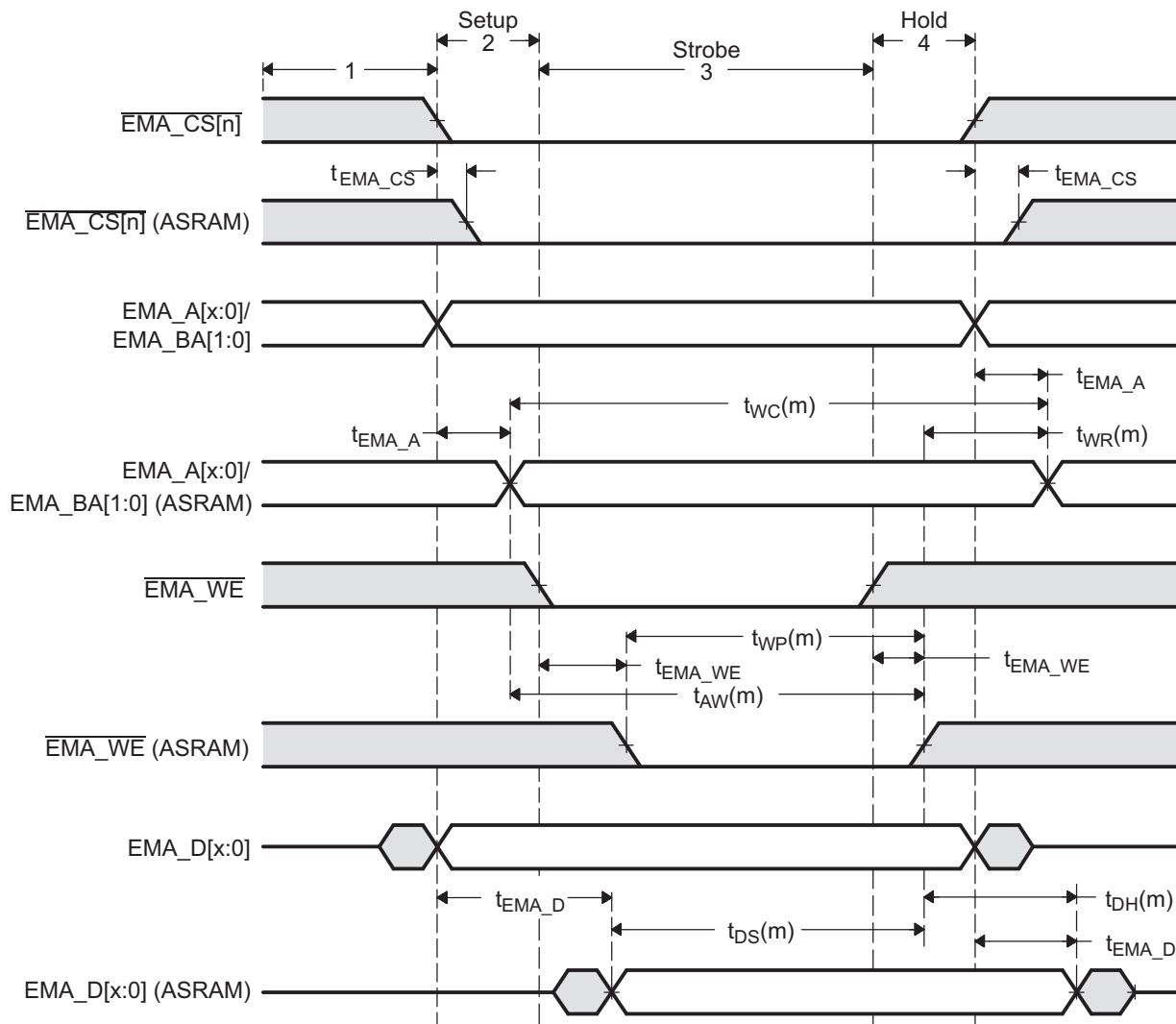
From Figure 16-26, the following equations may be derived.  $t_{cyc}$  is the period at which the EMIFA operates. The  $W\_SETUP$ ,  $W\_STROBE$ , and  $W\_HOLD$  fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nano seconds. This explains the presence of  $t_{cyc}$  in the denominator of the following equations. A minus 1 is included in the equations because each field in  $CE_nCFG$  is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example,  $W\_SETUP$  is equal to  $W\_SETUP$  width in EMIFA clock cycles minus 1 cycle.

$$W\_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$W\_SETUP + W\_STROBE \geq \max\left(\frac{(t_{EM\_A} + t_{AW}(m) - t_{EM\_WE})}{t_{cyc}}, \frac{(t_{EM\_D} + t_{DS}(m) - t_{EM\_WE})}{t_{cyc}}\right) - 2$$

$$W\_HOLD \geq \max\left(\frac{(t_{EM\_WE} + t_{WR}(m) - t_{EM\_A})}{t_{cyc}}, \frac{(t_{EM\_WE} + t_{DH}(m) - t_{EM\_D})}{t_{cyc}}\right) - 1$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

**Figure 16-26. Timing Waveform of an ASRAM Write with PCB Delays**


### 16.3.2.2.3 Example Using TC5516100FT-12

This section takes you through the configuration steps required to implement Toshiba's TC55V1664FT-12 ASRAM with the EMIFA. The following assumptions are made:

- ASRAM is connected to chip select space 3 ( $\overline{\text{EMA\_CS}}[3]$ )
- EMIFA clock speed is 100 MHz ( $t_{\text{cyc}} = 10 \text{ nS}$ )

Table 16-36 lists the data sheet specifications for the EMIFA and Table 16-37 lists the data sheet specifications for the ASRAM.

**Table 16-36. EMIFA Timing Requirements for TC5516100FT-12 Example**

Parameter	Description	Min	Max	Units
$t_{\text{SU}}$	Data Setup time, data valid before $\overline{\text{EMA\_OE}}$ high	3 to 7 <sup>(1)</sup>		nS
$t_{\text{H}}$	Data Hold time, data valid after $\overline{\text{EMA\_OE}}$ high	0		nS

<sup>(1)</sup> Depending on operating conditions. See your device-specific data manual for the value.

**Table 16-37. ASRAM Timing Requirements for TC5516100FT-12 Example**

Parameter	Description	Min	Max	Units
$t_{\text{ACC}}$	Address Access time		12	nS
$t_{\text{OH}}$	Output data Hold time for address change	3		nS
$t_{\text{RC}}$	Read cycle time	12		nS
$t_{\text{WP}}$	Write Pulse width	8		nS
$t_{\text{AW}}$	Address valid to end of Write	9		nS
$t_{\text{DS}}$	Data Setup time	7		nS
$t_{\text{WR}}$	Write Recovery time	0		nS
$t_{\text{DH}}$	Data Hold time	0		nS
$t_{\text{WC}}$	Write Cycle time	12		nS
$t_{\text{COD}}$	Output Disable time from chip enable		7	

Table 16-38 lists the values of the PCB board delays. The delays were estimated using the rule that there is 180 pS of delay for every 1 inch of trace.

**Table 16-38. Measured PCB Delays for TC5516100FT-12 Example**

Parameter	Description	Delay (ns)
<b>Read Access</b>		
$t_{\text{EM\_CS}}$	Delay on $\overline{\text{EMA\_CS}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_CS}}$ is driven by EMIF.	0.36
$t_{\text{EM\_A}}$	Delay on $\overline{\text{EMA\_A}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_A}}$ is driven by EMIF.	0.27
$t_{\text{EM\_OE}}$	Delay on $\overline{\text{EMA\_OE}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_OE}}$ is driven by EMIF.	0.36
$t_{\text{EM\_D}}$	Delay on $\overline{\text{EMA\_D}}$ from ASRAM to EMIFA. $\overline{\text{EMA\_D}}$ is driven by ASRAM.	0.45
<b>Write Access</b>		
$t_{\text{EM\_CS}}$	Delay on $\overline{\text{EMA\_CS}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_CS}}$ is driven by EMIF.	0.36
$t_{\text{EM\_A}}$	Delay on $\overline{\text{EMA\_A}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_A}}$ is driven by EMIF.	0.27
$t_{\text{EM\_WE}}$	Delay on $\overline{\text{EMA\_WE}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_WE}}$ is driven by EMIF.	0.36
$t_{\text{EM\_D}}$	Delay on $\overline{\text{EMA\_D}}$ from EMIFA to ASRAM. $\overline{\text{EMA\_D}}$ is driven by EMIF.	0.45

Inserting these values into the equations defined above allows you to determine the values for SETUP, STROBE, HOLD, and TA. For a read:

$$R\_SETUP + R\_STROBE \geq \frac{(t_{EM\_A} + t_{ACC}(m) + t_{SU} + t_{EM\_D})}{t_{cyc}} - 2 \geq \frac{(0.27 + 12 + 5 + 0.45)}{10} - 2 \geq -0.23$$

$$R\_SETUP + R\_STROBE + R\_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3 \geq \left(\frac{12}{10}\right) - 3 \geq -1.8$$

$$R\_HOLD \geq \frac{(t_H - t_{EM\_D} - t_{OH}(m) - t_{EM\_A})}{t_{cyc}} - 1 \geq \frac{(0 - 0.45 - 3 - 0.27)}{10} - 1 \geq -1.37$$

$$TA \geq \frac{(t_{EM\_CS} + T_{COD}(m) + t_{EM\_D})}{t_{cyc}} - 1 \geq \frac{(0.36 + 7 + 0.45)}{10} - 1 \geq -0.22$$

Therefore if R\_SETUP = 0, then R\_STROBE = 0, R\_HOLD = 0, and TA = 0.

For a write:

$$W\_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1 \geq \left(\frac{8}{10}\right) - 1 \geq -0.2$$

$$\begin{aligned} W\_SETUP + W\_STROBE &\geq \max\left(\frac{(t_{EM\_A} + t_{AW}(m) - t_{EM\_WE})}{t_{cyc}}, \frac{(t_{EM\_D} + t_{DS}(m) - t_{EM\_WE})}{t_{cyc}}\right) - 2 \\ &\geq \max\left(\frac{(0.36 + 0 - 0.27)}{10}, \frac{(0.36 + 0 - 0.45)}{10}\right) - 2 \geq -2.01 \end{aligned}$$

$$\begin{aligned} W\_HOLD &\geq \max\left(\frac{(t_{EM\_WE} + t_{WR}(m) - t_{EM\_A})}{t_{cyc}}, \frac{(t_{EM\_WE} + t_{DH}(m) - t_{EM\_D})}{t_{cyc}}\right) - 1 \\ &\geq \max\left(\frac{(0.27 + 9 - 0.36)}{10}, \frac{(0.45 + 7 - 0.36)}{10}\right) - 1 \geq -0.1 \end{aligned}$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3 \geq \left(\frac{12}{10}\right) - 3 \geq -1.8$$

Therefore, W\_SETUP = 0, W\_STROBE = 0, and W\_HOLD = 0.

Since the value of the W\_SETUP/R\_SETUP, W\_STROBE/R\_STROBE, W\_HOLD/R\_HOLD, and TA fields are equal to EMIFA clock cycles minus 1 cycle, the CE3CFG should be configured as in [Table 16-39](#). In this example, the EMA\_WAIT signal is not implemented; therefore, the asynchronous wait cycle configuration register (AWCC) does not need to be programmed.

**Table 16-39. Configuring CE3CFG for TC5516100FT-12 Example**

Parameter	Setting
SS	<b>Select Strobe mode.</b> <ul style="list-style-type: none"> <li>SS = 0. Places EMIFA in Normal Mode.</li> </ul>
EW	<b>Extended Wait mode enable.</b> <ul style="list-style-type: none"> <li>EW = 0. Disabled Extended wait mode.</li> </ul>
W_SETUP/R_SETUP	<b>Read/Write setup widths.</b> <ul style="list-style-type: none"> <li>W_SETUP = 0</li> <li>R_SETUP = 0</li> </ul>
W_STROBE/R_STROBE	<b>Read/Write strobe widths.</b> <ul style="list-style-type: none"> <li>W_STROBE = 0</li> <li>R_STROBE = 0</li> </ul>
W_HOLD/R_HOLD	<b>Read/Write hold widths.</b> <ul style="list-style-type: none"> <li>W_HOLD = 0</li> <li>R_HOLD = 0</li> </ul>
TA	<b>Minimum turnaround time.</b> <ul style="list-style-type: none"> <li>TA = 0</li> </ul>
ASIZE	<b>Asynchronous Device Bus Width.</b> <ul style="list-style-type: none"> <li>ASIZE = 1, select a 16-bit data bus width</li> </ul>

### 16.3.2.3 Interfacing to NAND Flash

The following example explains how to interface the EMIFA to the Hynix HY27UA081G1M NAND Flash device.

#### 16.3.2.3.1 Margin Requirements

The Flash interface is typically a low-performance interface compared to synchronous memory interfaces, high-speed asynchronous memory interfaces, and high-speed FIFO interfaces. For this reason, this example gives little attention to minimizing the amount of margin required when programming the asynchronous timing parameters. The approach used requires approximately 10 ns of margin on all parameters, which is not significant for a 100-ns read or write cycle. For additional details on minimizing the amount of margin, see the ASRAM example given in [Section 16.3.2.2](#).

**Table 16-40. Recommended Margins**

Timing Parameter	Recommended Margin
Output Setup	10 nS
Output Hold	10 nS
Input Setup	10 nS
Input Hold	10 nS



### 16.3.2.3.2 Meeting AC Timing Requirements for NAND Flash

When configuring the EMIFA to interface to NAND Flash, you must consider the AC timing requirements of the NAND Flash as well as the AC timing requirements of the EMIFA. These can be found in the data sheet for each respective device. The read and write asynchronous cycles are programmed separately in the asynchronous configuration register (CE<sub>n</sub>CFG).

A NAND Flash access cycle is composed of a command, address, and data phases. The EMIFA will not automatically generate these three phases to complete a NAND access with one transfer request. To complete a NAND access cycle, multiple single asynchronous access cycles must be completed by the EMIFA. The command and address phases of a NAND Flash access cycle are asynchronous writes performed by the EMIFA where as the data phase can be either an asynchronous write or a read depending on whether the NAND Flash is being programmed or read.

Therefore, to determine the required EMIFA configuration to interface to the NAND Flash for a read operation, [Table 16-41](#) and [Table 16-42](#) list the AC timing parameters that must be considered.

**Table 16-41. EMIFA Read Timing Requirements**

Parameter	Description
t <sub>SU</sub>	Data Setup time, data valid before $\overline{\text{EMA\_OE}}$ high
t <sub>H</sub>	Data Hold time, data valid after $\overline{\text{EMA\_OE}}$ high

**Table 16-42. NAND Flash Read Timing Requirements**

Parameter	Description
t <sub>RP</sub>	Read Pulse width
t <sub>REA</sub>	Read Enable Access time
t <sub>CEA</sub>	Chip Enable low to output valid
t <sub>CHZ</sub>	Chip Enable high to output High-Z
t <sub>RC</sub>	Read Cycle time
t <sub>RHZ</sub>	Read enable high to output High-Z
t <sub>CLR</sub>	Command Latch low to Read enable low

[Figure 16-27](#) shows an asynchronous read access and describes how the EMIFA and NAND Flash AC timing requirements work together to define the values for R\_SETUP, R\_STROBE, and R\_HOLD.

From Figure 16-27, the following equations may be derived.  $t_{cyc}$  is the period at which the EMIFA operates. The R\_SETUP, R\_STROBE, and R\_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nano seconds. This explains the presence of  $t_{cyc}$  in the denominator of the following equations. A minus 1 is included in the equations because each field in CEnCFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, R\_SETUP is equal to R\_SETUP width in EMIFA clock cycles minus 1 cycle.

$$R\_SETUP \geq \frac{t_{CLR}(m)}{t_{cyc}} - 1$$

$$R\_STROBE \geq \max\left(\frac{(t_{REA}(m) + t_{SU})}{t_{cyc}}, \frac{t_{RP}(m)}{t_{cyc}}\right) - 1$$

$$R\_SETUP + R\_STROBE \geq \frac{(t_{CEA}(m) + t_{SU})}{t_{cyc}} - 2$$

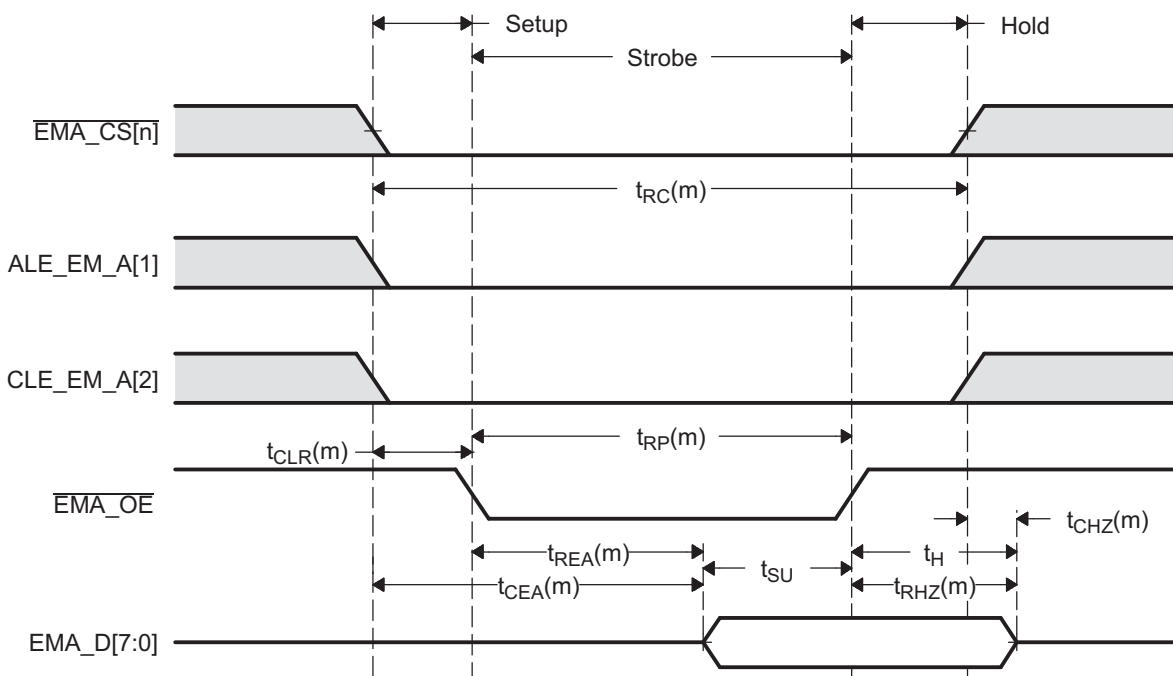
$$R\_HOLD \geq \frac{(t_H - t_{CHZ}(m))}{t_{cyc}} - 1$$

$$R\_SETUP + R\_STROBE + R\_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

The EMIFA offers an additional parameter, TA, that defines the turnaround time between read and write cycles. This parameter protects against the situation when the output turn-off time of the memory is longer than the time it takes to start the next write cycle. If this is the case, the EMIFA will drive data at the same time as the memory, causing contention on the bus. By examining Figure 16-27, the equation for TA can be derived as:

$$TA \geq \max\left(\frac{t_{CHZ}(m)}{t_{cyc}}, \frac{t_{RHZ}(m) - (R\_HOLD + 1)t_{cyc}}{t_{cyc}}\right) - 1$$

Figure 16-27. Timing Waveform of a NAND Flash Read



To determine the required EMIFA configuration to interface to the NAND Flash for a write operation, [Table 16-43](#) lists the NAND AC timing parameters for a command latch, address latch, and data input latch that must be considered.

**Table 16-43. NAND Flash Write Timing Requirements**

Parameter	Description
$t_{WP}$	Write Pulse width
$t_{CLS}$	CLE Setup time
$t_{ALS}$	ALE Setup time
$t_{CS}$	$\overline{CS}$ Setup time
$t_{DS}$	Data Setup time
$t_{CLH}$	CLE Hold time
$t_{ALH}$	ALE Hold time
$t_{CH}$	$\overline{CS}$ Hold time
$t_{DH}$	Data Hold time
$t_{WC}$	Write Cycle time

[Figure 16-28](#) to [Figure 16-30](#) show the command latch, address latch, and data input latch of the NAND access.

From [Figure 16-28](#) to [Figure 16-30](#), the following equations may be derived.  $t_{cyc}$  is the period at which the EMIFA operates. The  $W\_SETUP$ ,  $W\_STROBE$ , and  $W\_HOLD$  fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nano seconds. This explains the presence of  $t_{cyc}$  in the denominator of the following equations. A minus 1 is included in the equations because each field in  $CE_nCFG$  is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example,  $W\_SETUP$  is equal to  $W\_SETUP$  width in EMIFA clock cycles minus 1 cycle.

$$W\_SETUP \geq \max\left(\frac{t_{CLS}(m)}{t_{cyc}}, \frac{t_{ALS}(m)}{t_{cyc}}, \frac{t_{CS}(m)}{t_{cyc}}\right) - 1$$

$$W\_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$W\_SETUP + W\_STROBE \geq \frac{t_{DS}(m)}{t_{cyc}} - 2$$

$$W\_HOLD \geq \max\left(\frac{t_{CLH}(m)}{t_{cyc}}, \frac{t_{ALH}(m)}{t_{cyc}}, \frac{t_{CH}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Figure 16-28. Timing Waveform of a NAND Flash Command Write

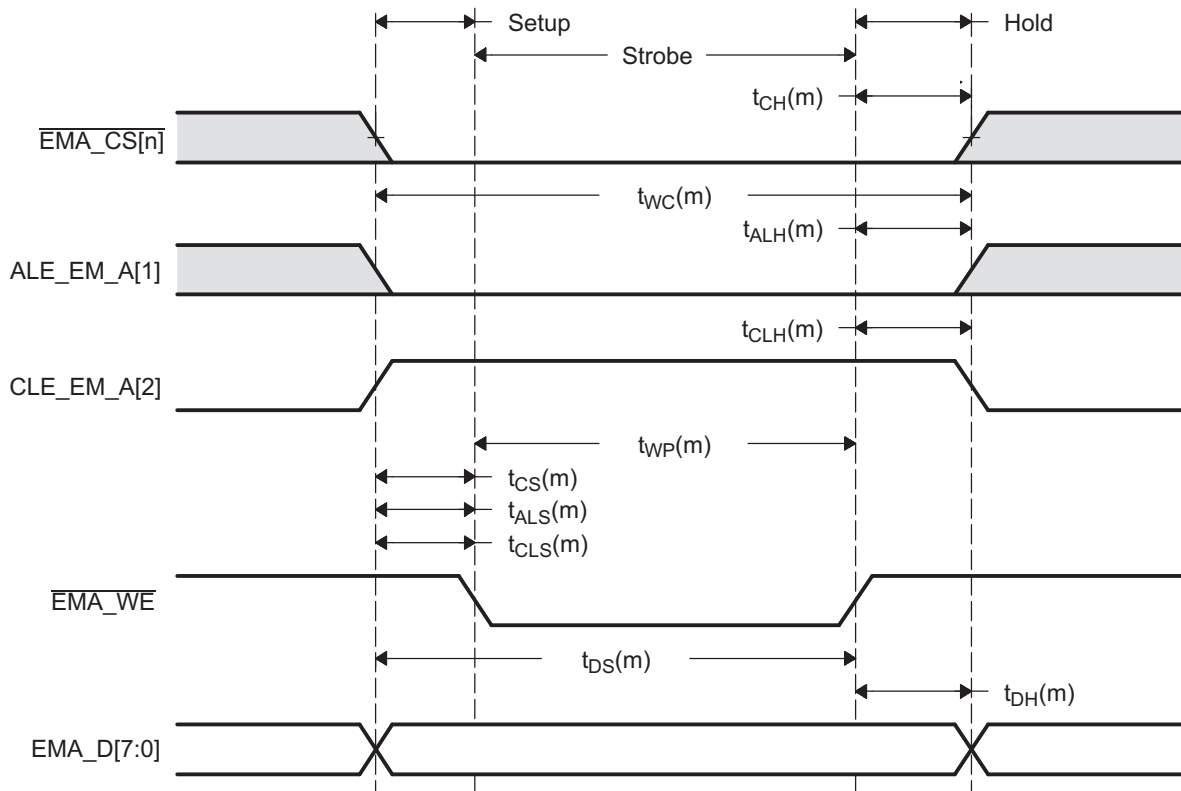


Figure 16-29. Timing Waveform of a NAND Flash Address Write

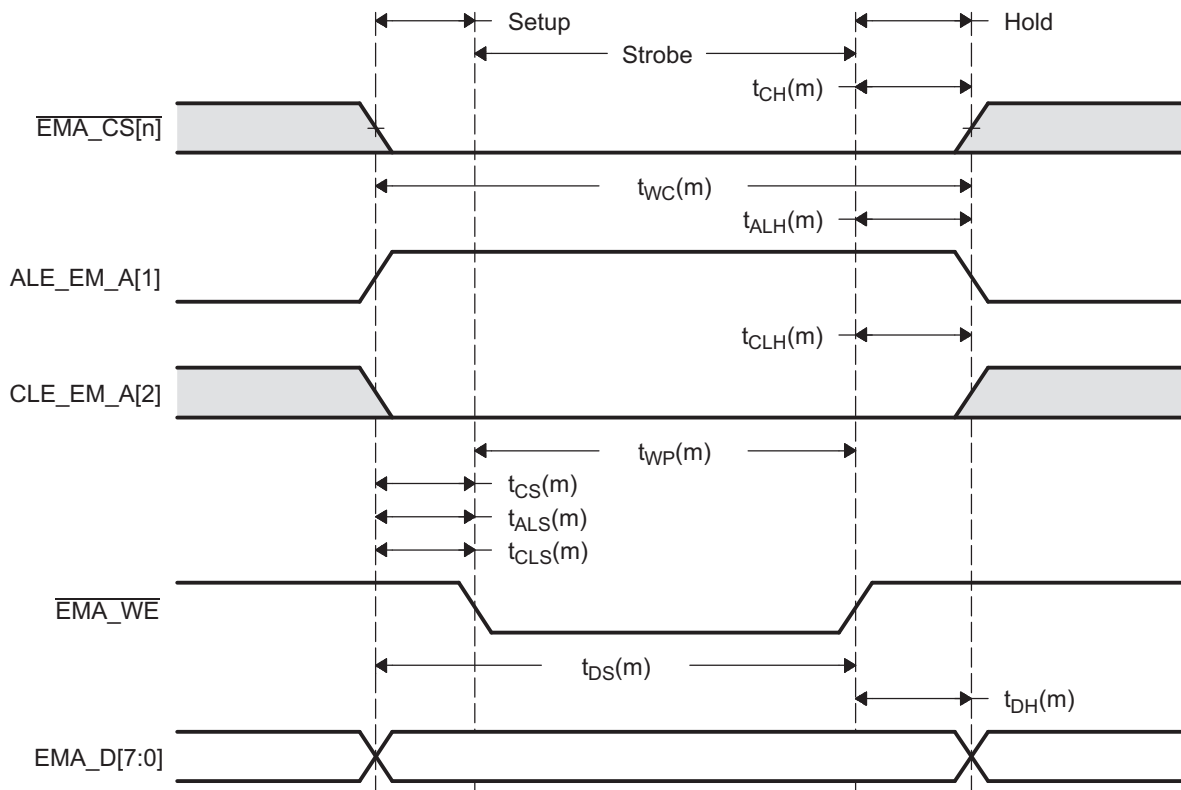
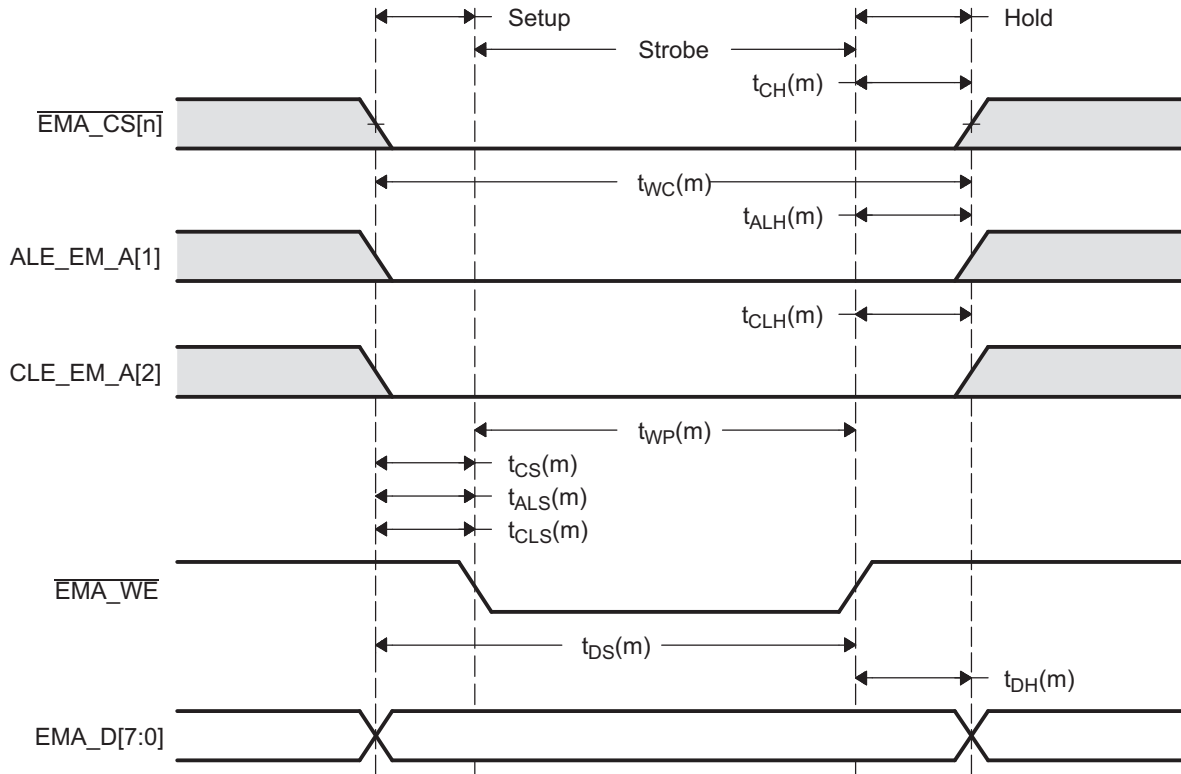


Figure 16-30. Timing Waveform of a NAND Flash Data Write



### 16.3.2.3.3 Example Using Hynix HY27UA081G1M

This section takes you through the configuration steps required to implement Hynix's HY27UA081G1M NAND Flash with the EMIFA. The following assumptions are made:

- NAND Flash is connected to chip select space 2 ( $\overline{\text{EMA\_CS}}[2]$ )
- EMIFA clock speed is 100 MHz ( $t_{\text{cyc}} = 10 \text{ nS}$ )

[Table 16-44](#) lists the data sheet specifications for the EMIFA and [Table 16-45](#) lists the data sheet specifications for the NAND Flash.

**Table 16-44. EMIFA Timing Requirements for HY27UA081G1M Example**

Parameter	Description	Min	Max	Units
$t_{\text{SU}}$	Data Setup time, data valid before $\overline{\text{EMA\_OE}}$ high	3 to 7 <sup>(1)</sup>		nS
$t_{\text{H}}$	Data Hold time, data valid after $\overline{\text{EMA\_OE}}$ high	0		nS

<sup>(1)</sup> Depending on operating conditions. See your device-specific data manual for the value.

**Table 16-45. NAND Flash Timing Requirements for HY27UA081G1M Example**

Parameter	Description	Min	Max	Units
$t_{\text{RP}}$	Read Pulse width	60		nS
$t_{\text{REA}}$	Read Enable Access time		60	nS
$t_{\text{CEA}}$	Chip Enable low to output valid		75	nS
$t_{\text{CHZ}}$	Chip Enable high to output High-Z		20	nS
$t_{\text{RC}}$	Read Cycle time	80		nS
$t_{\text{RHZ}}$	Read Enable high to output High-Z		30	nS
$t_{\text{CLR}}$	Command Latch low to Read enable low	10		nS
$t_{\text{WP}}$	Write Pulse width	60		nS
$t_{\text{CLS}}$	CLE Setup time	0		nS
$t_{\text{ALS}}$	ALE Setup time	0		nS
$t_{\text{CS}}$	$\overline{\text{CS}}$ Setup time	0		nS
$t_{\text{DS}}$	Data Setup time	20		nS
$t_{\text{CLH}}$	CLE Hold time	10		nS
$t_{\text{ALH}}$	ALE Hold time	10		nS
$t_{\text{CH}}$	$\overline{\text{CS}}$ Hold time	10		nS
$t_{\text{DH}}$	Data Hold time	10		nS
$t_{\text{WC}}$	Write Cycle time	80		nS

Inserting these values into the equations defined above allows you to determine the values for SETUP, STROBE, HOLD, and TA. For a read:

$$R\_SETUP \geq \frac{t_{CLR}(m)}{t_{cyc}} - 1 \geq \left(\frac{10}{10}\right) - 1 \geq 0$$

$$R\_STROBE \geq \max\left(\frac{(t_{REA}(m) + t_{SU})}{t_{cyc}}, \frac{t_{RP}}{t_{cyc}}\right) - 1 \geq \left(\frac{65}{10}\right) - 1 \geq 5.5$$

$$R\_SETUP + R\_STROBE \geq \frac{(t_{CEA} + t_{SU})}{t_{cyc}} - 2 \geq \frac{(75 + 5)}{10} - 2 \geq 6$$

$$R\_HOLD \geq \frac{(t_H - t_{CHZ}(m))}{t_{cyc}} - 1 \geq \frac{(0 - 20)}{10} - 1 \geq -3$$

$$R\_SETUP + R\_STROBE + R\_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3 \geq \left(\frac{80}{10}\right) - 3 \geq 5$$

Therefore with a 10 nS margin added in,  $R\_SETUP \geq 1.0$ ,  $R\_STROBE \geq 6.5$ , and  $R\_HOLD \geq 0$ .

After solving for  $R\_HOLD$ , TA may be calculated:

$$TA \geq \max\left(\frac{t_{CHZ}(m)}{t_{cyc}}, \frac{t_{RHZ}(m) - (R\_HOLD + 1)t_{cyc}}{t_{cyc}}\right) - 1 \geq \left(\frac{20}{10}\right) - 1 \geq 1$$

Adding a 10 ns margin,  $TA \geq 2$ .

For a write:

$$W\_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1 \geq \left(\frac{60}{10}\right) - 1 \geq 5$$

$$W\_SETUP \geq \max\left(\frac{t_{CLS}(m)}{t_{cyc}}, \frac{t_{ALS}(m)}{t_{cyc}}, \frac{t_{CS}(m)}{t_{cyc}}\right) - 1 \geq \left(\frac{0}{10}\right) - 1 \geq -1$$

$$W\_SETUP + W\_STROBE \geq \frac{t_{DS}(m)}{t_{cyc}} - 2 \geq \frac{20}{10} - 2 \geq 0$$

$$W\_HOLD \geq \max\left(\frac{t_{CLH}(m)}{t_{cyc}}, \frac{t_{ALH}(m)}{t_{cyc}}, \frac{t_{CH}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1 \geq \left(\frac{10}{10}\right) - 1 \geq 0$$

$$W\_SETUP + W\_STROBE + W\_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3 \geq \left(\frac{80}{10}\right) - 3 \geq 5$$

Therefore with a 10 nS margin added in,  $W\_SETUP \geq 0$ ,  $W\_STROBE \geq 6$ , and  $W\_HOLD \geq 0$ .

Since the value of the W\_SETUP/R\_SETUP, W\_STROBE/R\_STROBE, W\_HOLD/R\_HOLD, and TA fields are equal to EMIFA clock cycles minus 1 cycle, the CE2CFG should be configured as in [Table 16-46](#). In this example, although the EMA\_WAIT signal is connected to the R/B signal of the NAND Flash the Extended Wait mode of the EMIFA is not used, therefore the asynchronous wait cycle configuration register (AWCC) does not need to be programmed.

**Table 16-46. Configuring CE2CFG for HY27UA081G1M Example**

Parameter	Setting
SS	<b>Select Strobe mode.</b> <ul style="list-style-type: none"> <li>SS = 0. Places EMIFA in Normal Mode.</li> </ul>
EW	<b>Extended Wait mode enable.</b> <ul style="list-style-type: none"> <li>EW = 0. Disabled Extended wait mode.</li> </ul>
W_SETUP/R_SETUP	<b>Read/Write setup widths.</b> <ul style="list-style-type: none"> <li>W_SETUP = 0</li> <li>R_SETUP = 2</li> </ul>
W_STROBE/R_STROBE	<b>Read/Write strobe widths.</b> <ul style="list-style-type: none"> <li>W_STROBE = 6</li> <li>R_STROBE = 7</li> </ul>
W_HOLD/R_HOLD	<b>Read/Write hold widths.</b> <ul style="list-style-type: none"> <li>W_HOLD = 1</li> <li>R_HOLD = 0</li> </ul>
TA	<b>Minimum turnaround time.</b> <ul style="list-style-type: none"> <li>TA = 2</li> </ul>
ASIZE	<b>Asynchronous device bus width.</b> <ul style="list-style-type: none"> <li>ASIZE = 0, select an 8-bit data bus width.</li> </ul>

Since this is a NAND Flash example, the EMIFA must be configured for NAND Flash mode. This is accomplished by configuring the NAND Flash control register (NANDFCR) as in [Table 16-47](#). In NANDFCR, chip select space 2 must be configured with NAND Flash mode enabled.

**Table 16-47. Configuring NANDFCR for HY27UA081G1M Example**

Parameter	Setting
CS5ECC	<b>NAND Flash ECC start for chip select 5.</b> <ul style="list-style-type: none"> <li>CS5ECC = 0. Not set during configuration. Only set just prior to reading or writing data.</li> </ul>
CS4ECC	<b>NAND Flash ECC start for chip select 4.</b> <ul style="list-style-type: none"> <li>CS4ECC = 0. Not set during configuration. Only set just prior to reading or writing data.</li> </ul>
CS3ECC	<b>NAND Flash ECC start for chip select 3.</b> <ul style="list-style-type: none"> <li>CS3ECC = 0. Not set during configuration. Only set just prior to reading or writing data.</li> </ul>
CS2ECC	<b>NAND Flash ECC start for chip select 2.</b> <ul style="list-style-type: none"> <li>CS2ECC = 0. Not set during configuration. Only set just prior to reading or writing data.</li> </ul>
CS5NAND	<b>NAND Flash mode for chip select 5.</b> <ul style="list-style-type: none"> <li>CS5NAND = 0. NAND Flash mode is disabled.</li> </ul>
CS4NAND	<b>NAND Flash mode for chip select 4.</b> <ul style="list-style-type: none"> <li>CS4NAND = 0. NAND Flash mode is disabled.</li> </ul>
CS3NAND	<b>NAND Flash mode for chip select 3.</b> <ul style="list-style-type: none"> <li>CS3NAND = 0. NAND Flash mode is disabled.</li> </ul>
CS2NAND	<b>NAND Flash mode for chip select 2.</b> <ul style="list-style-type: none"> <li>CS5NAND = 1. NAND Flash mode is enabled.</li> </ul>



## 16.4 Registers

The external memory interface (EMIFA) is controlled by programming its internal memory-mapped registers (MMRs). [Table 16-48](#) lists the memory-mapped registers for the EMIFA.

**NOTE:** All EMIFA MMRs, except SDCR, support only word (32-bit) accesses. Performing a byte (8-bit) or halfword (16-bit) write to these registers results in undefined behavior. The SDCR is byte writable to allow the setting of the SR, PD and PDWR bits without triggering the SDRAM initialization sequence.

The EMIFA registers must always be accessed using 32-bit accesses (unless otherwise specified in this chapter). For the base address of the memory-mapped registers of EMIFA, see your device-specific data manual.

**Table 16-48. External Memory Interface (EMIFA) Registers**

Offset	Acronym	Register Description	Section
0h	MIDR	Module ID Register	<a href="#">Section 16.4.1</a>
4h	AWCC	Asynchronous Wait Cycle Configuration Register	<a href="#">Section 16.4.2</a>
8h	SDCR	SDRAM Configuration Register	<a href="#">Section 16.4.3</a>
Ch	SDRCR	SDRAM Refresh Control Register	<a href="#">Section 16.4.4</a>
10h	CE2CFG	Asynchronous 1 Configuration Register	<a href="#">Section 16.4.5</a>
14h	CE3CFG	Asynchronous 2 Configuration Register	<a href="#">Section 16.4.5</a>
18h	CE4CFG	Asynchronous 3 Configuration Register	<a href="#">Section 16.4.5</a>
1Ch	CE5CFG	Asynchronous 4 Configuration Register	<a href="#">Section 16.4.5</a>
20h	SDTIMR	SDRAM Timing Register	<a href="#">Section 16.4.6</a>
3Ch	SDSRETR	SDRAM Self Refresh Exit Timing Register	<a href="#">Section 16.4.7</a>
40h	INTRAW	EMIFA Interrupt Raw Register	<a href="#">Section 16.4.8</a>
44h	INTMSK	EMIFA Interrupt Mask Register	<a href="#">Section 16.4.9</a>
48h	INTMSKSET	EMIFA Interrupt Mask Set Register	<a href="#">Section 16.4.10</a>
4Ch	INTMSKCLR	EMIFA Interrupt Mask Clear Register	<a href="#">Section 16.4.11</a>
60h	NANDFCR	NAND Flash Control Register	<a href="#">Section 16.4.12</a>
64h	NANDFSR	NAND Flash Status Register	<a href="#">Section 16.4.13</a>
70h	NANDF1ECC	NAND Flash 1 ECC Register (CS2 Space)	<a href="#">Section 16.4.14</a>
74h	NANDF2ECC	NAND Flash 2 ECC Register (CS3 Space)	<a href="#">Section 16.4.14</a>
78h	NANDF3ECC	NAND Flash 3 ECC Register (CS4 Space)	<a href="#">Section 16.4.14</a>
7Ch	NANDF4ECC	NAND Flash 4 ECC Register (CS5 Space)	<a href="#">Section 16.4.14</a>
BCh	NAND4BITECCLOAD	NAND Flash 4-Bit ECC Load Register	<a href="#">Section 16.4.15</a>
C0h	NAND4BITECC1	NAND Flash 4-Bit ECC Register 1	<a href="#">Section 16.4.16</a>
C4h	NAND4BITECC2	NAND Flash 4-Bit ECC Register 2	<a href="#">Section 16.4.17</a>
C8h	NAND4BITECC3	NAND Flash 4-Bit ECC Register 3	<a href="#">Section 16.4.18</a>
CCh	NAND4BITECC4	NAND Flash 4-Bit ECC Register 4	<a href="#">Section 16.4.19</a>
D0h	NANDERRADD1	NAND Flash 4-Bit ECC Error Address Register 1	<a href="#">Section 16.4.20</a>
D4h	NANDERRADD2	NAND Flash 4-Bit ECC Error Address Register 2	<a href="#">Section 16.4.21</a>
D8h	NANDERRVAL1	NAND Flash 4-Bit ECC Error Value Register 1	<a href="#">Section 16.4.22</a>
DCh	NANDERRVAL2	NAND Flash 4-Bit ECC Error Value Register 2	<a href="#">Section 16.4.23</a>

### 16.4.1 Module ID Register (MIDR)

This is a read-only register indicating the module ID of the EMIFA. The MIDR is shown in [Figure 16-31](#) and described in [Table 16-49](#).

**Figure 16-31. Module ID Register (MIDR)**



LEGEND: R = Read only; -n = value after reset

**Table 16-49. Module ID Register (MIDR) Field Descriptions**

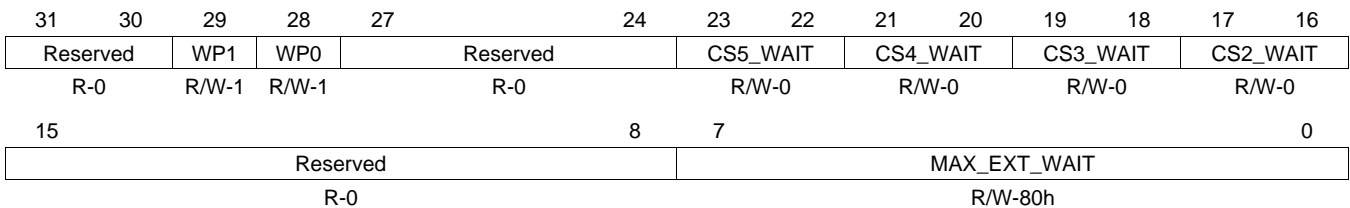
Bit	Field	Value	Description
31-0	REV	4000 0205h	Module ID of EMIFA.

### 16.4.2 Asynchronous Wait Cycle Configuration Register (AWCC)

The asynchronous wait cycle configuration register (AWCC) is used to configure the parameters for extended wait cycles. Both the polarity of the EMA\_WAIT pin(s) and the maximum allowable number of extended wait cycles can be configured. The AWCC is shown in [Figure 16-32](#) and described in [Table 16-50](#). Not all devices support both EMA\_WAIT[1] and EMA\_WAIT[0], see the device-specific data manual to determine support on each device.

**NOTE:** The EW bit in the asynchronous *n* configuration register (CE<sub>n</sub>CFG) must be set to allow for the insertion of extended wait cycles.

**Figure 16-32. Asynchronous Wait Cycle Configuration Register (AWCCR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-50. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29	WP1	0 1	EMA_WAIT[1] polarity bit. This bit defines the polarity of the EMA_WAIT[1] pin. Insert wait cycles if EMA_WAIT[1] pin is low. Insert wait cycles if EMA_WAIT[1] pin is high.
28	WP0	0 1	EMA_WAIT[0] polarity bit. This bit defines the polarity of the EMA_WAIT[0] pin. Insert wait cycles if EMA_WAIT[0] pin is low. Insert wait cycles if EMA_WAIT[0] pin is high.
27-24	Reserved	0	Reserved
23-22	CS5_WAIT	0-3h 0 1h 2h-3h	Chip Select 5 WAIT signal selection. This signal determines which EMA_WAIT[n] signal will be used for memory accesses to chip select 5 memory space. EMA_WAIT[0] pin is used to control external wait states. EMA_WAIT[1] pin is used to control external wait states. Reserved
21-20	CS4_WAIT	0-3h 0 1h 2h-3h	Chip Select 4 WAIT signal selection. This signal determines which EMA_WAIT[n] signal will be used for memory accesses to chip select 4 memory space. EMA_WAIT[0] pin is used to control external wait states. EMA_WAIT[1] pin is used to control external wait states. Reserved
19-18	CS3_WAIT	0-3h 0 1h 2h-3h	Chip Select 3 WAIT signal selection. This signal determines which EMA_WAIT[n] signal will be used for memory accesses to chip select 3 memory space. EMA_WAIT[0] pin is used to control external wait states. EMA_WAIT[1] pin is used to control external wait states. Reserved
17-16	CS2_WAIT	0-3h 0 1h 2h-3h	Chip Select 2 WAIT signal selection. This signal determines which EMA_WAIT[n] signal will be used for memory accesses to chip select 2 memory space. EMA_WAIT[0] pin is used to control external wait states.. EMA_WAIT[1] pin is used to control external wait states. Reserved
15-8	Reserved	0	Reserved
7-0	MAX_EXT_WAIT	0-FFh	Maximum extended wait cycles. The EMIFA will wait for a maximum of (MAX_EXT_WAIT + 1) × 16 clock cycles before it stops inserting asynchronous wait cycles and proceeds to the hold period of the access.

### 16.4.3 SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) is used to configure various parameters of the SDRAM controller such as the number of internal banks, the internal page size, and the CAS latency to match those of the attached SDRAM device. In addition, this register is used to put the attached SDRAM device into Self-Refresh mode. The SDCR is shown in [Figure 16-33](#) and described in [Table 16-51](#).

**NOTE:** Writing to the lower three bytes of this register will cause the EMIFA to start the SDRAM initialization sequence described in [Section 16.2.4.4](#).

**Figure 16-33. SDRAM Configuration Register (SDCR)**

31	30	29	28	24		
SR	PD	PDWR	Reserved			
R/W-0	R/W-0	R/W-0	R-0			
23	Reserved			16		
R-0						
15	14	13	12	11	9	8
Reserved	NM <sup>(A)</sup>	Reserved		CL		BIT11_9LOCK
R-0	R/W-0	R-0		R/W-3h		R/W-0
7	6	4	3	2	0	
Reserved	IBANK		Reserved	PAGESIZE		
R-0	R/W-2h		R-0	R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A. The NM bit must be set to 1 if the EMIFA on your device only has 16 data bus pins.

**Table 16-51. SDRAM Configuration Register (SDCR) Field Descriptions**

Bit	Field	Value	Description
31	SR	0 1	Self-Refresh mode bit. This bit controls entering and exiting of the Self-Refresh mode described in <a href="#">Section 16.2.4.7</a> . The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. Writing a 0 to this bit will cause connected SDRAM devices and the EMIFA to exit the Self-Refresh mode. Writing a 1 to this bit will cause connected SDRAM devices and the EMIFA to enter the Self-Refresh mode.
30	PD	0 1	Power Down bit. This bit controls entering and exiting of the power-down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIFA will go into Self Refresh. Writing a 0 to this bit will cause connected SDRAM devices and the EMIFA to exit the power-down mode. Writing a 1 to this bit will cause connected SDRAM devices and the EMIFA to enter the power-down mode.
29	PDWR		Perform refreshes during power down. Writing a 1 to this bit will cause EMIFA to exit power-down state and issue and AUTO REFRESH command every time Refresh May level is set.
28-15	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
14	NM	0 1	Narrow mode bit. This bit defines whether a 16- or 32-bit-wide SDRAM is connected to the EMIFA. This bit field must always be set to 1. Writing to this field triggers the SDRAM initialization sequence. 0 32-bit SDRAM data bus is used. 1 16-bit SDRAM data bus is used.
13-12	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.

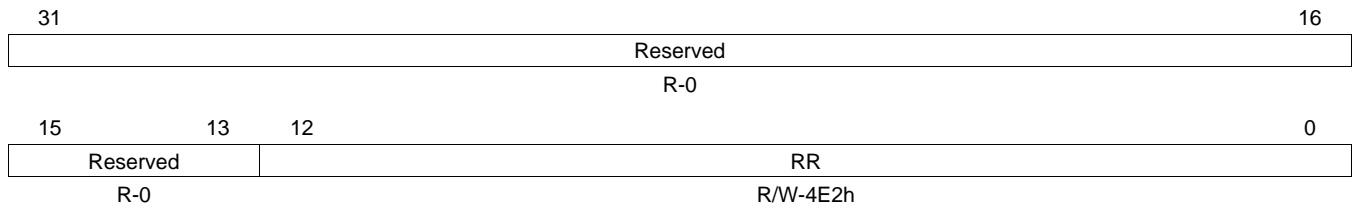
**Table 16-51. SDRAM Configuration Register (SDCR) Field Descriptions (continued)**

Bit	Field	Value	Description
11-9	CL	0-7h 0-1h 2h 3h 4h-7h	CAS Latency. This field defines the CAS latency to be used when accessing connected SDRAM devices. A 1 must be simultaneously written to the BIT11_9LOCK bit field of this register in order to write to the CL bit field. Writing to this field triggers the SDRAM initialization sequence. Reserved CAS latency = 2 EMA_CLK cycles CAS latency = 3 EMA_CLK cycles Reserved
8	BIT11_9LOCK	0 1	Bits 11 to 9 lock. CL can only be written if BIT11_9LOCK is simultaneously written with a 1. BIT11_9LOCK is always read as 0. Writing to this field triggers the SDRAM initialization sequence. CL cannot be written. CL can be written.
7	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
6-4	IBANK	0-7h 0 1 2 3h-7h	Internal SDRAM Bank size. This field defines number of banks inside the connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence. 1 bank SDRAM devices. 2 bank SDRAM devices. 4 bank SDRAM devices. Reserved.
3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2-0	PAGESIZE	0-7h 0 1h 2h 3h 4h-7h	Page Size. This field defines the internal page size of connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence. 8 column address bits (256 elements per row) 9 column address bits (512 elements per row) 10 column address bits (1024 elements per row) 11 column address bits (2048 elements per row) Reserved

### 16.4.4 SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) is used to configure the rate at which connected SDRAM devices will be automatically refreshed by the EMIFA. Refer to [Section 16.2.4.6](#) on the refresh controller for more details. The SDRCR is shown in [Figure 16-34](#) and described in [Table 16-52](#).

**Figure 16-34. SDRAM Refresh Control Register (SDRCR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-52. SDRAM Refresh Control Register (SDRCR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
12-0	RR	0-1FFFh	Refresh Rate. This field is used to define the SDRAM refresh period in terms of EMA_CLK cycles. Writing a value < 0x0020 to this field will cause it to be loaded with (2 × T_RFC) + 1 value from the SDRAM timing register (SDTIMR).

### 16.4.5 Asynchronous *n* Configuration Registers (CE2CFG-CE5CFG)

The asynchronous *n* configuration registers (CE2CFG, CE3CFG, CE4CFG, and CE5CFG) are used to configure the shaping of the address and control signals during an access to asynchronous memory connected to CS2, CS3, CS4, and CS5, respectively. It is also used to program the width of asynchronous interface and to select from various modes of operation. This register can be written prior to any transfer, and any asynchronous transfer following the write will use the new configuration. The CE $n$ CFG is shown in Figure 16-35 and described in Table 16-53.

**Figure 16-35. Asynchronous *n* Configuration Register (CE $n$ CFG)**

31	30	29	26	25	24
SS	EW <sup>(A)</sup>	W_SETUP		W_STROBE <sup>(B)</sup>	
R/W-0	R/W-0	R/W-Fh		R/W-3Fh	
23	20		19	17	16
W_STROBE <sup>(B)</sup>			W_HOLD		R_SETUP
R/W-3Fh			R/W-7h		R/W-Fh
15	13	12	7	6	4
R_SETUP		R_STROBE <sup>(B)</sup>		R_HOLD	TA
R/W-Fh		R/W-3Fh		R/W-7h	R/W-3h
				3	2
				1	0
				ASIZE	
				R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

A. The EW bit must be cleared to 0 when operating in NAND Flash mode.

B. This bit field must be cleared to 0 if the EMIFA on your device does not have an EMA\_WAIT pin.

**Table 16-53. Asynchronous *n* Configuration Register (CE $n$ CFG) Field Descriptions**

Bit	Field	Value	Description
31	SS	0 1	Select Strobe bit. This bit defines whether the asynchronous interface operates in Normal Mode or Select Strobe Mode. See Section 16.2.5 for details on the two modes of operation. 0 Normal Mode enabled. 1 Select Strobe Mode enabled.
30	EW	0 1	Extend Wait bit. This bit defines whether extended wait cycles will be enabled. See Section 16.2.5.7 on extended wait cycles for details. This bit field must be cleared to 0, if the EMIFA on your device does not have an EMA_WAIT pin. The CS $n$ _WAIT bit in the asynchronous wait cycle configuration register (AWCC) must also be configured to determine which EMA_WAIT pin is used for memory accesses. 0 Extended wait cycles disabled. 1 Extended wait cycles enabled.
29-26	W_SETUP	0-Fh	Write setup width in the format $n - 1$ , where $n$ = number of EMA_CLK cycles. See Section 16.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 ... 2h – Fh = Divide-by-3 to Divide-by-16
25-20	W_STROBE	0-3Fh	Write strobe width in the format $n - 1$ , where $n$ = number of EMA_CLK cycles. See Section 16.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 ... 2h – 3Fh = Divide-by-3 to Divide-by-64
19-17	W_HOLD	0-7h	Write hold width in the format $n - 1$ , where $n$ = number of EMA_CLK cycles. See Section 16.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 ... 2h – 7h = Divide-by-3 to Divide-by-8
16-13	R_SETUP	0-Fh	Read setup width in the format $n - 1$ , where $n$ = number of EMA_CLK cycles. See Section 16.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 ... 2h – 1Fh = Divide-by-3 to Divide-by-16

**Table 16-53. Asynchronous  $n$  Configuration Register (CE $n$ CFG) Field Descriptions (continued)**

Bit	Field	Value	Description
12-7	R_STROBE	0-3Fh	Read strobe width in the format $n - 1$ , where $n$ = number of EMA_CLK cycles. See <a href="#">Section 16.2.5.3</a> for details. 0h = Divide-by-1 1h = Divide-by-2 ... 2h – 3Fh = Divide-by-3 to Divide-by-64
6-4	R_HOLD	0-7h	Read hold width in the format $n - 1$ , where $n$ = number of EMA_CLK cycles. See <a href="#">Section 16.2.5.3</a> for details. 0h = Divide-by-1 1h = Divide-by-2 ... 2h – 7h = Divide-by-3 to Divide-by-8
3-2	TA	0-3h	Minimum Turn-Around time. This field defines the minimum number of EMA_CLK cycles between reads and writes, minus one cycle. See <a href="#">Section 16.2.5.3</a> for details.
1-0	ASIZE	0-3h	Asynchronous Data Bus Width. This field defines the width of the asynchronous device's data bus.
		0	8-bit data bus
		1h	16-bit data bus
		2h-3h	Reserved



### 16.4.6 SDRAM Timing Register (SDTIMR)

The SDRAM timing register (SDTIMR) is used to program many of the SDRAM timing parameters. Consult the SDRAM datasheet for information on the appropriate values to program into each field. The SDTIMR is shown in [Figure 16-36](#) and described in [Table 16-54](#).

**Figure 16-36. SDRAM Timing Register (SDTIMR)**

31	27	26	24	23	22	20	19	18	16
T_RFC			T_RP		Rsvd	T_RCD		Rsvd	T_WR
R/W-8h			R/W-2h		R-0	R/W-2h		R-0	R/W-1h
15	12	11	8	7	6	4	3	0	
T_RAS		T_RC			Rsvd	T_RRD		Reserved	
R/W-5h		R/W-8h			R-0	R/W-1h		R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

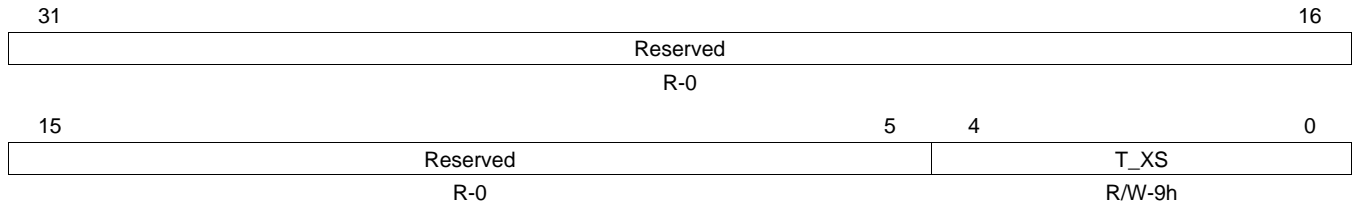
**Table 16-54. SDRAM Timing Register (SDTIMR) Field Descriptions**

Bit	Field	Value	Description
31-27	T_RFC	0-1Fh	Specifies the Trfc value of the SDRAM. This defines the minimum number of EMA_CLK cycles from Refresh (REFR) to Refresh (REFR), minus 1: $T\_RFC = (Trfc/t_{EMA\_CLK}) - 1$
26-24	T_RP	0-7h	Specifies the Trp value of the SDRAM. This defines the minimum number of EMA_CLK cycles from Precharge (PRE) to Activate (ACTV) or Refresh (REFR) command, minus 1: $T\_RP = (Trp/t_{EMA\_CLK}) - 1$
23	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
22-20	T_RCD	0-7h	Specifies the Trcd value of the SDRAM. This defines the minimum number of EMA_CLK cycles from Active (ACTV) to Read (READ) or Write (WRT), minus 1: $T\_RCD = (Trcd/t_{EMA\_CLK}) - 1$
19	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
18-16	T_WR	0-7h	Specifies the Twr value of the SDRAM. This defines the minimum number of EMA_CLK cycles from last Write (WRT) to Precharge (PRE), minus 1: $T\_WR = (Twr/t_{EMA\_CLK}) - 1$
15-12	T_RAS	0-Fh	Specifies the Tras value of the SDRAM. This defines the minimum number of EMA_CLK clock cycles from Activate (ACTV) to Precharge (PRE), minus 1: $T\_RAS = (Tras/t_{EMA\_CLK}) - 1$
11-8	T_RC	0-Fh	Specifies the Trc value of the SDRAM. This defines the minimum number of EMA_CLK clock cycles from Activate (ACTV) to Activate (ACTV), minus 1: $T\_RC = (Trc/t_{EMA\_CLK}) - 1$
7	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
6-4	T_RRD	0-7h	Specifies the Trrd value of the SDRAM. This defines the minimum number of EMA_CLK clock cycles from Activate (ACTV) to Activate (ACTV) for a different bank, minus 1: $T\_RRD = (Trrd/t_{EMA\_CLK}) - 1$
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.

### 16.4.7 SDRAM Self Refresh Exit Timing Register (SDSRETR)

The SDRAM self refresh exit timing register (SDSRETR) is used to program the amount of time between when the SDRAM exits Self-Refresh mode and when the EMIFA issues another command. The SDSRETR is shown in [Figure 16-37](#) and described in [Table 16-55](#).

**Figure 16-37. SDRAM Self Refresh Exit Timing Register (SDSRETR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-55. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions**

Bit	Field	Value	Description
31-5	Reserved	0	Reserved. The reserved bit location is always read as 0.
4-0	T_XS	0-1Fh	This field specifies the minimum number of ECLKOUT cycles from Self-Refresh exit to any command, minus one. $T\_XS = T_{xsr} / t_{EMA\_CLK} - 1$

### 16.4.8 EMIFA Interrupt Raw Register (INTRAW)

The EMIFA interrupt raw register (INTRAW) is used to monitor and clear the EMIFA's hardware-generated Asynchronous Timeout Interrupt. The AT bit in this register will be set when an Asynchronous Timeout occurs regardless of the status of the EMIFA interrupt mask set register (INTMSKSET) and EMIFA interrupt mask clear register (INTMSKCLR). Writing a 1 to this bit will clear it. The EMIFA on some devices does not have the EMA\_WAIT pin; therefore, these registers and fields are reserved on those devices. The INTRAW is shown in Figure 16-38 and described in Table 16-56.

**Figure 16-38. EMIFA Interrupt Raw Register (INTRAW)**

31	Reserved	8
	R-0	
7	3	2
	1	0
	Reserved	WR
	LT	AT
	R-0	R/W1C-0
	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 16-56. EMIFA Interrupt Raw Register (INTRAW) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR	0 1	Wait Rise. This bit is set to 1 by hardware to indicate that a rising edge on the EMA_WAIT pin has occurred.  0 Indicates that a rising edge has not occurred on the EMA_WAIT pin. Writing a 0 has no effect. 1 Indicates that a rising edge has occurred on the EMA_WAIT pin. Writing a 1 will clear this bit and the WR_MASKED bit in the EMIFA interrupt masked register (INTMSK).
1	LT	0 1	Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size.  0 Writing a 0 has no effect. 1 Indicates that a line trap has occurred. Writing a 1 will clear this bit as well as the LT_MASKED bit in the EMIFA interrupt masked register (INTMSK).
0	AT	0 1	Asynchronous Timeout. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMA_WAIT pin did not go inactive within the number of cycles defined by the MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC).  0 Indicates that an Asynchronous Timeout has not occurred. Writing a 0 has no effect. 1 Indicates that an Asynchronous Timeout has occurred. Writing a 1 will clear this bit as well as the AT_MASKED bit in the EMIFA interrupt masked register (INTMSK).

### 16.4.9 EMIFA Interrupt Masked Register (INTMSK)

Like the EMIFA interrupt raw register (INTRAW), the EMIFA interrupt masked register (INTMSK) is used to monitor and clear the status of the EMIFA's hardware-generated Asynchronous Timeout Interrupt. The main difference between the two registers is that when the AT\_MASKED bit in this register is set, an active-high pulse will be sent to the CPU interrupt controller. Also, the AT\_MASKED bit field in INTMSK is only set to 1 if the associated interrupt has been enabled in the EMIFA interrupt mask set register (INTMSKSET). The EMIFA on some devices does not have the EMA\_WAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSK is shown in [Figure 16-39](#) and described in [Table 16-57](#).

**Figure 16-39. EMIFA Interrupt Mask Register (INTMSK)**

31	Reserved				8
R-0					
7	3	2	1	0	
Reserved			WR_MASKED	LT_MASKED	AT_MASKED
R-0			R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 16-57. EMIFA Interrupt Mask Register (INTMSK) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASKED	0	Wait Rise Masked. This bit is set to 1 by hardware to indicate a rising edge has occurred on the EMA_WAIT pin, provided that the WR_MASK_SET bit is set to 1 in the EMIFA interrupt mask set register (INTMSKSET).
		1	Indicates that a wait rise interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that a wait rise interrupt has been generated. Writing a 1 will clear this bit and the WR bit in the EMIFA interrupt raw register (INTRAW).
1	LT_MASKED	0	Masked Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size, only if the LT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET) is set to 1.
		1	Writing a 0 has no effect.
		1	Writing a 1 will clear this bit as well as the LT bit in the EMIFA interrupt raw register (INTRAW).
0	AT_MASKED	0	Asynchronous Timeout Masked. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMA_WAIT pin did not go inactive within the number of cycles defined by the MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC), provided that the AT_MASK_SET bit is set to 1 in the EMIFA interrupt mask set register (INTMSKSET).
		1	Indicates that an Asynchronous Timeout Interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that an Asynchronous Timeout Interrupt has been generated. Writing a 1 will clear this bit as well as the AT bit in the EMIFA interrupt raw register (INTRAW).

### 16.4.10 EMIFA Interrupt Mask Set Register (INTMSKSET)

The EMIFA interrupt mask set register (INTMSKSET) is used to enable the Asynchronous Timeout Interrupt. If read as 1, the AT\_MASKED bit in the EMIFA interrupt masked register (INTMSK) will be set and an interrupt will be generated when an Asynchronous Timeout occurs. If read as 0, the AT\_MASKED bit will always read 0 and no interrupt will be generated when an Asynchronous Timeout occurs. Writing a 1 to the AT\_MASK\_SET bit enables the Asynchronous Timeout Interrupt. The EMIFA on some devices does not have the EMA\_WAIT pin; therefore, these registers and fields are reserved on those devices. The INTMSKSET is shown in Figure 16-40 and described in Table 16-58.

**Figure 16-40. EMIFA Interrupt Mask Set Register (INTMSKSET)**

31	Reserved				16
R-0					
15	3	2	1	0	
Reserved		WR_MASK_SET	Reserved	AT_MASK_SET	
R-0		R/W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-58. EMIFA Interrupt Mask Set Register (INTMSKSET) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASK_SET	0	Indicates that the wait rise interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the wait rise interrupt is enabled. Writing a 1 sets this bit and the WR_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR).
1	LT_MASK_SET	0	Indicates that the line trap interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the line trap interrupt is enabled. Writing a 1 sets this bit and the LT_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR).
0	AT_MASK_SET	0	Indicates that the Asynchronous Timeout Interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the Asynchronous Timeout Interrupt is enabled. Writing a 1 sets this bit and the AT_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR).

### 16.4.11 EMIFA Interrupt Mask Clear Register (INTMSKCLR)

The EMIFA interrupt mask clear register (INTMSKCLR) is used to disable the Asynchronous Timeout Interrupt. If read as 1, the AT\_MASKED bit in the EMIFA interrupt masked register (INTMSK) will be set and an interrupt will be generated when an Asynchronous Timeout occurs. If read as 0, the AT\_MASKED bit will always read 0 and no interrupt will be generated when an Asynchronous Timeout occurs. Writing a 1 to the AT\_MASK\_CLR bit disables the Asynchronous Timeout Interrupt. The EMIFA on some devices does not have the EMA\_WAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSKCLR is shown in [Figure 16-41](#) and described in [Table 16-59](#).

**Figure 16-41. EMIFA Interrupt Mask Clear Register (INTMSKCLR)**

31	Reserved				16
R-0					
15	3	2	1	0	
Reserved		WR_MASK_CLR	Reserved	AT_MASK_CLR	
R-0		R/W-0	R-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-59. EMIFA Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions**

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASK_CLR	0 1	Wait Rise Mask Clear. This bit determines whether or not the wait rise interrupt is enabled. Writing a 1 to this bit clears this bit, clears the WR_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET), and disables the wait rise interrupt. To set this bit, a 1 must be written to the WR_MASK_SET bit in INTMSKSET.  0 Indicates that the wait rise interrupt is disabled. Writing a 0 has no effect. 1 Indicates that the wait rise interrupt is enabled. Writing a 1 clears this bit and the WR_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).
1	LT_MASK_CLR	0 1	Line trap Mask Clear. This bit determines whether or not the line trap interrupt is enabled. Writing a 1 to this bit clears this bit, clears the LT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET), and disables the line trap interrupt. To set this bit, a 1 must be written to the LT_MASK_SET bit in INTMSKSET.  0 Indicates that the line trap interrupt is disabled. Writing a 0 has no effect. 1 Indicates that the line trap interrupt is enabled. Writing a 1 clears this bit and the LT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).
0	AT_MASK_CLR	0 1	Asynchronous Timeout Mask Clear. This bit determines whether or not the Asynchronous Timeout Interrupt is enabled. Writing a 1 to this bit clears this bit, clears the AT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET), and disables the Asynchronous Timeout Interrupt. To set this bit, a 1 must be written to the AT_MASK_SET bit of the EMIFA interrupt mask set register (INTMSKSET).  0 Indicates that the Asynchronous Timeout Interrupt is disabled. Writing a 0 has no effect. 1 Indicates that the Asynchronous Timeout Interrupt is enabled. Writing a 1 clears this bit and the AT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).

### 16.4.12 NAND Flash Control Register (NANDFCR)

The NAND Flash control register (NANDFCR) is shown in [Figure 16-42](#) and described in [Table 16-60](#).

**Figure 16-42. NAND Flash Control Register (NANDFCR)**

31	Reserved						16
R-0							
15	14	13	12	11	10	9	8
Reserved	4BITECC_ADD_CALC_START	4BITECC_START	CS5ECC	CS4ECC	CS3ECC	CS2ECC	
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved	4BITECCSEL		CS5NAND	CS4NAND	CS3NAND	CS2NAND	
R-0	R/W-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-60. NAND Flash Control Register (NANDFCR) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	4BITECC_ADD_CALC_START	1	NAND Flash 4-bit ECC address and error value calculation Start. Set to 1 to start 4_bit ECC error address and error value calculation on read syndrome. This bit is cleared when any of the NAND Flash error address registers or NAND Flash error value registers are read.
12	4BITECC_START	1	NAND Flash 4-bit ECC start for the selected chip select. Set to 1 to start 4_bit ECC calculation on data for NAND Flash on chip select selected by bit 4BITECCSEL. This bit is cleared when any of the NAND Flash 4_bit ECC registers are read.
11	CS5ECC	0 1	NAND Flash ECC start for chip select 5. Set to 1 to start 1_bit ECC calculation on data for NAND Flash for this chip select. This bit is cleared when CS5 1_bit ECC register is read. 0 Do not start ECC calculation. 1 Start ECC calculation on data for NAND Flash on <u>EMA_CS5</u> .
10	CS4ECC	0 1	NAND Flash ECC start for chip select 4. Set to 1 to start 1_bit ECC calculation on data for NAND Flash for this chip select. This bit is cleared when CS4 1_bit ECC register is read. 0 Do not start ECC calculation. 1 Start ECC calculation on data for NAND Flash on <u>EMA_CS4</u> .
9	CS3ECC	0 1	NAND Flash ECC start for chip select 3. Set to 1 to start 1_bit ECC calculation on data for NAND Flash for this chip select. This bit is cleared when CS3 1_bit ECC register is read. 0 Do not start ECC calculation. 1 Start ECC calculation on data for NAND Flash on <u>EMA_CS3</u> .
8	CS2ECC	0 1	NAND Flash ECC start for chip select 2. This bit is cleared when CS2 1_bit ECC register is read. 0 Do not start ECC calculation. 1 Start ECC calculation on data for NAND Flash on <u>EMA_CS2</u> .
7-6	Reserved	0	Reserved

**Table 16-60. NAND Flash Control Register (NANDFCR) Field Descriptions (continued)**

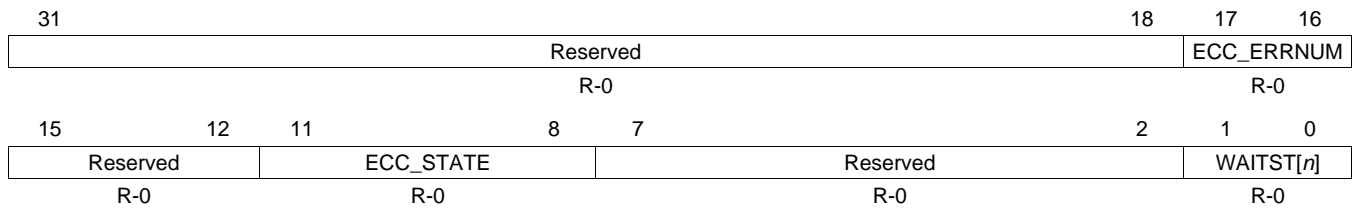
Bit	Field	Value	Description
5-4	4BITECCSEL	0-3h 0 1h 2h 3h	4-bit ECC selection. This field selects the chip select on which 4-bit ECC will be calculated. ECC will be calculated for CS2. ECC will be calculated for CS3. ECC will be calculated for CS4. ECC will be calculated for CS5.
3	CS5NAND	0 1	NAND Flash mode for chip select 5. Not using NAND Flash. Using NAND Flash on $\overline{\text{EMA\_CS5}}$ .
2	CS4NAND	0 1	NAND Flash mode for chip select 4. Not using NAND Flash. Using NAND Flash on $\overline{\text{EMA\_CS4}}$ .
1	CS3NAND	0 1	NAND Flash mode for chip select 3. Not using NAND Flash. Using NAND Flash on $\overline{\text{EMA\_CS3}}$ .
0	CS2NAND	0 1	NAND Flash mode for chip select 2. Not using NAND Flash. Using NAND Flash on $\overline{\text{EMA\_CS2}}$ .



### 16.4.13 NAND Flash Status Register (NANDFSR)

The NAND Flash status register (NANDFSR) is shown in [Figure 16-43](#) and described in [Table 16-61](#).

**Figure 16-43. NAND Flash Status Register (NANDFSR)**



LEGEND: R = Read only; -n = value after reset

**Table 16-61. NAND Flash Status Register (NANDFSR) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved.
17-16	ECC_ERRNUM	0-3h	Number of Errors found after the 4-Bit ECC Error Address and Error Value Calculation.
		0	1 error found.
		1h	2 errors found.
		2h	3 errors found.
		3h	4 errors found.
15-12	Reserved	0	Reserved.
11-8	ECC_STATE	0-Fh	ECC correction state while performing 4-bit ECC Address and Error Value Calculation
		0	No errors detected
		1h	Errors cannot be corrected (5 or more)
		2h	Error correction complete(errors on bit 8 or 9).
		3h	Error correction complete(error exists).
		4h	Reserved.
		5h	Calculating number of errors
		6h-7h	Preparing for error search
		8h	Searching for errors
		9h-Bh	Reserved.
		Ch-Fh	Calculating error value
7-2	Reserved	0	Reserved.
1-0	WAITST[n]		Status of the EMA_WAIT[n] input pins. Not all devices support both EMA_WAIT[1] and EMA_WAIT[0], see the device-specific data manual to determine support on each device. The WPn bit in the asynchronous wait cycle configuration register (AWCC) has no effect on WAITST.
		0	EMA_WAIT[n] pin is low.
		1	EMA_WAIT[n] pin is high.

### 16.4.14 NAND Flash *n* ECC Registers (NANDF1ECC-NANDF4ECC)

The NAND Flash *n* ECC register (NANDF*n*ECC) is shown in Figure 16-44 and described in Table 16-62. For 8-bit NAND Flash, the P1 to P4 bits are column parities; the P8 to P2048 bits are row parities. For 16-bit NAND Flash, the P1 to P8 bits are column parities; the P16 to P2048 bits are row parities.

**Figure 16-44. NAND Flash *n* ECC Register (NANDF*n*ECC)**

31		28				27	26	25	24
Reserved				P2048O		P1024O	P512O	P256O	
R-0				R-0		R-0	R-0	R-0	
23	22	21	20	19	18	17	16		
P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O		
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		
15		12				11	10	9	8
Reserved				P2048E		P1024E	P512E	P256E	
R-0				R-0		R-0	R-0	R-0	
7	6	5	4	3	2	1	0		
P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E		
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		

LEGEND: R = Read only; -*n* = value after reset

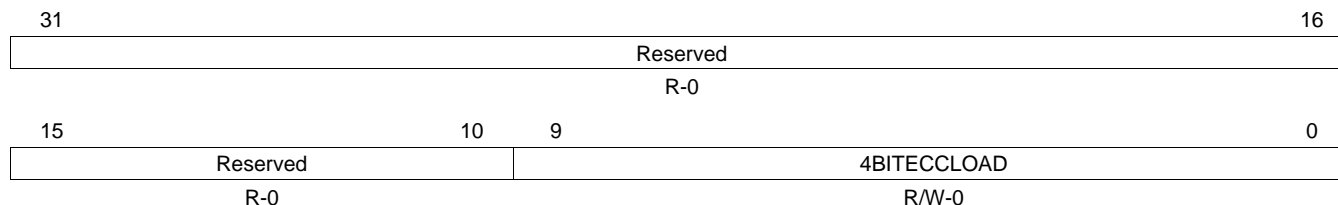
**Table 16-62. NAND Flash *n* ECC Register (NANDF*n*ECC) Field Descriptions**

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	P2048O	0-1	ECC code calculated while reading/writing NAND Flash.
26	P1024O	0-1	ECC code calculated while reading/writing NAND Flash.
25	P512O	0-1	ECC code calculated while reading/writing NAND Flash.
24	P256O	0-1	ECC code calculated while reading/writing NAND Flash.
23	P128O	0-1	ECC code calculated while reading/writing NAND Flash.
22	P64O	0-1	ECC code calculated while reading/writing NAND Flash.
21	P32O	0-1	ECC code calculated while reading/writing NAND Flash.
20	P16O	0-1	ECC code calculated while reading/writing NAND Flash.
19	P8O	0-1	ECC code calculated while reading/writing NAND Flash.
18	P4O	0-1	ECC code calculated while reading/writing NAND Flash.
17	P2O	0-1	ECC code calculated while reading/writing NAND Flash.
16	P1O	0-1	ECC code calculated while reading/writing NAND Flash.
15-12	Reserved	0	Reserved
11	P2948E	0-1	ECC code calculated while reading/writing NAND Flash.
10	P102E	0-1	ECC code calculated while reading/writing NAND Flash.
9	P512E	0-1	ECC code calculated while reading/writing NAND Flash.
8	P256E	0-1	ECC code calculated while reading/writing NAND Flash.
7	P128E	0-1	ECC code calculated while reading/writing NAND Flash.
6	P64E	0-1	ECC code calculated while reading/writing NAND Flash.
5	P32E	0-1	ECC code calculated while reading/writing NAND Flash.
4	P15E	0-1	ECC code calculated while reading/writing NAND Flash.
3	P8E	0-1	ECC code calculated while reading/writing NAND Flash.
2	P4E	0-1	ECC code calculated while reading/writing NAND Flash.
1	P2E	0-1	ECC code calculated while reading/writing NAND Flash.
0	P1E	0-1	ECC code calculated while reading/writing NAND Flash.

### 16.4.15 NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD)

The NAND Flash 4-bit ECC load register (NAND4BITECCLOAD) is shown in [Figure 16-45](#) and described in [Table 16-63](#).

**Figure 16-45. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-63. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	4BITECCLOAD	0-3FFh	4-bit ECC load. This value is used to load the ECC values when performing the Syndrome calculation during reads.

### 16.4.16 NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1)

The NAND Flash 4-bit ECC register 1 (NAND4BITECC1) is shown in [Figure 16-46](#) and described in [Table 16-64](#).

**Figure 16-46. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1)**

31	26	25	16
Reserved		4BITECCVAL2	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCVAL1	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-64. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL2	0-3FFh	Calculated 4-bit ECC or Syndrom Value2.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL1	0-3FFh	Calculated 4-bit ECC or Syndrom Value1.

### 16.4.17 NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2)

The NAND Flash 4-bit ECC register 2 (NAND4BITECC2) is shown in [Figure 16-47](#) and described in [Table 16-65](#).

**Figure 16-47. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2)**

31	26	25	16
Reserved		4BITECCVAL4	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCVAL3	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-65. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL4	0-3FFh	Calculated 4-bit ECC or Syndrom Value4.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL3	0-3FFh	Calculated 4-bit ECC or Syndrom Value3.

### 16.4.18 NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3)

The NAND Flash 4-bit ECC register 3 (NAND4BITECC3) is shown in [Figure 16-48](#) and described in [Table 16-66](#).

**Figure 16-48. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3)**

31	26	25	16
Reserved		4BITECCVAL6	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCVAL5	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-66. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL6	0-3FFh	Calculated 4-bit ECC or Syndrom Value6.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL5	0-3FFh	Calculated 4-bit ECC or Syndrom Value5.

### 16.4.19 NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4)

The NAND Flash 4-bit ECC register 4 (NAND4BITECC4) is shown in [Figure 16-49](#) and described in [Table 16-67](#).

**Figure 16-49. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4)**

31	26	25	16
Reserved		4BITECCVAL8	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCVAL7	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-67. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL8	0-3FFh	Calculated 4-bit ECC or Syndrom Value8.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL7	0-3FFh	Calculated 4-bit ECC or Syndrom Value7.

### 16.4.20 NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1)

The NAND Flash 4-bit ECC error register 1 (NANDERRADD1) is shown in [Figure 16-50](#) and described in [Table 16-68](#).

**Figure 16-50. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1)**

31	26	25	16
Reserved		4BITECCERRADD2	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCERRADD1	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-68. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRADD2	0-3FFh	Calculated 4-bit ECC Error Address 2.
15-10	Reserved	0	Reserved
9-0	4BITECCERRADD1	0-3FFh	Calculated 4-bit ECC Error Address 1.

### 16.4.21 NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2)

The NAND Flash 4-bit ECC error register 2 (NANDERRADD2) is shown in [Figure 16-51](#) and described in [Table 16-69](#).

**Figure 16-51. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2)**

31	26	25	16
Reserved		4BITECCERRADD4	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCERRADD3	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-69. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRADD4	0-3FFh	Calculated 4-bit ECC Error Address 4.
15-10	Reserved	0	Reserved
9-0	4BITECCERRADD3	0-3FFh	Calculated 4-bit ECC Error Address 3.

### 16.4.22 NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1)

The NAND Flash 4-bit ECC error value register 1 (NANDERRVAL1) is shown in [Figure 16-52](#) and described in [Table 16-70](#).

**Figure 16-52. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1)**

31	26	25	16
Reserved		4BITECCERRVAL2	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCERRVAL1	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-70. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRVAL2	0-3FFh	Calculated 4-bit ECC Error Value 2.
15-10	Reserved	0	Reserved
9-0	4BITECCERRVAL1	0-3FFh	Calculated 4-bit ECC Error Value 1.

### 16.4.23 NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2)

The NAND Flash 4-bit ECC error value register 2 (NANDERRVAL2) is shown in [Figure 16-53](#) and described in [Table 16-71](#).

**Figure 16-53. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2)**

31	26	25	16
Reserved		4BITECCERRVAL4	
R-0		R/W-0	
15	10	9	0
Reserved		4BITECCERRVAL3	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 16-71. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRVAL4	0-3FFh	Calculated 4-bit ECC Error Value 4.
15-10	Reserved	0	Reserved
9-0	4BITECCERRVAL3	0-3FFh	Calculated 4-bit ECC Error Value 3.

## General-Purpose Input/Output (GPIO)

---

---

The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an output, you can write to an internal register to control the state driven on the output pin. When configured as an input, you can detect the state of the input by reading the state of an internal register. This chapter describes the GPIO.

Topic	Page
<b>17.1 Introduction</b> .....	<b>670</b>
<b>17.2 Architecture</b> .....	<b>671</b>
<b>17.3 Registers</b> .....	<b>679</b>



## 17.1 Introduction

### 17.1.1 Purpose of the Peripheral

Most system-on-chip (SoC) devices require some general-purpose input/output (GPIO) functionality in order to interact with other components in the system using low-speed interface pins. The control and use of the GPIO capability on this device is grouped together in the GPIO peripheral and is described in the following sections.

### 17.1.2 Features

The GPIO peripheral consists of the following features.

- Output set/clear functionality through separate data set and clear registers allows multiple software processes to control GPIO signals without critical section protection.
- Set/clear functionality through writing to a single output data register is also supported.
- Separate input/output registers
  - Output register can be read to reflect output drive status.
  - Input register can be read to reflect pin status.
- All GPIO signals can be used as interrupt sources with configurable edge detection.
- All GPIO signals can be used to generate events to the EDMA.

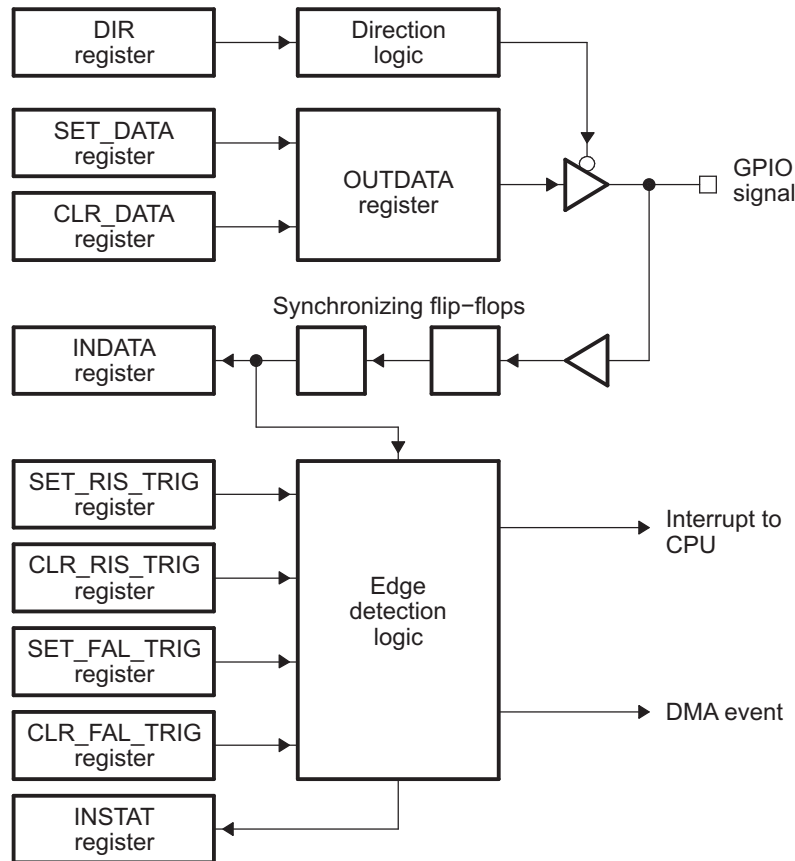
### 17.1.3 Functional Block Diagram

[Figure 17-1](#) shows a block diagram of the GPIO peripheral.

### 17.1.4 Industry Standard(s) Compliance Statement

The GPIO peripheral connects to external devices. While it is possible that the software implements some standard connectivity protocol over GPIO, the GPIO peripheral itself is not compliant with any such standards.

Figure 17-1. GPIO Block Diagram



## 17.2 Architecture

The following sections describe the GPIO peripheral.

### 17.2.1 Clock Control

The input clock to the GPIO peripheral is indicated in the device datasheet. The maximum operating speed of the GPIO peripheral is limited by system-level latencies. More specifically, how quickly the GPIO registers can be written to or read from.

### 17.2.2 Signal Descriptions

The number of GPIO signals supported will vary between devices. For information on the number of signals supported and the package pinout of each GPIO signal, see your device-specific data manual.

### 17.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data manual to determine how pin multiplexing affects the GPIO module.

### 17.2.4 Endianness Considerations

The GPIO operation is independent of endianness; therefore, there are no endianness considerations for the GPIO module.

### 17.2.5 GPIO Register Structure

The GPIO signals are grouped by banks of 16 signals per bank. Each bank of GPIO signals has several registers with various control fields for each GPIO signal. Each 32-bit GPIO control register controls a pair of GPIO banks.

The register names for each bank of control registers (or pair of banks of GPIO bits) are all of the form *register\_nameXY*, where *X* and *Y* are the two banks of GPIO bits controlled, such as 01, 23, 45, etc. The register fields associated with each GPIO are all of the form *BkPj*, where *k* is the GPIO bank and *j* is the pin number within the GPIO bank. For example, for GP2[5], which is located in GPIO bank 2, the control register names are of the form *register\_name23*, and the register field associated with GP2[5] is GP2P5.

Table 17-1 shows the banks and register control bit information associated with each GPIO pin for up to 144 supportable pins. The table is not indicative of how many GPIO pins are supported on a device; it is only a reference for what register and field mappings look like for the first 144 supportable GPIO pins. For devices with less than 144 GPIO pins, assume that the extraneous fields and registers listed in the table are Reserved with no function. For devices with more than 144 GPIO pins, additional control registers and fields should be appended using the same numbering scheme in the table. Detailed information regarding the specific register names for each bank and the contents and function of these registers is presented in Section 17.3.

**Table 17-1. GPIO Register Bits and Banks Associated With GPIO Signals**

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
1	GP0[0]	0	<i>register_name01</i>	Bit 0	GP0P0
2	GP0[1]	0	<i>register_name01</i>	Bit 1	GP0P1
3	GP0[2]	0	<i>register_name01</i>	Bit 2	GP0P2
4	GP0[3]	0	<i>register_name01</i>	Bit 3	GP0P3
5	GP0[4]	0	<i>register_name01</i>	Bit 4	GP0P4
6	GP0[5]	0	<i>register_name01</i>	Bit 5	GP0P5
7	GP0[6]	0	<i>register_name01</i>	Bit 6	GP0P6
8	GP0[7]	0	<i>register_name01</i>	Bit 7	GP0P7
9	GP0[8]	0	<i>register_name01</i>	Bit 8	GP0P8
10	GP0[9]	0	<i>register_name01</i>	Bit 9	GP0P9
11	GP0[10]	0	<i>register_name01</i>	Bit 10	GP0P10
12	GP0[11]	0	<i>register_name01</i>	Bit 11	GP0P11
13	GP0[12]	0	<i>register_name01</i>	Bit 12	GP0P12
14	GP0[13]	0	<i>register_name01</i>	Bit 13	GP0P13
15	GP0[14]	0	<i>register_name01</i>	Bit 14	GP0P14
16	GP0[15]	0	<i>register_name01</i>	Bit 15	GP0P15
17	GP1[0]	1	<i>register_name01</i>	Bit 16	GP1P0
18	GP1[1]	1	<i>register_name01</i>	Bit 17	GP1P1
19	GP1[2]	1	<i>register_name01</i>	Bit 18	GP1P2
20	GP1[3]	1	<i>register_name01</i>	Bit 19	GP1P3
21	GP1[4]	1	<i>register_name01</i>	Bit 20	GP1P4
22	GP1[5]	1	<i>register_name01</i>	Bit 21	GP1P5
23	GP1[6]	1	<i>register_name01</i>	Bit 22	GP1P6
24	GP1[7]	1	<i>register_name01</i>	Bit 23	GP1P7
25	GP1[8]	1	<i>register_name01</i>	Bit 24	GP1P8
26	GP1[9]	1	<i>register_name01</i>	Bit 25	GP1P9
27	GP1[10]	1	<i>register_name01</i>	Bit 26	GP1P10
28	GP1[11]	1	<i>register_name01</i>	Bit 27	GP1P11
29	GP1[12]	1	<i>register_name01</i>	Bit 28	GP1P12
30	GP1[13]	1	<i>register_name01</i>	Bit 29	GP1P13

**Table 17-1. GPIO Register Bits and Banks Associated With GPIO Signals (continued)**

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
31	GP1[14]	1	<i>register_name01</i>	Bit 30	GP1P14
32	GP1[15]	1	<i>register_name01</i>	Bit 31	GP1P15
33	GP2[0]	2	<i>register_name23</i>	Bit 0	GP2P0
34	GP2[1]	2	<i>register_name23</i>	Bit 1	GP2P1
35	GP2[2]	2	<i>register_name23</i>	Bit 2	GP2P2
36	GP2[3]	2	<i>register_name23</i>	Bit 3	GP2P3
37	GP2[4]	2	<i>register_name23</i>	Bit 4	GP2P4
38	GP2[5]	2	<i>register_name23</i>	Bit 5	GP2P5
39	GP2[6]	2	<i>register_name23</i>	Bit 6	GP2P6
40	GP2[7]	2	<i>register_name23</i>	Bit 7	GP2P7
41	GP2[8]	2	<i>register_name23</i>	Bit 8	GP2P8
42	GP2[9]	2	<i>register_name23</i>	Bit 9	GP2P9
43	GP2[10]	2	<i>register_name23</i>	Bit 10	GP2P10
44	GP2[11]	2	<i>register_name23</i>	Bit 11	GP2P11
45	GP2[12]	2	<i>register_name23</i>	Bit 12	GP2P12
46	GP2[13]	2	<i>register_name23</i>	Bit 13	GP2P13
47	GP2[14]	2	<i>register_name23</i>	Bit 14	GP2P14
48	GP2[15]	2	<i>register_name23</i>	Bit 15	GP2P15
49	GP3[0]	3	<i>register_name23</i>	Bit 16	GP3P0
50	GP3[1]	3	<i>register_name23</i>	Bit 17	GP3P1
51	GP3[2]	3	<i>register_name23</i>	Bit 18	GP3P2
52	GP3[3]	3	<i>register_name23</i>	Bit 19	GP3P3
53	GP3[4]	3	<i>register_name23</i>	Bit 20	GP3P4
54	GP3[5]	3	<i>register_name23</i>	Bit 21	GP3P5
55	GP3[6]	3	<i>register_name23</i>	Bit 22	GP3P6
56	GP3[7]	3	<i>register_name23</i>	Bit 23	GP3P7
57	GP3[8]	3	<i>register_name23</i>	Bit 24	GP3P8
58	GP3[9]	3	<i>register_name23</i>	Bit 25	GP3P9
59	GP3[10]	3	<i>register_name23</i>	Bit 26	GP3P10
60	GP3[11]	3	<i>register_name23</i>	Bit 27	GP3P11
61	GP3[12]	3	<i>register_name23</i>	Bit 28	GP3P12
62	GP3[13]	3	<i>register_name23</i>	Bit 29	GP3P13
63	GP3[14]	3	<i>register_name23</i>	Bit 30	GP3P14
64	GP3[15]	3	<i>register_name23</i>	Bit 31	GP3P15
65	GP4[0]	4	<i>register_name45</i>	Bit 0	GP4P0
66	GP4[1]	4	<i>register_name45</i>	Bit 1	GP4P1
67	GP4[2]	4	<i>register_name45</i>	Bit 2	GP4P2
68	GP4[3]	4	<i>register_name45</i>	Bit 3	GP4P3
69	GP4[4]	4	<i>register_name45</i>	Bit 4	GP4P4
70	GP4[5]	4	<i>register_name45</i>	Bit 5	GP4P5
71	GP4[6]	4	<i>register_name45</i>	Bit 6	GP4P6
72	GP4[7]	4	<i>register_name45</i>	Bit 7	GP4P7
73	GP4[8]	4	<i>register_name45</i>	Bit 8	GP4P8
74	GP4[9]	4	<i>register_name45</i>	Bit 9	GP4P9
75	GP4[10]	4	<i>register_name45</i>	Bit 10	GP4P10
76	GP4[11]	4	<i>register_name45</i>	Bit 11	GP4P11
77	GP4[12]	4	<i>register_name45</i>	Bit 12	GP4P12

**Table 17-1. GPIO Register Bits and Banks Associated With GPIO Signals (continued)**

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
78	GP4[13]	4	<i>register_name45</i>	Bit 13	GP4P13
79	GP4[14]	4	<i>register_name45</i>	Bit 14	GP4P14
80	GP4[15]	4	<i>register_name45</i>	Bit 15	GP4P15
81	GP5[0]	5	<i>register_name45</i>	Bit 16	GP5P0
82	GP5[1]	5	<i>register_name45</i>	Bit 17	GP5P1
83	GP5[2]	5	<i>register_name45</i>	Bit 18	GP5P2
84	GP5[3]	5	<i>register_name45</i>	Bit 19	GP5P3
85	GP5[4]	5	<i>register_name45</i>	Bit 20	GP5P4
86	GP5[5]	5	<i>register_name45</i>	Bit 21	GP5P5
87	GP5[6]	5	<i>register_name45</i>	Bit 22	GP5P6
88	GP5[7]	5	<i>register_name45</i>	Bit 23	GP5P7
89	GP5[8]	5	<i>register_name45</i>	Bit 24	GP5P8
90	GP5[9]	5	<i>register_name45</i>	Bit 25	GP5P9
91	GP5[10]	5	<i>register_name45</i>	Bit 26	GP5P10
92	GP5[11]	5	<i>register_name45</i>	Bit 27	GP5P11
93	GP5[12]	5	<i>register_name45</i>	Bit 28	GP5P12
94	GP5[13]	5	<i>register_name45</i>	Bit 29	GP5P13
95	GP5[14]	5	<i>register_name45</i>	Bit 30	GP5P14
96	GP5[15]	5	<i>register_name45</i>	Bit 31	GP5P15
97	GP6[0]	6	<i>register_name67</i>	Bit 0	GP6P0
98	GP6[1]	6	<i>register_name67</i>	Bit 1	GP6P1
99	GP6[2]	6	<i>register_name67</i>	Bit 2	GP6P2
100	GP6[3]	6	<i>register_name67</i>	Bit 3	GP6P3
101	GP6[4]	6	<i>register_name67</i>	Bit 4	GP6P4
102	GP6[5]	6	<i>register_name67</i>	Bit 5	GP6P5
103	GP6[6]	6	<i>register_name67</i>	Bit 6	GP6P6
104	GP6[7]	6	<i>register_name67</i>	Bit 7	GP6P7
105	GP6[8]	6	<i>register_name67</i>	Bit 8	GP6P8
106	GP6[9]	6	<i>register_name67</i>	Bit 9	GP6P9
107	GP6[10]	6	<i>register_name67</i>	Bit 10	GP6P10
108	GP6[11]	6	<i>register_name67</i>	Bit 11	GP6P11
109	GP6[12]	6	<i>register_name67</i>	Bit 12	GP6P12
110	GP6[13]	6	<i>register_name67</i>	Bit 13	GP6P13
111	GP6[14]	6	<i>register_name67</i>	Bit 14	GP6P14
112	GP6[15]	6	<i>register_name67</i>	Bit 15	GP6P15
113	GP7[0]	7	<i>register_name67</i>	Bit 16	GP7P0
114	GP7[1]	7	<i>register_name67</i>	Bit 17	GP7P1
115	GP7[2]	7	<i>register_name67</i>	Bit 18	GP7P2
116	GP7[3]	7	<i>register_name67</i>	Bit 19	GP7P3
117	GP7[4]	7	<i>register_name67</i>	Bit 20	GP7P4
118	GP7[5]	7	<i>register_name67</i>	Bit 21	GP7P5
119	GP7[6]	7	<i>register_name67</i>	Bit 22	GP7P6
120	GP7[7]	7	<i>register_name67</i>	Bit 23	GP7P7
121	GP7[8]	7	<i>register_name67</i>	Bit 24	GP7P8
122	GP7[9]	7	<i>register_name67</i>	Bit 25	GP7P9
123	GP7[10]	7	<i>register_name67</i>	Bit 26	GP7P10
124	GP7[11]	7	<i>register_name67</i>	Bit 27	GP7P11

**Table 17-1. GPIO Register Bits and Banks Associated With GPIO Signals (continued)**

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
125	GP7[12]	7	<i>register_name67</i>	Bit 28	GP7P12
126	GP7[13]	7	<i>register_name67</i>	Bit 29	GP7P13
127	GP7[14]	7	<i>register_name67</i>	Bit 30	GP7P14
128	GP7[15]	7	<i>register_name67</i>	Bit 31	GP7P15
129	GP8[0]	8	<i>register_name8</i>	Bit 0	GP8P0
130	GP8[1]	8	<i>register_name8</i>	Bit 1	GP8P1
131	GP8[2]	8	<i>register_name8</i>	Bit 2	GP8P2
132	GP8[3]	8	<i>register_name8</i>	Bit 3	GP8P3
133	GP8[4]	8	<i>register_name8</i>	Bit 4	GP8P4
134	GP8[5]	8	<i>register_name8</i>	Bit 5	GP8P5
135	GP8[6]	8	<i>register_name8</i>	Bit 6	GP8P6
136	GP8[7]	8	<i>register_name8</i>	Bit 7	GP8P7
137	GP8[8]	8	<i>register_name8</i>	Bit 8	GP8P8
138	GP8[9]	8	<i>register_name8</i>	Bit 9	GP8P9
139	GP8[10]	8	<i>register_name8</i>	Bit 10	GP8P10
140	GP8[11]	8	<i>register_name8</i>	Bit 11	GP8P11
141	GP8[12]	8	<i>register_name8</i>	Bit 12	GP8P12
142	GP8[13]	8	<i>register_name8</i>	Bit 13	GP8P13
143	GP8[14]	8	<i>register_name8</i>	Bit 14	GP8P14
144	GP8[15]	8	<i>register_name8</i>	Bit 15	GP8P15

### 17.2.6 Using a GPIO Signal as an Output

GPIO signals are configured to operate as inputs or outputs by writing the appropriate value to the GPIO direction register (DIR). This section describes using the GPIO signal as an output signal.

#### 17.2.6.1 Configuring a GPIO Output Signal

To configure a given GPIO signal as an output, clear the bit in DIR that is associated with the desired GPIO signal. For detailed information on DIR, see [Section 17.3](#).

#### 17.2.6.2 Controlling the GPIO Output Signal State

There are three registers that control the output state driven on a GPIO signal configured as an output:

1. GPIO set data register (SET\_DATA) controls driving GPIO signals high.
2. GPIO clear data register (CLR\_DATA) controls driving GPIO signals low.
3. GPIO output data register (OUT\_DATA) contains the current state of the output signals.

Reading SET\_DATA, CLR\_DATA, and OUT\_DATA returns the output state, not necessarily the actual signal state (since some signals may be configured as inputs). The actual signal state is read using the GPIO input data register (IN\_DATA) associated with the desired GPIO signal. IN\_DATA contains the actual logic state on the external signal.

For detailed information on these registers, see [Section 17.3](#).

### 17.2.6.2.1 Driving a GPIO Output Signal High

To drive a GPIO signal high, use one of the following methods:

- Write a logic 1 to the bit in SET\_DATA associated with the desired GPIO signal(s) to be driven high. Bit positions in SET\_DATA containing logic 0 do not affect the state of the associated output signals.
- Modify the bit in OUT\_DATA associated with the desired GPIO signal by using a read-modify-write operation. The logic states driven on the GPIO output signals match the logic values written to all bits in OUT\_DATA.

For GPIO signals configured as inputs, the values written to the associated SET\_DATA, CLR\_DATA, and OUT\_DATA bits have no effect.

### 17.2.6.2.2 Driving a GPIO Output Signal Low

To drive a GPIO signal low, use one of the following methods:

- Write a logic 1 to the bit in CLR\_DATA associated with the desired GPIO signal(s) to be driven low. Bit positions in CLR\_DATA containing logic 0 do not affect the state of the associated output signals.
- Modify the bit in OUT\_DATA associated with the desired GPIO signal by using a read-modify-write operation. The logic states driven on the GPIO output signals match the logic values written to all bits in OUT\_DATA.

For GPIO signals configured as inputs, the values written to the associated SET\_DATA, CLR\_DATA, and OUT\_DATA bits have no effect.

## 17.2.7 Using a GPIO Signal as an Input

GPIO signals are configured to operate as inputs or outputs by writing the appropriate value to the GPIO direction register (DIR). This section describes using the GPIO signal as an input signal.

### 17.2.7.1 Configuring a GPIO Input Signal

To configure a given GPIO signal as an input, set the bit in DIR that is associated with the desired GPIO signal. For detailed information on DIR, see [Section 17.3](#).

### 17.2.7.2 Reading a GPIO Input Signal

The current state of the GPIO signals is read using the GPIO input data register (IN\_DATA).

- For GPIO signals configured as inputs, reading IN\_DATA returns the state of the input signal synchronized to the GPIO peripheral clock.
- For GPIO signals configured as outputs, reading IN\_DATA returns the output value being driven by the device.

Some signals may utilize open-drain output buffers for wired-logic operations. For open-drain GPIO signals, reading IN\_DATA returns the wired-logic value on the signal (which will not be driven by the device alone). Information on any signals using open-drain outputs is available in your device-specific data manual.

To use GPIO input signals as interrupt sources, see [Section 17.2.10](#).

## 17.2.8 Reset Considerations

The GPIO peripheral has two reset sources: software reset and hardware reset.

### 17.2.8.1 Software Reset Considerations

A software reset (such as a reset initiated through the emulator) does not modify the configuration and state of the GPIO signals. A reset invoked via the Power and Sleep Controller (PSC) (GPIO clock disable, PSC reset, followed by GPIO clock enable) will result in the default configuration register settings. For details on the PSC, see the *Power and Sleep Controller (PSC)* chapter.



### 17.2.8.2 Hardware Reset Considerations

A hardware reset does reset the GPIO configuration and data registers to their default states; therefore, affecting the configuration and state of the GPIO signals.

### 17.2.9 Initialization

The following steps are required to configure the GPIO module after a hardware reset:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Program the Power and Sleep Controller (PSC) to enable the GPIO module. For details on the PSC, see the *Power and Sleep Controller (PSC)* chapter.
3. Program the direction, data, and interrupt control registers to set the configuration of the desired GPIO pins (described in this chapter).

The GPIO module is now ready to perform data transactions.

### 17.2.10 Interrupt Support

The GPIO peripheral can send an interrupt event to the CPU.

#### 17.2.10.1 Interrupt Events and Requests

All GPIO signals can be configured to generate interrupts. The device supports interrupts from single GPIO signals, interrupts from banks of GPIO signals, or both.

Note that the GPIO interrupts may also be used to provide synchronization events to the DMA controller.

#### 17.2.10.2 Enabling GPIO Interrupt Events

GPIO interrupt events are enabled in banks of 16 by setting the appropriate bit(s) in the GPIO interrupt per-bank enable register (BINTEN). For example, to enable bank 0 interrupts (events from GP0[15-0]), set bit 0 in BINTEN; to enable bank 3 interrupts (events from GP3[15-0]), set bit 3 in BINTEN.

For detailed information on BINTEN, see [Section 17.3](#).

#### 17.2.10.3 Configuring GPIO Interrupt Edge Triggering

Each GPIO interrupt source can be configured to generate an interrupt on the GPIO signal rising edge, falling edge, both edges, or neither edge (no event). The edge detection is synchronized to the GPIO peripheral module clock.

The following four registers control the configuration of the GPIO interrupt edge detection:

1. The GPIO set rising edge interrupt register (SET\_RIS\_TRIG) enables GPIO interrupts on the occurrence of a rising edge on the GPIO signal.
2. The GPIO clear rising edge interrupt register (CLR\_RIS\_TRIG) disables GPIO interrupts on the occurrence of a rising edge on the GPIO signal.
3. The GPIO set falling edge interrupt register (SET\_FAL\_TRIG) enables GPIO interrupts on the occurrence of a falling edge on the GPIO signal.
4. The GPIO clear falling edge interrupt register (CLR\_FAL\_TRIG) disables GPIO interrupts on the occurrence of a falling edge on the GPIO signal.

To configure a GPIO interrupt to occur only on rising edges of the GPIO signal:

- Write a logic 1 to the associated bit in SET\_RIS\_TRIG.
- Write a logic 1 to the associated bit in CLR\_FAL\_TRIG.

To configure a GPIO interrupt to occur only on falling edges of the GPIO signal:

- Write a logic 1 to the associated bit in SET\_FAL\_TRIG.
- Write a logic 1 to the associated bit in CLR\_RIS\_TRIG.



To configure a GPIO interrupt to occur on both the rising and falling edges of the GPIO signal:

- Write a logic 1 to the associated bit in SET\_RIS\_TRIG.
- Write a logic 1 to the associated bit in SET\_FAL\_TRIG.

To disable a specific GPIO interrupt:

- Write a logic 1 to the associated bit in CLR\_RIS\_TRIG.
- Write a logic 1 to the associated bit in CLR\_FAL\_TRIG.

For detailed information on these registers, see [Section 17.3](#).

Note that the direction of the GPIO signal does not have to be an input for the interrupt event generation to work. When a GPIO signal is configured as an output, the software can change the GPIO signal state and, in turn, generate an interrupt. This can be useful for debugging interrupt signal connectivity.

#### 17.2.10.4 GPIO Interrupt Status

The status of GPIO interrupt events can be monitored by reading the GPIO interrupt status register (INTSTAT). Pending GPIO interrupts are indicated with a logic 1 in the associated bit position; interrupts that are not pending are indicated with a logic 0.

For individual GPIO interrupts that are directly routed to the DSP subsystem, the interrupt status can be read by reading the associated interrupt flag in the CPU. For the GPIO bank interrupts, INTSTAT can be used to determine which GPIO interrupt occurred. It is the responsibility of software to ensure that all pending GPIO interrupts are appropriately serviced.

Pending GPIO interrupt flags can be cleared by writing a logic 1 to the associated bit position in INTSTAT.

For detailed information on INTSTAT, see [Section 17.3](#).

#### 17.2.10.5 Interrupt Multiplexing

GPIO interrupts may be multiplexed with other interrupt functions on the device.

#### 17.2.11 EDMA Event Support

The GPIO peripheral may provide synchronization events to the DMA controller.

#### 17.2.12 Power Management

The GPIO peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the GPIO peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

When the GPIO peripheral is placed in a low-power state by the PSC, the interrupt generation capability is suspended until the GPIO peripheral is removed from the low-power state. While in the low-power state, the GPIO signals configured as outputs are maintained at their state prior to the GPIO peripheral entering the low-power state.

#### 17.2.13 Emulation Considerations

The GPIO peripheral is not affected by emulation suspend events (such as halts and breakpoints).

## 17.3 Registers

Table 17-2 lists the memory-mapped registers for the general-purpose input/output (GPIO). The table enumerates the registers required to support 144 GPIO pins, however not all devices will support 144 GPIO pins. For devices with less than 144 GPIO pins, assume that the extraneous fields and registers are Reserved and serve no function. For devices with more than 144 GPIO pins, append registers and fields as necessary using the address offset scheme in the table. See your device-specific data manual for the number of GPIO pins supported and the base memory address for these registers.

**Table 17-2. GPIO Registers**

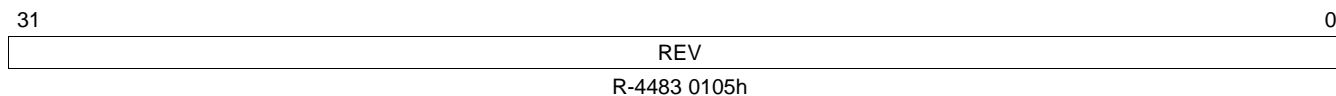
Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	<a href="#">Section 17.3.1</a>
8h	BINTEN	GPIO Interrupt Per-Bank Enable Register	<a href="#">Section 17.3.2</a>
<b>GPIO Banks 0 and 1</b>			
10h	DIR01	GPIO Banks 0 and 1 Direction Register	<a href="#">Section 17.3.3</a>
14h	OUT_DATA01	GPIO Banks 0 and 1 Output Data Register	<a href="#">Section 17.3.4</a>
18h	SET_DATA01	GPIO Banks 0 and 1 Set Data Register	<a href="#">Section 17.3.5</a>
1Ch	CLR_DATA01	GPIO Banks 0 and 1 Clear Data Register	<a href="#">Section 17.3.6</a>
20h	IN_DATA01	GPIO Banks 0 and 1 Input Data Register	<a href="#">Section 17.3.7</a>
24h	SET_RIS_TRIG01	GPIO Banks 0 and 1 Set Rising Edge Interrupt Register	<a href="#">Section 17.3.8</a>
28h	CLR_RIS_TRIG01	GPIO Banks 0 and 1 Clear Rising Edge Interrupt Register	<a href="#">Section 17.3.9</a>
2Ch	SET_FAL_TRIG01	GPIO Banks 0 and 1 Set Falling Edge Interrupt Register	<a href="#">Section 17.3.10</a>
30h	CLR_FAL_TRIG01	GPIO Banks 0 and 1 Clear Falling Edge Interrupt Register	<a href="#">Section 17.3.11</a>
34h	INTSTAT01	GPIO Banks 0 and 1 Interrupt Status Register	<a href="#">Section 17.3.12</a>
<b>GPIO Banks 2 and 3</b>			
38h	DIR23	GPIO Banks 2 and 3 Direction Register	<a href="#">Section 17.3.3</a>
3Ch	OUT_DATA23	GPIO Banks 2 and 3 Output Data Register	<a href="#">Section 17.3.4</a>
40h	SET_DATA23	GPIO Banks 2 and 3 Set Data Register	<a href="#">Section 17.3.5</a>
44h	CLR_DATA23	GPIO Banks 2 and 3 Clear Data Register	<a href="#">Section 17.3.6</a>
48h	IN_DATA23	GPIO Banks 2 and 3 Input Data Register	<a href="#">Section 17.3.7</a>
4Ch	SET_RIS_TRIG23	GPIO Banks 2 and 3 Set Rising Edge Interrupt Register	<a href="#">Section 17.3.8</a>
50h	CLR_RIS_TRIG23	GPIO Banks 2 and 3 Clear Rising Edge Interrupt Register	<a href="#">Section 17.3.9</a>
54h	SET_FAL_TRIG23	GPIO Banks 2 and 3 Set Falling Edge Interrupt Register	<a href="#">Section 17.3.10</a>
58h	CLR_FAL_TRIG23	GPIO Banks 2 and 3 Clear Falling Edge Interrupt Register	<a href="#">Section 17.3.11</a>
5Ch	INTSTAT23	GPIO Banks 2 and 3 Interrupt Status Register	<a href="#">Section 17.3.12</a>
<b>GPIO Banks 4 and 5</b>			
60h	DIR45	GPIO Banks 4 and 5 Direction Register	<a href="#">Section 17.3.3</a>
64h	OUT_DATA45	GPIO Banks 4 and 5 Output Data Register	<a href="#">Section 17.3.4</a>
68h	SET_DATA45	GPIO Banks 4 and 5 Set Data Register	<a href="#">Section 17.3.5</a>
6Ch	CLR_DATA45	GPIO Banks 4 and 5 Clear Data Register	<a href="#">Section 17.3.6</a>
70h	IN_DATA45	GPIO Banks 4 and 5 Input Data Register	<a href="#">Section 17.3.7</a>
74h	SET_RIS_TRIG45	GPIO Banks 4 and 5 Set Rising Edge Interrupt Register	<a href="#">Section 17.3.8</a>
78h	CLR_RIS_TRIG45	GPIO Banks 4 and 5 Clear Rising Edge Interrupt Register	<a href="#">Section 17.3.9</a>
7Ch	SET_FAL_TRIG45	GPIO Banks 4 and 5 Set Falling Edge Interrupt Register	<a href="#">Section 17.3.10</a>
80h	CLR_FAL_TRIG45	GPIO Banks 4 and 5 Clear Falling Edge Interrupt Register	<a href="#">Section 17.3.11</a>
84h	INTSTAT45	GPIO Banks 4 and 5 Interrupt Status Register	<a href="#">Section 17.3.12</a>

**Table 17-2. GPIO Registers (continued)**

Offset	Acronym	Register Description	Section
<b>GPIO Banks 6 and 7</b>			
88h	DIR67	GPIO Banks 6 and 7 Direction Register	<a href="#">Section 17.3.3</a>
8Ch	OUT_DATA67	GPIO Banks 6 and 7 Output Data Register	<a href="#">Section 17.3.4</a>
90h	SET_DATA67	GPIO Banks 6 and 7 Set Data Register	<a href="#">Section 17.3.5</a>
94h	CLR_DATA67	GPIO Banks 6 and 7 Clear Data Register	<a href="#">Section 17.3.6</a>
98h	IN_DATA67	GPIO Banks 6 and 7 Input Data Register	<a href="#">Section 17.3.7</a>
9Ch	SET_RIS_TRIG67	GPIO Banks 6 and 7 Set Rising Edge Interrupt Register	<a href="#">Section 17.3.8</a>
A0h	CLR_RIS_TRIG67	GPIO Banks 6 and 7 Clear Rising Edge Interrupt Register	<a href="#">Section 17.3.9</a>
A4h	SET_FAL_TRIG67	GPIO Banks 6 and 7 Set Falling Edge Interrupt Register	<a href="#">Section 17.3.10</a>
A8h	CLR_FAL_TRIG67	GPIO Banks 6 and 7 Clear Falling Edge Interrupt Register	<a href="#">Section 17.3.11</a>
ACH	INTSTAT67	GPIO Banks 6 and 7 Interrupt Status Register	<a href="#">Section 17.3.12</a>
<b>GPIO Bank 8</b>			
B0h	DIR8	GPIO Bank 8 Direction Register	<a href="#">Section 17.3.3</a>
B4h	OUT_DATA8	GPIO Bank 8 Output Data Register	<a href="#">Section 17.3.4</a>
B8h	SET_DATA8	GPIO Bank 8 Set Data Register	<a href="#">Section 17.3.5</a>
BCh	CLR_DATA8	GPIO Bank 8 Clear Data Register	<a href="#">Section 17.3.6</a>
C0h	IN_DATA8	GPIO Bank 8 Input Data Register	<a href="#">Section 17.3.7</a>
C4h	SET_RIS_TRIG8	GPIO Bank 8 Set Rising Edge Interrupt Register	<a href="#">Section 17.3.8</a>
C8h	CLR_RIS_TRIG8	GPIO Bank 8 Clear Rising Edge Interrupt Register	<a href="#">Section 17.3.9</a>
CCh	SET_FAL_TRIG8	GPIO Bank 8 Set Falling Edge Interrupt Register	<a href="#">Section 17.3.10</a>
D0h	CLR_FAL_TRIG8	GPIO Bank 8 Clear Falling Edge Interrupt Register	<a href="#">Section 17.3.11</a>
D4h	INTSTAT8	GPIO Bank 8 Interrupt Status Register	<a href="#">Section 17.3.12</a>

### 17.3.1 Revision ID Register (REVID)

The revision ID register (REVID) contains the peripheral version information. REVID is shown in [Figure 17-2](#) and described in [Table 17-3](#).

**Figure 17-2. Revision ID Register (REVID)**

LEGEND: R = Read only; -n = value after reset

**Table 17-3. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4483 0105h	Peripheral Revision

### 17.3.2 GPIO Interrupt Per-Bank Enable Register (BINTEN)

The GPIO interrupt per-bank enable register (BINTEN) is shown in [Figure 17-3](#) and described in [Table 17-4](#). For information on which GPIO signals are associated with each bank, see [Table 17-1](#). Note that the bits in BINTEN control both the interrupt and EDMA events.

**Figure 17-3. GPIO Interrupt Per-Bank Enable Register (BINTEN)**

31	Reserved										16
R-0											
15	9	8	7	6	5	4	3	2	1	0	
Reserved		EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 17-4. GPIO Interrupt Per-Bank Enable Register (BINTEN) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	EN8	0	Bank 8 interrupt enable is used to disable or enable the bank 8 interrupts (events from GP8[15-0]). Bank 8 interrupts are disabled.
		1	Bank 8 interrupts are enabled.
7	EN7	0	Bank 7 interrupt enable is used to disable or enable the bank 7 interrupts (events from GP7[15-0]). Bank 7 interrupts are disabled.
		1	Bank 7 interrupts are enabled.
6	EN6	0	Bank 6 interrupt enable is used to disable or enable the bank 6 interrupts (events from GP6[15-0]). Bank 6 interrupts are disabled.
		1	Bank 6 interrupts are enabled.
5	EN5	0	Bank 5 interrupt enable is used to disable or enable the bank 5 interrupts (events from GP5[15-0]). Bank 5 interrupts are disabled.
		1	Bank 5 interrupts are enabled.
4	EN4	0	Bank 4 interrupt enable is used to disable or enable the bank 4 interrupts (events from GP4[15-0]). Bank 4 interrupts are disabled.
		1	Bank 4 interrupts are enabled.
3	EN3	0	Bank 3 interrupt enable is used to disable or enable the bank 3 interrupts (events from GP3[15-0]). Bank 3 interrupts are disabled.
		1	Bank 3 interrupts are enabled.
2	EN2	0	Bank 2 interrupt enable is used to disable or enable the bank 2 interrupts (events from GP2[15-0]). Bank 2 interrupts are disabled.
		1	Bank 2 interrupts are enabled.
1	EN1	0	Bank 1 interrupt enable is used to disable or enable the bank 1 interrupts (events from GP1[15-0]). Bank 1 interrupts are disabled.
		1	Bank 1 interrupts are enabled.
0	EN0	0	Bank 0 interrupt enable is used to disable or enable the bank 0 interrupts (events from GP0[15-0]). Bank 0 interrupts are disabled.
		1	Bank 0 interrupts are enabled.

### 17.3.3 GPIO Direction Registers (DIRn)

The GPIO direction register (DIRn) determines if GPIO pin *j* in GPIO bank *k* is an input or an output. Each of the GPIO banks may have up to 16 GPIO pins. By default, all the GPIO pins are configured as inputs (bit value = 1). The GPIO direction register (DIR01) is shown in Figure 17-4, DIR23 is shown in Figure 17-5, DIR45 is shown in Figure 17-6, DIR67 is shown in Figure 17-7, DIR8 is shown in Figure 17-8, and described in Table 17-5. See Table 17-1 to determine the DIRn bit associated with each GPIO bank and pin number.

**Figure 17-4. GPIO Banks 0 and 1 Direction Register (DIR01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-1															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-5. GPIO Banks 2 and 3 Direction Register (DIR23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-1															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-6. GPIO Banks 4 and 5 Direction Register (DIR45)**

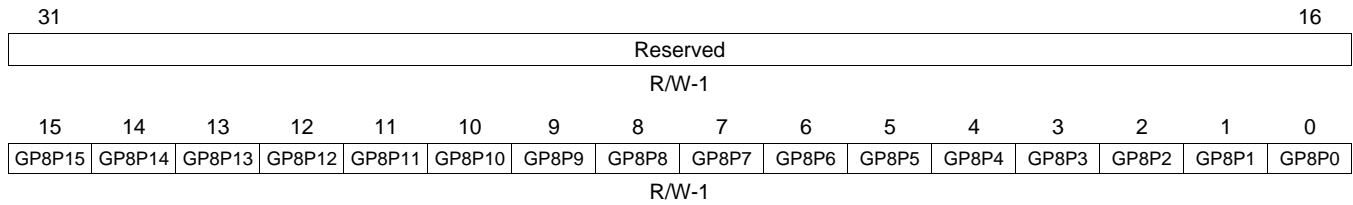
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-1															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-7. GPIO Banks 6 and 7 Direction Register (DIR67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-1															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-8. GPIO Bank 8 Direction Register (DIR8)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-5. GPIO Direction Register (DIR<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkPj	0	Direction of pin GPk[j]. The GPkPj bit is used to control the direction (output = 0, input = 1) of pin j in GPIO bank k.
		1	GPk[j] is an output.
			GPk[j] is an input.

### 17.3.4 GPIO Output Data Registers (OUT\_DATA $n$ )

The GPIO output data register (OUT\_DATA $n$ ) determines the value driven on the corresponding GPIO pin  $j$  in GPIO bank  $k$ , if the pin is configured as an output (DIR $n$  = 0). Writes do not affect pins not configured as GPIO outputs. The bits in OUT\_DATA $n$  are set or cleared by writing directly to this register. A read of OUT\_DATA $n$  returns the value of the register not the value at the pin (that might be configured as an input). The GPIO output data register (OUT\_DATA01) is shown in [Figure 17-9](#), OUT\_DATA23 is shown in [Figure 17-10](#), OUT\_DATA45 is shown in [Figure 17-11](#), OUT\_DATA67 is shown in [Figure 17-12](#), OUT\_DATA8 is shown in [Figure 17-13](#), and described in [Table 17-6](#). See [Table 17-1](#) to determine the OUT\_DATA $n$  bit associated with each GPIO bank and pin number.

**Figure 17-9. GPIO Banks 0 and 1 Output Data Register (OUT\_DATA01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-10. GPIO Banks 2 and 3 Output Data Register (OUT\_DATA23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-11. GPIO Banks 4 and 5 Output Data Register (OUT\_DATA45)**

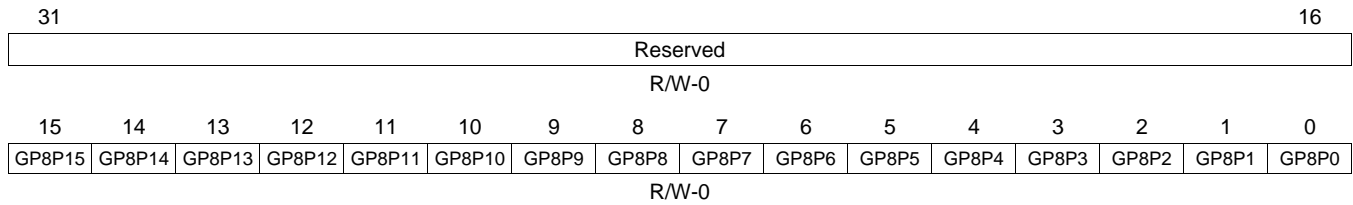
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-12. GPIO Banks 6 and 7 Output Data Register (OUT\_DATA67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-13. GPIO Bank 8 Output Data Register (OUT\_DATA8)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-6. GPIO Output Data Register (OUT\_DATA<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkP <sub>j</sub>	0	Output drive state of GPk[j]. The GPkP <sub>j</sub> bit is used to drive the output (low = 0, high = 1) of pin <i>j</i> in GPIO bank <i>k</i> . The GPkP <sub>j</sub> bit is ignored when GPk[j] is configured as an input.
		1	GPk[j] is driven low.
			GPk[j] is driven high.



### 17.3.5 GPIO Set Data Registers (SET\_DATA<sub>n</sub>)

The GPIO set data register (SET\_DATA<sub>n</sub>) controls driving high of the corresponding GPIO pin *j* in GPIO bank *k*, if the pin is configured as an output (DIR<sub>n</sub> = 0). Writes do not affect pins not configured as GPIO outputs. Writing a 1 to a specific bit in SET\_DATA<sub>n</sub> sets the corresponding GPIO pin *j* in GPIO bank *k*. A read of the BkPj bit returns the output drive state of the corresponding pin GPIOk[j]. The GPIO set data register (SET\_DATA01) is shown in Figure 17-14, SET\_DATA23 is shown in Figure 17-15, SET\_DATA45 is shown in Figure 17-16, SET\_DATA67 is shown in Figure 17-17, SET\_DATA8 is shown in Figure 17-18, and described in Table 17-7. See Table 17-1 to determine the SET\_DATA<sub>n</sub> bit associated with each GPIO bank and pin number.

**Figure 17-14. GPIO Banks 0 and 1 Set Data Register (SET\_DATA01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-15. GPIO Banks 2 and 3 Set Data Register (SET\_DATA23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-16. GPIO Banks 4 and 5 Set Data Register (SET\_DATA45)**

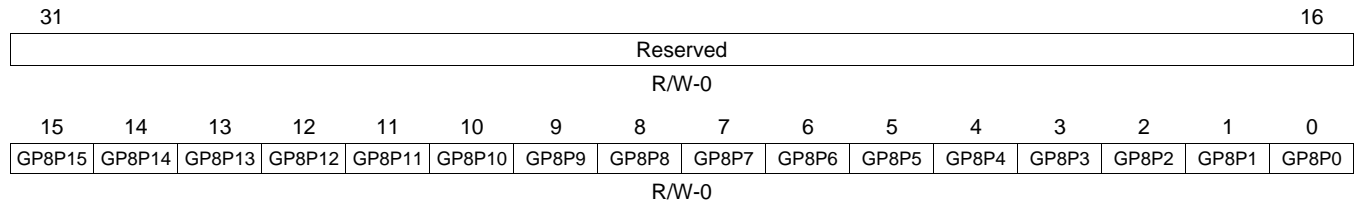
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-17. GPIO Banks 6 and 7 Set Data Register (SET\_DATA67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-18. GPIO Bank 8 Set Data Register (SET\_DATA8)**

LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-7. GPIO Set Data Register (SET\_DATA<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkP <sub>j</sub>	0	Set the output drive state of GPk[j] to logic high. The GPkP <sub>j</sub> bit is used to drive the output high on pin <i>j</i> in GPIO bank <i>k</i> . The GPkP <sub>j</sub> bit is ignored when GPk[j] is configured as an input. Reading the GPkP <sub>j</sub> bit returns the output drive state of GPk[j].
		1	No effect.
			GPk[j] is set to output logic high.

### 17.3.6 GPIO Clear Data Registers (CLR\_DATA $n$ )

The GPIO clear data register (CLR\_DATA $n$ ) controls clearing low of the corresponding GPIO pin  $j$  in GPIO bank  $k$ , if the pin is configured as an output (DIR $n$  = 0). Writes do not affect pins not configured as GPIO outputs. Writing a 1 to a specific bit in CLR\_DATA $n$  resets the corresponding GPIO pin  $j$  in GPIO bank  $k$ . A read of the BkPj bit returns the output drive state of the corresponding pin GPIOk[j]. The GPIO clear data register (CLR\_DATA01) is shown in Figure 17-19, CLR\_DATA23 is shown in Figure 17-20, CLR\_DATA45 is shown in Figure 17-21, CLR\_DATA67 is shown in Figure 17-22, CLR\_DATA8 is shown in Figure 17-23, and described in Table 17-8. See Table 17-1 to determine the CLR\_DATA $n$  bit associated with each GPIO bank and pin number.

**Figure 17-19. GPIO Banks 0 and 1 Clear Data Register (CLR\_DATA01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-20. GPIO Banks 2 and 3 Clear Data Register (CLR\_DATA23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-21. GPIO Banks 4 and 5 Clear Data Register (CLR\_DATA45)**

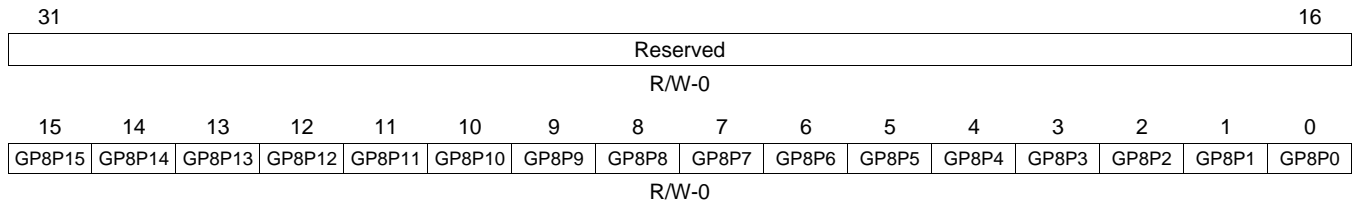
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-22. GPIO Banks 6 and 7 Clear Data Register (CLR\_DATA67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-23. GPIO Bank 8 Clear Data Register (CLR\_DATA8)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-8. GPIO Clear Data Register (CLR\_DATA<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkP <sub>j</sub>	0	Clear the output drive state of GPk[j] to logic low. The GPkP <sub>j</sub> bit is used to drive the output low on pin <i>j</i> in GPIO bank <i>k</i> . The GPkP <sub>j</sub> bit is ignored when GPk[j] is configured as an input. Reading the GPkP <sub>j</sub> bit returns the output drive state of GPk[j].  No effect.
		1	

### 17.3.7 GPIO Input Data Registers (IN\_DATA $n$ )

The current state of the GPIO signals is read using the GPIO input data register (IN\_DATA $n$ ).

- For GPIO signals configured as inputs, reading IN\_DATA $n$  returns the state of the input signal synchronized to the GPIO peripheral clock.
- For GPIO signals configured as outputs, reading IN\_DATA $n$  returns the output value being driven by the device.

The GPIO input data register (IN\_DATA01) is shown in [Figure 17-24](#), IN\_DATA23 is shown in [Figure 17-25](#), IN\_DATA45 is shown in [Figure 17-26](#), IN\_DATA67 is shown in [Figure 17-27](#), IN\_DATA8 is shown in [Figure 17-28](#), and described in [Table 17-9](#). See [Table 17-1](#) to determine the IN\_DATA $n$  bit associated with each GPIO bank and pin number.

**Figure 17-24. GPIO Banks 0 and 1 Input Data Register (IN\_DATA01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R-0															

LEGEND: R = Read only; - $n$  = value after reset

**Figure 17-25. GPIO Banks 2 and 3 Input Data Register (IN\_DATA23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R-0															

LEGEND: R = Read only; - $n$  = value after reset

**Figure 17-26. GPIO Banks 4 and 5 Input Data Register (IN\_DATA45)**

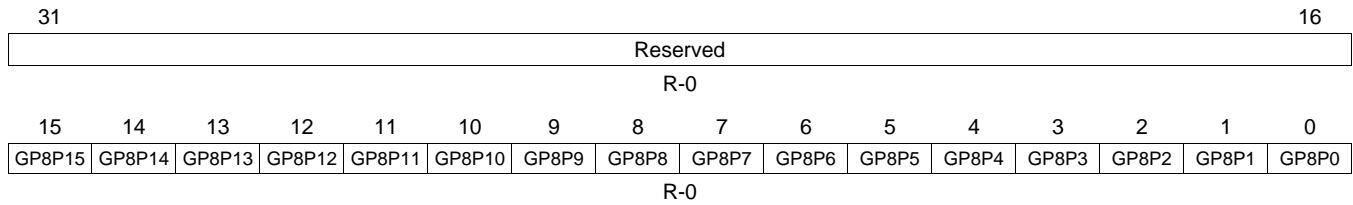
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R-0															

LEGEND: R = Read only; - $n$  = value after reset

**Figure 17-27. GPIO Banks 6 and 7 Input Data Register (IN\_DATA67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R-0															

LEGEND: R = Read only; - $n$  = value after reset

**Figure 17-28. GPIO Bank 8 Input Data Register (IN\_DATA8)**


LEGEND: R = Read only; -n = value after reset

**Table 17-9. GPIO Input Data Register (IN\_DATA<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkPj	0	Status of pin GPk[j]. Reading the GPkPj bit returns the state of pin j in GPIO bank k.
		1	GPk[j] is logic high.

### 17.3.8 GPIO Set Rising Edge Interrupt Registers (SET\_RIS\_TRIGn)

The GPIO set rising edge trigger interrupt register (SET\_RIS\_TRIGn) enables a rising edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO set rising edge interrupt register (SET\_RIS\_TRIG01) is shown in Figure 17-29, SET\_RIS\_TRIG23 is shown in Figure 17-30, SET\_RIS\_TRIG45 is shown in Figure 17-31, SET\_RIS\_TRIG67 is shown in Figure 17-32, SET\_RIS\_TRIG8 is shown in Figure 17-33, and described in Table 17-10. See Table 17-1 to determine the SET\_RIS\_TRIGn bit associated with each GPIO bank and pin number.

**Figure 17-29. GPIO Banks 0 and 1 Set Rise Trigger Register (SET\_RIS\_TRIG01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-30. GPIO Banks 2 and 3 Set Rise Trigger Register (SET\_RIS\_TRIG23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-31. GPIO Banks 4 and 5 Set Rise Trigger Register (SET\_RIS\_TRIG45)**

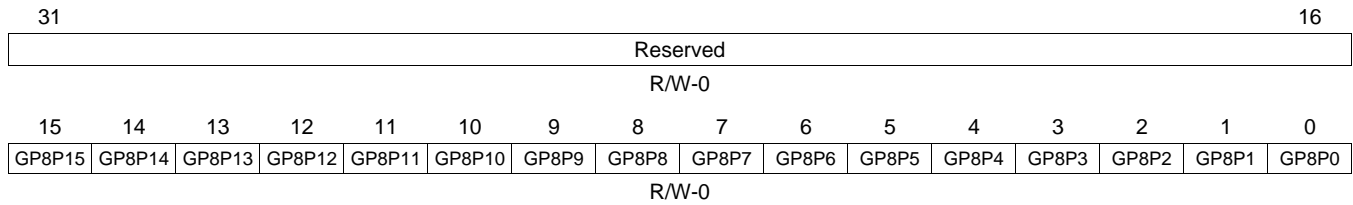
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-32. GPIO Banks 6 and 7 Set Rise Trigger Register (SET\_RIS\_TRIG67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-33. GPIO Bank 8 Set Rise Trigger Register (SET\_RIS\_TRIG8)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-10. GPIO Set Rising Edge Trigger Interrupt Register (SET\_RIS\_TRIG<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GP <sub>j</sub> P <sub>k</sub>	0 1	Enable rising edge trigger interrupt detection on GP <sub>k</sub> [ <sub>j</sub> ]. Reading the GP <sub>k</sub> P <sub>j</sub> bit in either SET_RIS_TRIG <sub>n</sub> or CLR_RIS_TRIG <sub>n</sub> always returns an indication of whether the rising edge interrupt generation function is enabled for pin GP <sub>k</sub> [ <sub>j</sub> ]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.  No effect.  Interrupt is caused by a low-to-high transition on GP <sub>k</sub> [ <sub>j</sub> ].



### 17.3.9 GPIO Clear Rising Edge Interrupt Registers (CLR\_RIS\_TRIGn)

The GPIO clear rising edge trigger interrupt register (CLR\_RIS\_TRIGn) disables the rising edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO clear rising edge interrupt register (CLR\_RIS\_TRIG01) is shown in [Figure 17-34](#), CLR\_RIS\_TRIG23 is shown in [Figure 17-35](#), CLR\_RIS\_TRIG45 is shown in [Figure 17-36](#), CLR\_RIS\_TRIG67 is shown in [Figure 17-37](#), CLR\_RIS\_TRIG8 is shown in [Figure 17-38](#), and described in [Table 17-11](#). See [Table 17-1](#) to determine the CLR\_RIS\_TRIGn bit associated with each GPIO bank and pin number.

**Figure 17-34. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR\_RIS\_TRIG01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-35. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR\_RIS\_TRIG23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-36. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR\_RIS\_TRIG45)**

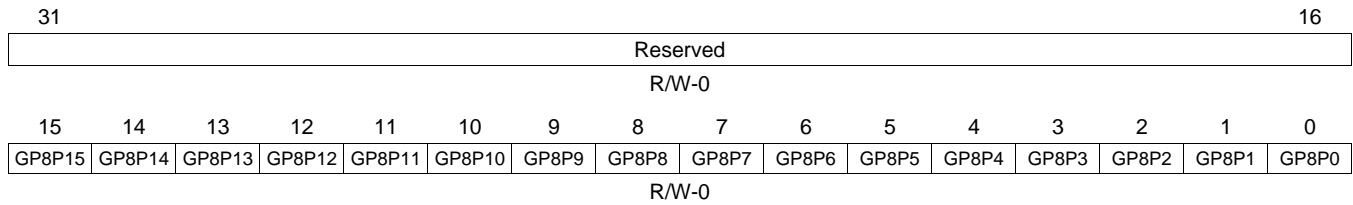
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-37. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR\_RIS\_TRIG67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-38. GPIO Bank 8 Clear Rise Trigger Register (CLR\_RIS\_TRIG8)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-11. GPIO Clear Rising Edge Interrupt Register (CLR\_RIS\_TRIG<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkPj	0	Disable rising edge interrupt detection on GPk[j]. Reading the GPkPj bit in either SET_RIS_TRIG <sub>n</sub> or CLR_RIS_TRIG <sub>n</sub> always returns an indication of whether the rising edge interrupt generation function is enabled for GPk[j]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		1	No effect.
		1	No interrupt is caused by a low-to-high transition on GPk[j].

### 17.3.10 GPIO Set Falling Edge Interrupt Registers (SET\_FAL\_TRIGn)

The GPIO set falling edge trigger interrupt register (SET\_FAL\_TRIGn) enables a falling edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO set falling edge interrupt register (SET\_FAL\_TRIG01) is shown in [Figure 17-39](#), SET\_FAL\_TRIG23 is shown in [Figure 17-40](#), SET\_FAL\_TRIG45 is shown in [Figure 17-41](#), SET\_FAL\_TRIG67 is shown in [Figure 17-42](#), SET\_FAL\_TRIG8 is shown in [Figure 17-43](#), and described in [Table 17-12](#). See [Table 17-1](#) to determine the SET\_FAL\_TRIGn bit associated with each GPIO bank and pin number.

**Figure 17-39. GPIO Banks 0 and 1 Set Rise Trigger Register (SET\_FAL\_TRIG01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-40. GPIO Banks 2 and 3 Set Rise Trigger Register (SET\_FAL\_TRIG23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-41. GPIO Banks 4 and 5 Set Rise Trigger Register (SET\_FAL\_TRIG45)**

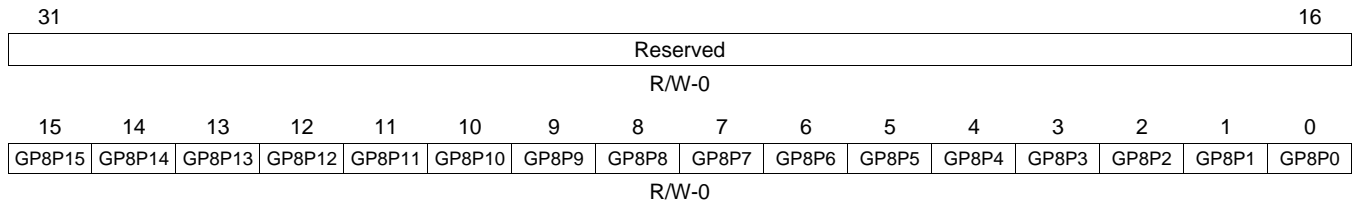
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-42. GPIO Banks 6 and 7 Set Rise Trigger Register (SET\_FAL\_TRIG67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 17-43. GPIO Bank 8 Set Rise Trigger Register (SET\_FAL\_TRIG8)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-12. GPIO Set Falling Edge Trigger Interrupt Register (SET\_FAL\_TRIG<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkPj	0	Enable falling edge trigger interrupt detection on GPk[j]. Reading the GPkPj bit in either SET_FAL_TRIG <sub>n</sub> or CLR_FAL_TRIG <sub>n</sub> always returns an indication of whether the falling edge interrupt generation function is enabled for pin GPk[j]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		1	No effect.
			Interrupt is caused by a high-to-low transition on GPk[j].

### 17.3.11 GPIO Clear Falling Edge Interrupt Registers (CLR\_FAL\_TRIG $n$ )

The GPIO clear falling edge trigger interrupt register (CLR\_FAL\_TRIG $n$ ) disables the falling edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO clear falling edge interrupt register (CLR\_FAL\_TRIG01) is shown in Figure 17-44, CLR\_FAL\_TRIG23 is shown in Figure 17-45, CLR\_FAL\_TRIG45 is shown in Figure 17-46, CLR\_FAL\_TRIG67 is shown in Figure 17-47, CLR\_FAL\_TRIG8 is shown in Figure 17-48, and described in Table 17-13. See Table 17-1 to determine the CLR\_FAL\_TRIG $n$  bit associated with each GPIO bank and pin number.

**Figure 17-44. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR\_FAL\_TRIG01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-45. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR\_FAL\_TRIG23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-46. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR\_FAL\_TRIG45)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-47. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR\_FAL\_TRIG67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

LEGEND: R/W = Read/Write; - $n$  = value after reset

**Figure 17-48. GPIO Bank 8 Clear Rise Trigger Register (CLR\_FAL\_TRIG8)**

31	Reserved														16
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

LEGEND: R/W = Read/Write; -n = value after reset

**Table 17-13. GPIO Clear Falling Edge Interrupt Register (CLR\_FAL\_TRIG<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkPj		Disable falling edge interrupt detection on GPk[j]. Reading the GPkPj bit in either SET_FAL_TRIG <sub>n</sub> or CLR_FAL_TRIG <sub>n</sub> always returns an indication of whether the falling edge interrupt generation function is enabled for GPk[j]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		0	No effect.
		1	No interrupt is caused by a high-to-low transition on GPk[j].

### 17.3.12 GPIO Interrupt Status Registers (INTSTAT $n$ )

The status of GPIO interrupt events can be monitored by reading the GPIO interrupt status register (INTSTAT $n$ ). In the associated bit position, pending GPIO interrupts are indicated with a logic 1 and GPIO interrupts that are not pending are indicated with a logic 0. The GPIO interrupt status register (INTSTAT01) is shown in [Figure 17-49](#), INTSTAT23 is shown in [Figure 17-50](#), INTSTAT45 is shown in [Figure 17-51](#), INTSTAT67 is shown in [Figure 17-52](#), INTSTAT8 is shown in [Figure 17-53](#), and described in [Table 17-14](#). See [Table 17-1](#) to determine the INTSTAT $n$  bit associated with each GPIO bank and pin number.

**Figure 17-49. GPIO Banks 0 and 1 Interrupt Status Register (INTSTAT01)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W1C-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); - $n$  = value after reset

**Figure 17-50. GPIO Banks 2 and 3 Interrupt Status Register (INTSTAT23)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W1C-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); - $n$  = value after reset

**Figure 17-51. GPIO Banks 4 and 5 Interrupt Status Register (INTSTAT45)**

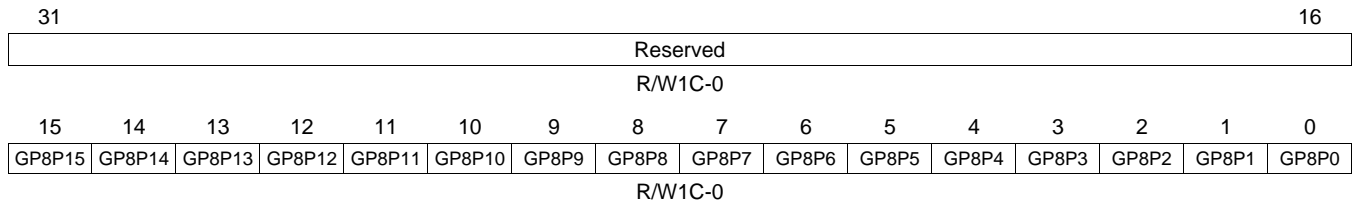
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W1C-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); - $n$  = value after reset

**Figure 17-52. GPIO Banks 6 and 7 Interrupt Status Register (INTSTAT67)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W1C-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); - $n$  = value after reset

**Figure 17-53. GPIO Bank 8 Interrupt Status Register (INTSTAT8)**


LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); -n = value after reset

**Table 17-14. GPIO Interrupt Status Register (INTSTAT<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-0	GPkPj	0	Interrupt status of GPk[j]. The GPkPj bit is used to monitor pending GPIO interrupts on pin j of GPIO bank k. Write a 1 to the GPkPj bit to clear the status bit; a write of 0 has no effect.
		1	No pending interrupt on GPk[j].
			Pending interrupt on GPk[j].



## Host Port Interface (HPI)

---

---

This chapter describes the host port interface (HPI).

Topic	Page
<b>18.1 Introduction</b> .....	<b>703</b>
<b>18.2 Architecture</b> .....	<b>706</b>
<b>18.3 Registers</b> .....	<b>726</b>

## 18.1 Introduction

The host port interface (HPI) provides a parallel port interface through which an external host processor can directly access the processor's resources (configuration and program/data memories). The external host device is asynchronous to the CPU clock and functions as a master to the HPI interface. The HPI enables a host device and the processor to exchange information via internal or external memory. Dedicated address (HPIA) and data (HPID) registers within the HPI provide the data path between the external host interface and the processor resources. An HPI control register (HPIC) is available to the host and the CPU for various configuration and interrupt functions.

### 18.1.1 Purpose of the Peripheral

The HPI enables an external host processor (host) to directly access program/data memory on the processor using a parallel interface. The primary purpose is to provide a mechanism to move data to and from the processor. In addition to data transfer, the host can also use the HPI to bootload the processor by downloading program and data information to the processor's memory after power-up.

### 18.1.2 Features

The HPI supports the following features:

- Multiplexed address/data
- Dual 16-bit halfword cycle access (internal data word is 32-bits wide)
- 16-bit-wide host data bus interface
- Internal data bursting using 8-word read and write first-in, first-out (FIFO) buffers
- HPI control register (HPIC) accessible by both the DSP CPU and the external host
- HPI address register (HPIA) accessible by both the DSP CPU and the external host
- Separate HPI address registers for read (HPIAR) and write (HPIAW) with configurable option for operating as a single HPI address register
- HPI data register (HPID)/FIFOs providing data-path between external host interface and CPU resources
- Multiple strobes and control signals to allow flexible host connection
- Asynchronous UHPI\_HRDY output to allow the HPI to insert wait states to the host
- Software control of data prefetching to the HPID/FIFOs
- Processor-to-Host interrupt output signal controlled by HPIC accesses
- Host-to-Processor interrupt controlled by HPIC accesses
- Register controlled HPIA and HPIC ownership and FIFO timeout
- Memory-mapped peripheral identification register (PID)
- Bus holders on host data and address buses (these are actually external to HPI module)

### 18.1.3 Functional Block Diagram

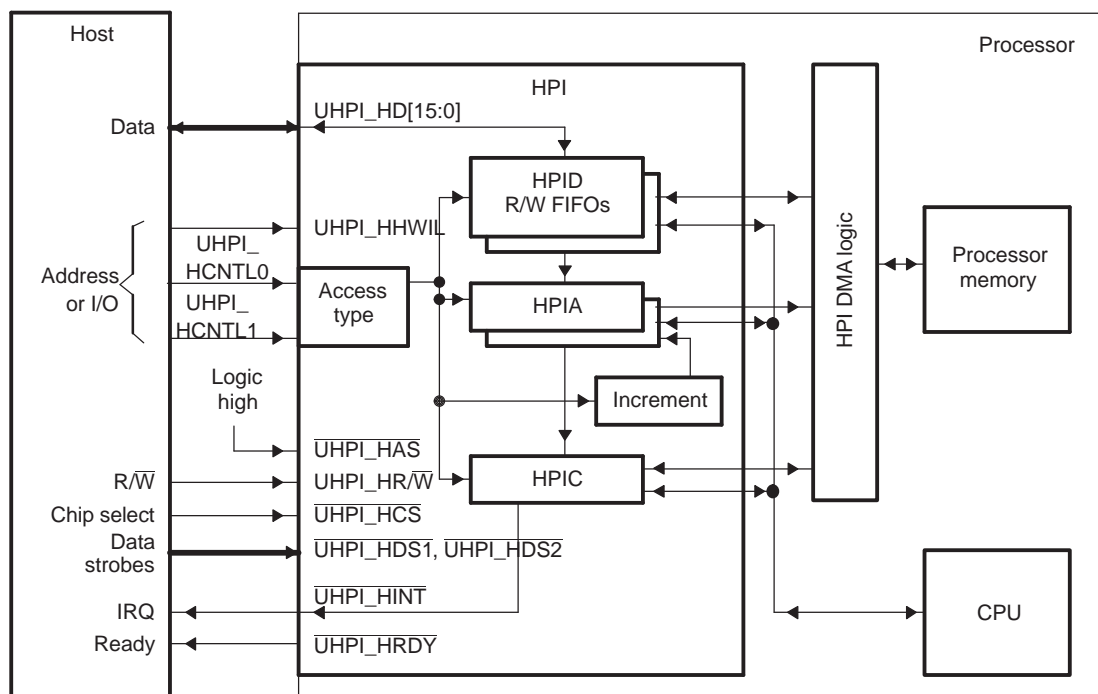
Figure 18-1 is a high-level block diagram showing how the HPI connects a host (left side of figure) and the processor internal memory (right side of figure). Host activity is asynchronous to the internal processor clock that drives the HPI. The host functions as a master to the HPI. When HPI resources are temporarily busy or unavailable, the HPI communicates this to the host by deasserting the HPI ready ( $\overline{\text{UHPI\_HRDY}}$ ) output signal.

The HPI supports multiplexed operation meaning the data bus is used for both address and data. Each host cycle consists of two consecutive 16-bit transfers. When the host drives an address on the bus, the address is stored in a 32-bit address register (HPIA) in the HPI, so that the bus can then be used for data. The HPI contains two address registers (HPIAR and HPIAW), which can be used as separate address registers for read accesses and write accesses (for details, see Section 18.2.6.1).

A control register (HPIC) is accessible by the CPU and the host. The CPU uses HPIC to send an interrupt request to the host, to clear an interrupt request from the host, and to monitor the HPI. The host uses HPIC to configure and monitor the HPI, to send an interrupt request to the CPU, and to clear an interrupt request from the CPU.

Data flow between the host and the HPI uses a temporary storage register, the 32-bit data register (HPID). Data arriving from the host is held in HPID until the data can be stored elsewhere in the processor. Data to be sent to the host is held in HPID until the HPI is ready to perform the transfer. When address autoincrementing is used, read and write FIFOs are used to store burst data. If autoincrementing is not used, the FIFO memory acts as a single register (only one location is used).

**Figure 18-1. HPI Block Diagram**



### 18.1.4 Industry Standard(s) Compliance Statement

The HPI is not an industry standard interface that is developed and monitored by an international organization. It is a generic parallel interface that can be configured to gluelessly interface to a variety of parallel devices.

### 18.1.5 Terminology Used in This Document

The following is a brief explanation of some terms used in this document:

Term	Meaning
CPU	DSP CPU
host	External host device
HPI DMA logic	Logic used to communicate between the HPI and the DMA system that moves data to and from memory. This is independent of the EDMA system on the processor
processor	Entire system-on-chip

## 18.2 Architecture

### 18.2.1 Clock Control

For detailed information on the PLLs and clock distribution on the processor, see the *Phase-Locked Loop Controller (PLLCC)* chapter.

### 18.2.2 Memory Map

The HPI can be used by the host to access on-chip device memory, peripheral, and memory-mapped registers. See your device-specific data manual for more detailed information.

### 18.2.3 Signal Descriptions

Table 18-1 shows the a description of the HPI signals.

**Table 18-1. HPI Pins**

Pin	Type	Host Connection	Function
UHPI_HCNTL[1:0]	I	Address or control pins	<b>HPI access control inputs.</b> The HPI latches the logic levels of these pins on the falling edge of internal $\overline{\text{HSTRB}}$ (for details about internal $\overline{\text{HSTRB}}$ , see Section 18.2.6.4). The four binary states of these pins determine the access type of the current transfer (HPIC, HPIA, HPID with and without autoincrementing).
$\overline{\text{UHPI\_HCS}}$	I	Chip select pin	<b>HPI chip select.</b> $\overline{\text{UHPI\_HCS}}$ must be low for the HPI to be selected by the host. $\overline{\text{UHPI\_HCS}}$ can be kept low between accesses. $\overline{\text{UHPI\_HCS}}$ normally precedes an active $\overline{\text{UHPI\_HDS}}$ (data strobe) signal, but can be connected to a $\overline{\text{UHPI\_HDS}}$ pin for simultaneous select and strobe activity.
UHPI_HR/ $\overline{\text{W}}$	I	R/W strobe pin	<b>HPI read/write.</b> On the falling edge of internal $\overline{\text{HSTRB}}$ , UHPI_HR/ $\overline{\text{W}}$ indicates whether the current access is to be a read or write operation. Driving UHPI_HR/ $\overline{\text{W}}$ high indicates the transfer is a read from the HPI, while driving UHPI_HR/ $\overline{\text{W}}$ low indicates a write to the HPI.
UHPI_HHWIL	I	Address or control pins	<b>Halfword identification line.</b> The host uses UHPI_HHWIL to identify the first and second halfwords of the host cycle. UHPI_HHWIL must be driven low for the first halfword and high for the second halfword.
$\overline{\text{UHPI\_HAS}}$	I	None	<b>Address strobe.</b> Connect to logic high.
$\overline{\text{UHPI\_HINT}}$	O/Z	Interrupt pin	<b>Host interrupt.</b> The CPU can interrupt the host processor by writing a 1 to the HINT bit of HPIC. Before subsequent $\overline{\text{UHPI\_HINT}}$ interrupts can occur, the host must acknowledge interrupts by writing a 1 to the HINT bit. This pin is active-low (that is, when an interrupt is asserted from the host, the state of this signal is low) and inverted from the HINT bit value in HPIC.
$\overline{\text{UHPI\_HDS1}}$ and $\overline{\text{UHPI\_HDS2}}$	I	Read strobe and write strobe pins or any data strobe pin	<b>HPI data strobe pins.</b> These pins are used for strobing data in and out of the HPI (for data strobing details, see Section 18.2.6.4). The direction of the data transfer depends on the logic level of the UHPI_HR/ $\overline{\text{W}}$ signal. The $\overline{\text{UHPI\_HDS}}$ signals are also used to latch control information on the falling edge. During an HPID write access, data is latched into the HPID register on the rising edge of $\overline{\text{UHPI\_HDS}}$ . During read operations, these pins act as output-enable pins of the host data bus.
UHPI_HD[15:0]	I/O/Z	Data bus	<b>HPI data bus.</b> The HPI data bus carries the address and data to/from the HPI.
$\overline{\text{UHPI\_HRDY}}$	O/Z	Asynchronous ready pin	<b>HPI-ready signal.</b> When the HPI drives $\overline{\text{UHPI\_HRDY}}$ low, the host has permission to complete the current host cycle. When the HPI drives $\overline{\text{UHPI\_HRDY}}$ high, the HPI is not ready for the current host cycle to complete.

## 18.2.4 Pin Multiplexing and General-Purpose I/O Control Blocks

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. See your device-specific data manual to determine how pin multiplexing affects the HPI.

The HPI supports general-purpose I/O (GPIO) capability on all pins. All HPI pins may be enabled for GPIO mode when the HPI is disabled via the HPIENA bit in the chip configuration 1 register (CFGCHIP1) in the *System Configuration (SYSCFG) Module* chapter.

When the HPI is enabled, the pins not being used for host accesses may be configured as general-purpose I/O.

### 18.2.4.1 Treatment of Optional Pins when Configured as General-Purpose I/O

Certain pins are optional, but if used to interface to the external hosts, the pins are inputs to the HPI. For the purpose of host accesses, the HPI treats these pins as if they were driven to the values listed in [Table 18-2](#).

**Table 18-2. Value on Optional Pins when Configured as General-Purpose I/O**

Pin	GPIO Enable Bit(s)	When Enabled as GPIO, Treated as Driven:
UHPI_HD[15:8]	GPIOEN8	0
UHPI_HD[7:0]	GPIOEN7	0
UHPI_HCNTL0	GPIOEN1	1
UHPI_HCNTL1	GPIOEN1	1
UHPI_HAS	GPIOEN2	1

### 18.2.4.2 General-Purpose I/O Programmer's Model

For each HPI pin, there are three bits that control this pin as general-purpose I/O (GPIO):

- Enable: GPIO\_EN.GPIOEN[xx]
- Direction: GPIO\_DIRn.DIR[yy]
- Data: GPIO\_DATn.DIR[yy]

For example, the  $\overline{\text{UHPI\_HAS}}$  pin is enabled with the GPIO\_EN.GPIOEN2 bit. In the default setting, the GPIO\_EN.GPIOEN2 bit is cleared to 0; therefore, the  $\overline{\text{UHPI\_HAS}}$  pin functions as the host address strobe.

Enabling this pin for GPIO, by setting the GPIO\_EN.GPIOEN2 bit to 1 does two things:

- Transfers control of the  $\overline{\text{UHPI\_HAS}}$  pin to GPIO direction and data bits.
- Drives a 1 into the  $\overline{\text{UHPI\_HAS}}$  pin input of the external host interface block (regardless of the actual pin value).

Once enabled as GPIO, the direction of the  $\overline{\text{UHPI\_HAS}}$  pin is controlled by the GPIO\_DIR2.HASZ bit. If this bit is set to 1, the pin will be driven as an output; if this bit is cleared to 0, the pin will be an input.

Once the direction of the  $\overline{\text{UHPI\_HAS}}$  pin is set, the data value is either written to or read from the GPIO\_DAT2.HASZ bit. If the pin was configured as an output, then writing to this bit determines the value to drive out the pin. Reading from this bit will return the value written.

When the GPIO\_DIR2.HASZ bit is cleared to 0, configuring the  $\overline{\text{UHPI\_HAS}}$  pin as an input, writing to the GPIO\_DAT2.HASZ bit has no effect. Reading from this bit will return the value on the  $\overline{\text{UHPI\_HAS}}$  pin.

---

**NOTE:** Note that you cannot preload a value into the DAT field before configuring the pin as an output. This means that when switching the pin from input to output, the pin will initially drive the last value input back out on the pin. Then the DAT bit can be written to change the value on the pin. If the intermediate value between writing to DIR and writing to DAT will cause a problem at the system level, it is suggested to use another general-purpose I/O pin on the device.

---

### 18.2.5 Protocol Description

The HPI does not conform to any industry standard protocol.

### 18.2.6 Operation

#### 18.2.6.1 Using the Address Registers

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). These roles are unchanging from the viewpoint of the HPI logic. The HPI DMA logic gets the address from HPIAR when reading from processor resources (see [Section 18.2.2](#)) and gets the address from HPIAW when writing to processor resources (see [Section 18.2.2](#)).

However, unlike the HPI logic, the host can choose how to interact with the two HPI address registers. Using the DUALHPIA bit in the HPI control register (HPIC), the host determines whether HPIAR and HPIAW act as a single 32-bit register (single-HPIA mode) or as two independent 32-bit registers (dual-HPIA mode).

Note that the addresses loaded into the HPI address registers must be byte addresses, and must be 32-bit word aligned (with the least-significant two bits equal to zero), for use in addressing memory space within the DSP.

##### 18.2.6.1.1 Single-HPIA Mode

When DUALHPIA = 0 in HPIC, HPIAR and HPIAW become a single HPI address register (HPIA) from the perspective of the host. In this mode:

- A host HPIA write cycle (UHPI\_HCNTL[1:0] = 10b, UHPI\_HR/ $\overline{W}$  = 0) updates HPIAR and HPIAW with the same value.
- Both HPI address registers are incremented during autoincrement read/write cycles (UHPI\_HCNTL[1:0] = 01b).
- An HPIA read cycle (UHPI\_HCNTL[1:0] = 10b, UHPI\_HR/ $\overline{W}$  = 1) returns the content of HPIAR, which should be identical to the content of HPIAW.

To maintain consistency between the contents of HPIAR and HPIAW, the host should always reinitialize the HPI address registers after changing the state of the DUALHPIA bit. In addition, when DUALHPIA = 0, the host must always reinitialize the HPI address registers when it changes the data direction (from an HPID read cycle to an HPID write cycle, or conversely). Otherwise, the memory location accessed by the HPI DMA logic might not be the location intended by the host.

### 18.2.6.1.2 Dual-HPIA Mode

When DUALHPIA = 1 in HPIC, HPIAR and HPIAW are two independent HPI address registers from the perspective of the host. In this mode:

- A host HPIA access (UHPI\_HCNTL[1:0] = 10b) reads/updates either HPIAR or HPIAW, depending on the value of the HPIA read/write select (HPIASEL) bit in HPIC. This bit is programmed by the host. While HPIASEL = 1, only HPIAR is read or updated by the host. While HPIASEL = 0, only HPIAW is read or updated by the host. The HPIASEL bit is only meaningful in the dual-HPIA mode.

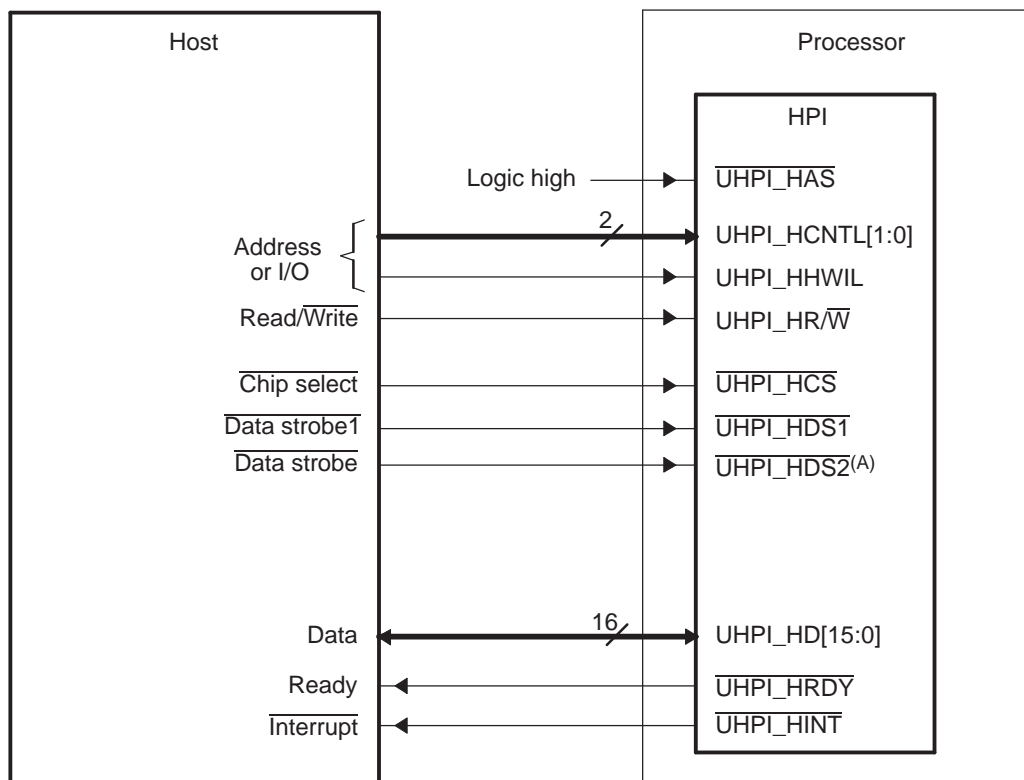
**NOTE:** The HPIASEL bit does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAR when reading from memory and HPIAW when writing to memory.

- A host HPID access with autoincrementing (UHPI\_HCNTL[1:0] = 01b) causes only the relevant HPIA value to be incremented to the next consecutive memory address. In an autoincrement read cycle, HPIAR is incremented after it has been used to perform the current read from memory. In an autoincrement write cycle, HPIAW is incremented after it has been used for the write operation.

### 18.2.6.2 Host-HPI Signal Connections

Figure 18-2 shows an example of a signal connection between the HPI and a host.

Figure 18-2. Example of Host-Processor Signal Connections



A Data strobing options are given in [Section 18.2.6.4](#)



### 18.2.6.3 HPI Configuration and Data Flow

The host accomplishes a multiplexed access in the following manner:

1. The host writes to the HPI control register (HPIC) to properly configure the HPI. Typically, this means programming the halfword order bit (HWOB) and the HPIA-related bits (DUALHPIA and HPIASEL). This step is normally performed once before the initial data access.
2. The host writes the desired internal processor memory address to an address register (HPIAR and/or HPIAW). For an introduction to the two HPI address registers and the two ways the host can interact with them, see [Section 18.2.6.1](#).
3. The host reads from or writes to the data register (HPID). Data transfers between HPID and the internal memory of the processor are handled by the HPI DMA logic and are transparent to the CPU.

Each step of the access uses the same bus. Therefore, the host must drive the appropriate levels on the UHPI\_HCNTL1 and UHPI\_HCNTL0 signals to indicate which register is to be accessed. The host must also drive the appropriate level on the UHPI\_HR $\overline{W}$  signal to indicate the data direction (read or write) and must drive other control signals as appropriate. When HPI resources are temporarily busy or unavailable, the HPI can communicate this to the host by deasserting the HPI-ready (UHPI\_HRDY) output signal.

When performing an access, the HPI first latches the levels on UHPI\_HCNTL[1:0], UHPI\_HR $\overline{W}$ , and other control signals. This latching can occur on the falling edge of the internal strobe signal (for details, see [Section 18.2.6.4](#)). After the control information is latched, the HPI initiates an access based on the control signals.

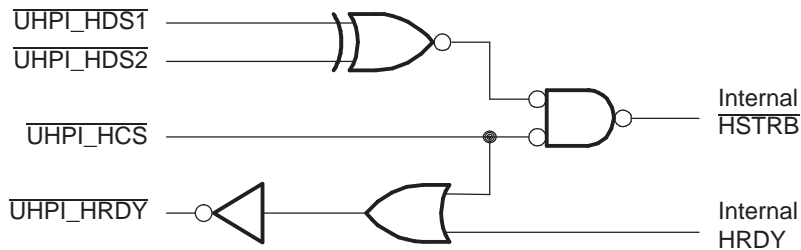
If the host wants to read data from processor resources (see [Section 18.2.2](#)), the HPI DMA logic reads the resource address from HPIAR and retrieves the data from the addressed memory location. When the data has been placed in HPID, the HPI drives the data onto its UHPI\_HD bus. The UHPI\_HRDY signal informs the host whether the data on the UHPI\_HD bus is valid (UHPI\_HRDY low) or not valid yet (UHPI\_HRDY high). When the data is valid, the host should latch the data and drive the connected data strobe (UHPI\_HDS1 or UHPI\_HDS2) inactive, which, in turn, will cause the internal strobe (internal HSTRB) signal to transition from low to high.

If the host wants to write data to processor resources (see [Section 18.2.2](#)), the operation is similar. After the host determines that the HPI is ready to latch the data (UHPI\_HRDY is low), it must cause internal HSTRB to transition from low to high, which causes the data to be latched into HPID. Once the data is in HPID, the HPI DMA logic reads the memory address from HPIAW and transfers the data from HPID to the addressed memory location.

**18.2.6.4 UHPI\_HDS2, UHPI\_HDS1, and UHPI\_HCS: Data Strobing and Chip Selection**

As shown in Figure 18-3, the strobing logic is a function of three key inputs: the chip select pin (UHPI\_HCS) and two data strobe signals (UHPI\_HDS1 and UHPI\_HDS2). The internal strobe signal, which is referred to as internal HSTRB throughout this document, functions as the actual strobe signal inside the HPI. UHPI\_HCS must be low (HPI selected) during strobe activity on the UHPI\_HDS pins. If UHPI\_HCS remains high (HPI not selected), activity on the UHPI\_HDS pins is ignored.

**Figure 18-3. HPI Strobe and Select Logic**



Strobe connections between the host and the HPI depend in part on the number and types of strobe pins available on the host. Table 18-3 describes some options for connecting to the UHPI\_HDS pins.

Notice in Figure 18-3 that UHPI\_HRDY is also gated by UHPI\_HCS. If UHPI\_HCS goes high (HPI not selected), UHPI\_HRDY goes low, regardless of whether the current internal transfer is completed in the processor.

**NOTE:** The UHPI\_HCS input and one UHPI\_HDS strobe input can be tied together and driven with a single strobe signal from the host. This technique selects the HPI and provides the strobe, simultaneously. When using this method, be aware that UHPI\_HRDY is gated by UHPI\_HCS as previously described.

It is not recommended to tie both UHPI\_HDS1 and UHPI\_HDS2 to static logic levels and use UHPI\_HCS as a strobe.

**Table 18-3. Options for Connecting Host and HPI Data Strobe Pins**

Available Host Data Strobe Pins	Connections to HPI Data Strobe Pins
Host has separate read and write strobe pins, both active-low	Connect one strobe pin to UHPI_HDS1 and the other to UHPI_HDS2 <sup>(1)</sup> . Since such a host might not provide a R/W line, take care to satisfy UHPI_HR/W timings as stated in your device-specific data manual. This could possibly be done using a host address line.
Host has separate read and write strobe pins, both active-high	Connect one strobe pin to UHPI_HDS1 and the other to UHPI_HDS2 <sup>(1)</sup> . Since such a host might not provide a R/W line, take care to satisfy UHPI_HR/W timings as stated in your device-specific data manual. This could possibly be done using a host address line.
Host has one active-low strobe pin	Connect the strobe pin to UHPI_HDS1 or UHPI_HDS2, and connect the other pin to logic-level 1.
Host has one active-high strobe pin	Connect the strobe pin to UHPI_HDS1 or UHPI_HDS2, and connect the other strobe pin to logic-level 0.

<sup>(1)</sup> The UHPI\_HR/W signal could be driven by a host address line in this case.

### 18.2.6.5 UHPI\_HCNTL[1:0] and UHPI\_HR/W: Indicating the Cycle Type

The cycle type consists of:

- The access type that the host selects by driving the appropriate levels on the UHPI\_HCNTL[1:0] pins of the HPI. [Table 18-4](#) describes the four available access types.
- The transfer direction that the host selects with the UHPI\_HR/W pin. The host must drive the UHPI\_HR/W signal high (read) or low (write).

A summary of cycle types is in [Table 18-5](#). The HPI samples the UHPI\_HCNTL levels at the falling edge of the internal strobe signal  $\overline{\text{HSTRB}}$ .

**Table 18-4. Access Types Selectable With the UHPI\_HCNTL Signals**

UHPI_HCNTL1	UHPI_HCNTL0	Access Type
0	0	<b>HPIC access.</b> The host requests to access the HPI control register (HPIC).
0	1	<b>HPID access with autoincrementing.</b> The host requests to access the HPI data register (HPID) and to have the appropriate HPI address register (HPIAR and/or HPIAW) automatically incremented by 1 after the access.
1	0	<b>HPIA access.</b> The host requests to access the appropriate HPI address register (HPIAR and/or HPIAW).
1	1	<b>HPID access without autoincrementing.</b> The host requests to access the HPI data register (HPID) but requests no automatic post-increment of the HPI address register.

**Table 18-5. Cycle Types Selectable With the UHPI\_HCNTL and UHPI\_HR/W Signals**

UHPI_HCNTL1	UHPI_HCNTL0	UHPI_HR/W	Cycle Type
0	0	0	HPIC write cycle
0	0	1	HPIC read cycle
0	1	0	HPID write cycle with autoincrementing
0	1	1	HPID read cycle with autoincrementing
1	0	0	HPIA write cycle
1	0	1	HPIA read cycle
1	1	0	HPID write cycle without autoincrementing
1	1	1	HPID read cycle without autoincrementing

### 18.2.6.6 UHPI\_HHWIL: Identifying the First and Second Halfwords in Multiplexed Mode Transfers

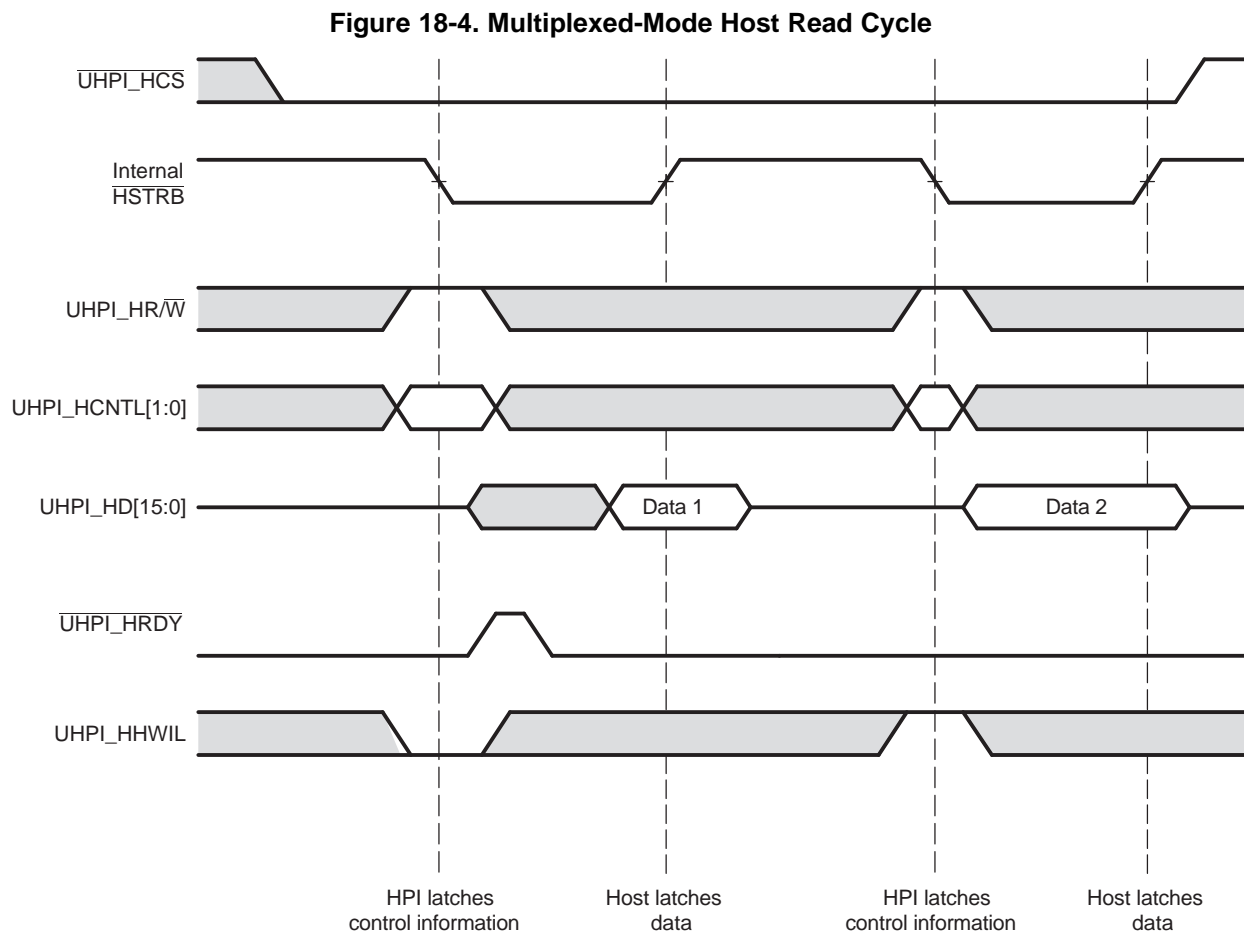
Each host cycle consists of two consecutive halfword transfers. For each transfer, the host must specify the cycle type with UHPI\_HCNTL[1:0] and UHPI\_HR/W, and the host must use UHPI\_HHWIL to indicate whether the first or second halfword is being transferred. For HPID and HPIA accesses, UHPI\_HHWIL must always be driven low for the first halfword transfer and high for the second halfword transfer. Results are undefined if the sequence is broken. For examples of using UHPI\_HHWIL, see [Section 18.2.6.7](#).

When the host sends the two halfwords of a 32-bit word in this manner, the host can send the most-significant and the least-significant halfwords of the word in either order (most-significant halfword first or most-significant halfword second). However, the host must inform the HPI of the selected order before beginning the host cycle. This is done by programming the halfword order (HWOB) bit in HPIC. Although HWOB is written at bit 0 in HPIC, its current value is readable at both bit 0 and bit 8 (HWOBSTAT). Thus, the host can determine the current halfword order configuration by checking the least-significant bit of either half of HPIC.

There is one case when the HPI does not require a dual halfword access with UHPI\_HHWIL low for the first halfword and UHPI\_HHWIL high for the second halfword. This is the case when accessing the HPIC register. When accessing HPIC, the state of UHPI\_HHWIL is ignored and the same 16-bit HPIC register is accessed regardless of whether the host performs a single or dual access. For an example timing diagram of this case, see [Section 18.2.6.8](#).

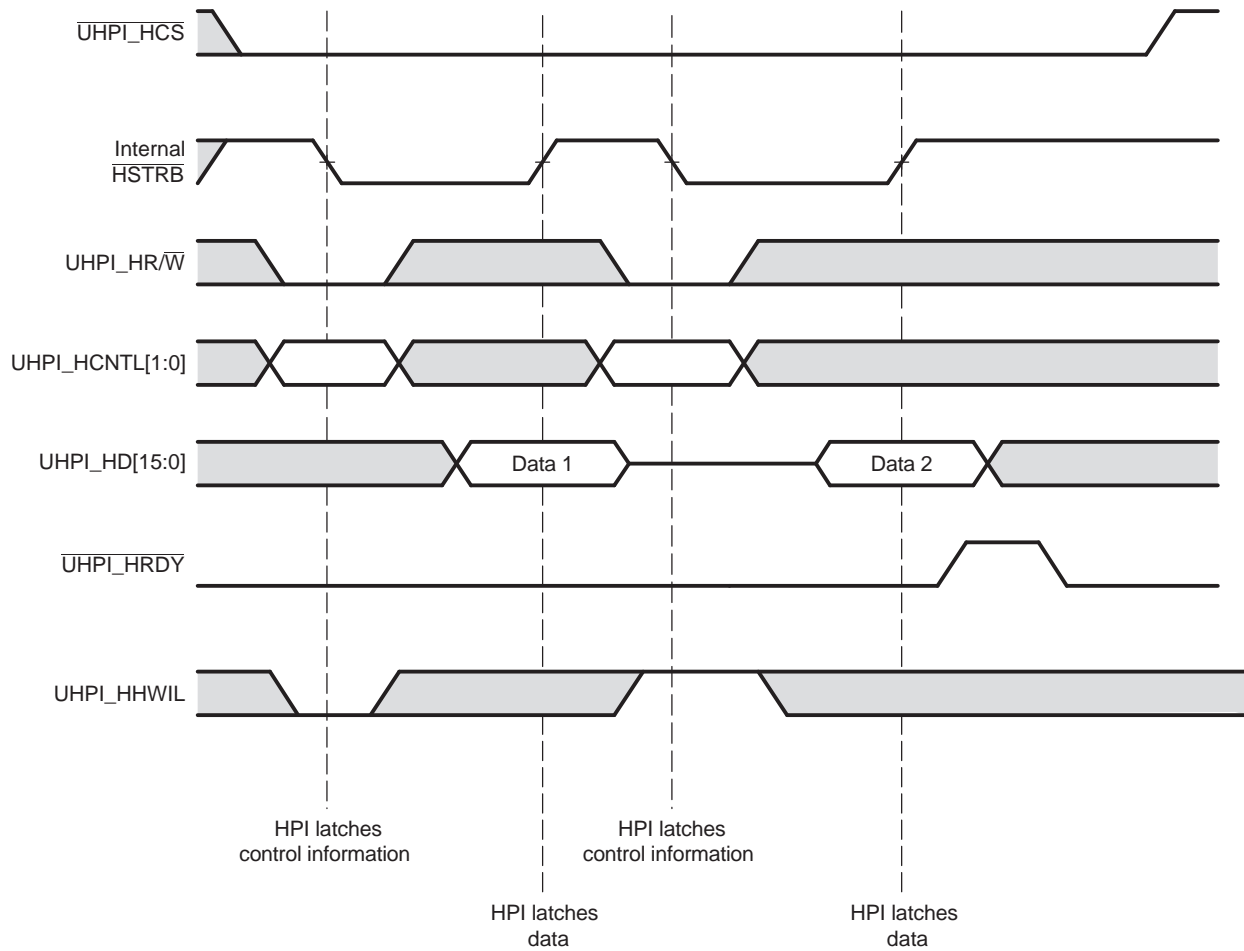
### 18.2.6.7 Performing a Multiplexed Access

Figure 18-2 shows an example of signal connections for multiplexed transfers. Figure 18-4 and Figure 18-5 show typical HPI signal activity when performing a read and write transfer, respectively. In these cases, the falling edge of internal  $\overline{\text{HSTRB}}$  is used to latch the UHPI\_HCNTL[1:0], UHPI\_HR/W, and UHPI\_HHWIL states into the HPI. Internal  $\overline{\text{HSTRB}}$  is derived from UHPI\_HCS, UHPI\_HDS1, and UHPI\_HDS2 as described in [Section 18.2.6.4](#).



NOTE: Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on UHPI\_HRDY may or may not occur. For more information, see [Section 18.2.6.9](#).

Figure 18-5. Multiplexed-Mode Host Write Cycle

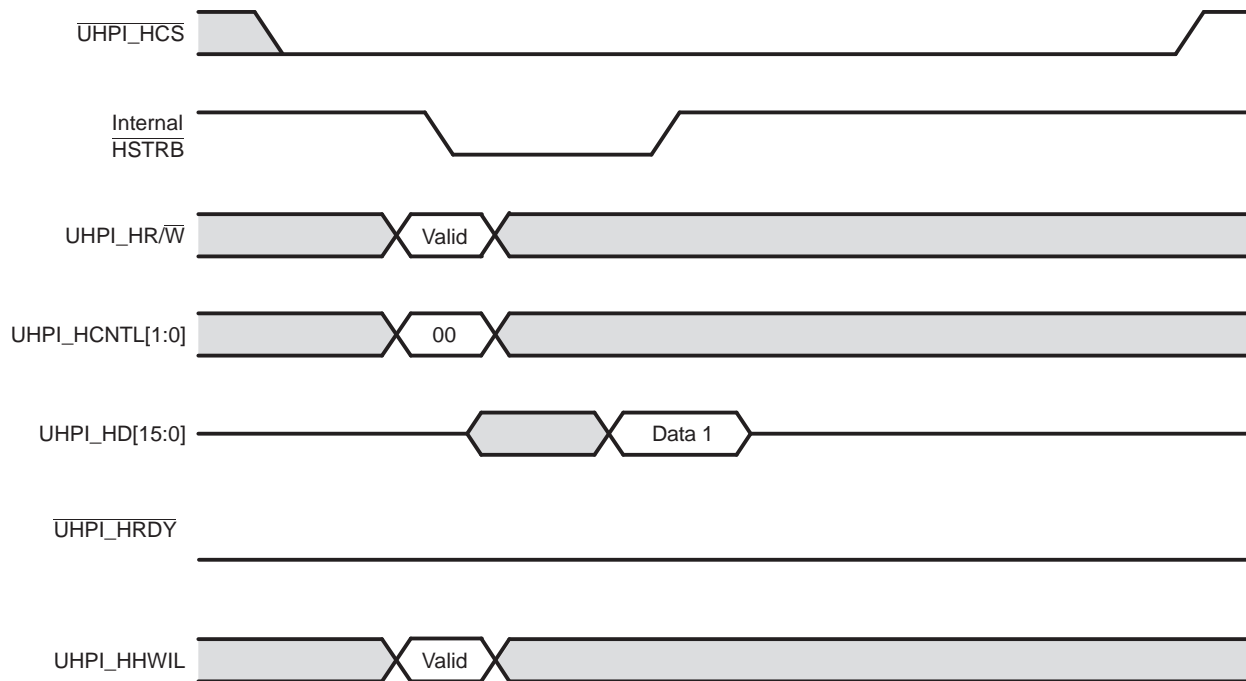


NOTE: Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on  $\overline{\text{UHPI\_HRDY}}$  may or may not occur. For more information, see [Section 18.2.6.9](#).

### 18.2.6.8 Single-Halfword HPIC Cycle

Figure 18-6 shows the special case (see Section 18.2.6.6) when the host performs a single-halfword cycle to access the HPIC. The state of UHPI\_HHWIL is ignored and if a dual-halfword access is performed, then the same HPIC register is accessed twice.

Figure 18-6. Multiplexed-Mode Single-Halfword HPIC Cycle (Read or Write)



### 18.2.6.9 Hardware Handshaking Using the HPI-Ready (UHPI\_HRDY) Signal

The HPI uses its ready signal,  $\overline{\text{UHPI\_HRDY}}$ , to tell the host whether it is ready to complete an access. During a read cycle, the HPI is ready ( $\overline{\text{UHPI\_HRDY}}$  is low) when it has data available for the host. During a write cycle, the HPI is ready ( $\overline{\text{UHPI\_HRDY}}$  is low) when it is ready to latch data from the host. If the HPI is not ready, it can drive  $\overline{\text{UHPI\_HRDY}}$  high to insert wait states. These wait states indicate to the host that read data is not yet valid (read cycle) or that the HPI is not ready to latch write data (write cycle). The number of wait states that must be inserted by the HPI is dependent upon the state of the resource that is being accessed.

When the HPI is not ready to complete the current cycle ( $\overline{\text{UHPI\_HRDY}}$  is high), the host can begin a new host cycle by forcing the HPI to latch new control information. However, once the cycle has been initiated, the host must wait until  $\overline{\text{UHPI\_HRDY}}$  goes low before causing a rising edge on the internal strobe signal (internal  $\overline{\text{HSTRB}}$ ) to complete the cycle. If internal  $\overline{\text{HSTRB}}$  goes high when the HPI is not ready, the cycle will be terminated with invalid data being returned (read cycle) or written (write cycle).

One reason the HPI may drive  $\overline{\text{UHPI\_HRDY}}$  high is a not-ready condition in one of its first-in, first-out buffers (FIFOs). For example, any HPID access that occurs while the write FIFO is full or the read FIFO is empty may result in some number of wait states being inserted by the HPI. The FIFOs are explained in Section 18.2.6.10.

The following sections describe the behavior of  $\overline{\text{UHPI\_HRDY}}$  during HPI register accesses. In all cases, the chip select signal,  $\overline{\text{UHPI\_HCS}}$ , must be asserted for  $\overline{\text{UHPI\_HRDY}}$  to go low.

### 18.2.6.9.1 $\overline{UHPI\_HRDY}$ Behavior During Multiplexed-Mode Read Operations

Figure 18-7 shows an HPIC ( $UHPI\_HCNTL[1:0] = 00b$ ) or HPIA ( $UHPI\_HCNTL[1:0] = 10b$ ) read cycle. Neither an HPIC read cycle nor an HPIA read cycle causes  $\overline{UHPI\_HRDY}$  to go high. For this type of access, the state of  $UHPI\_HHWIL$  is ignored, so if a dual halfword access is performed, the same register will be accessed twice.

**Figure 18-7.  $\overline{UHPI\_HRDY}$  Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode**

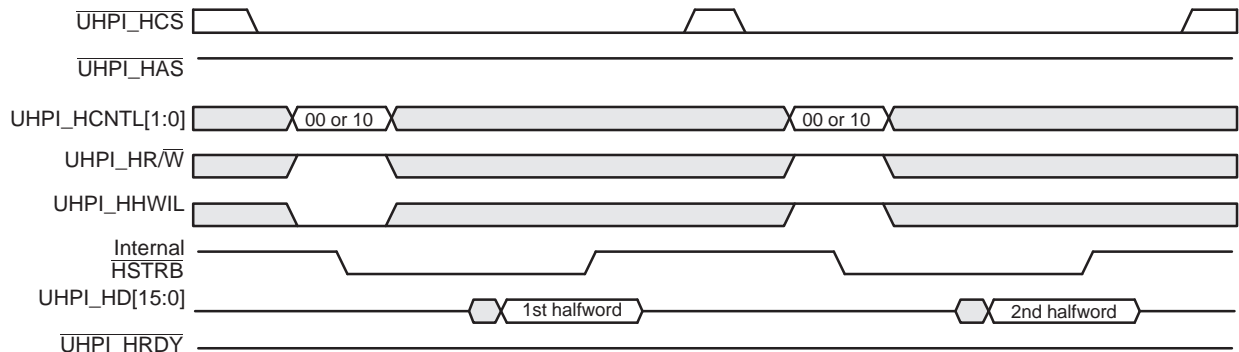


Figure 18-8 includes an HPID read cycle without autoincrementing. The host writes the memory address during the HPIA ( $UHPI\_HCNTL[1:0] = 10b$ ) write cycle, and the host reads the data during the HPID ( $UHPI\_HCNTL[1:0] = 11b$ ) read cycle.  $\overline{UHPI\_HRDY}$  goes high for each HPIA halfword access, but  $\overline{UHPI\_HRDY}$  goes high for only the first halfword access in each HPID read cycle.

**Figure 18-8.  $\overline{UHPI\_HRDY}$  Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle)**

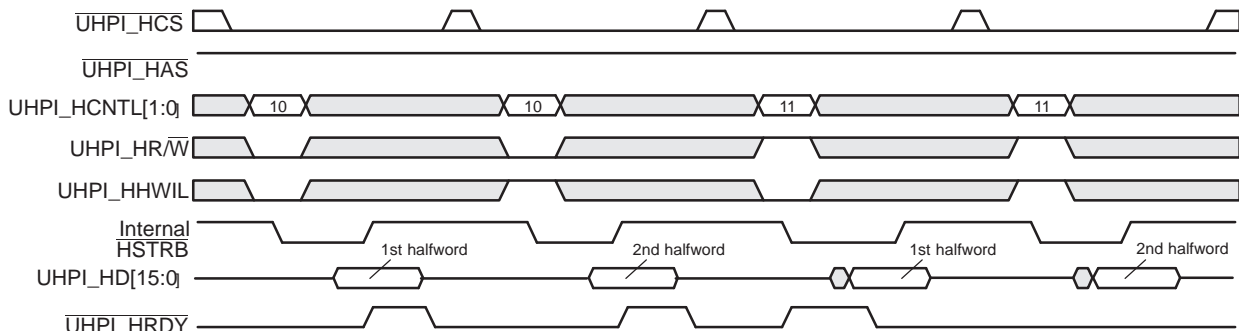
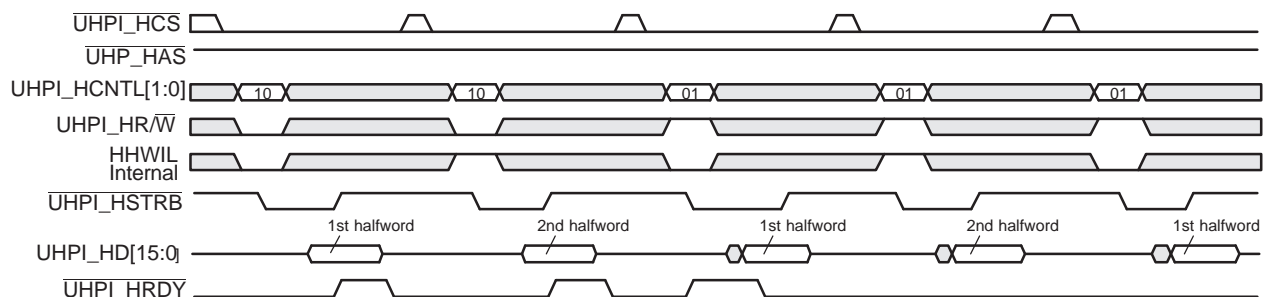


Figure 18-9 includes an autoincrement HPID read cycle. The host writes the memory address while asserting  $UHPI\_HCNTL[1:0] = 10b$  and reads the data while asserting  $UHPI\_HCNTL[1:0] = 01b$ . During the first HPID read cycle,  $\overline{UHPI\_HRDY}$  goes high for only the first halfword access, and subsequent HPID read cycles do not cause  $\overline{UHPI\_HRDY}$  to go high.

**Figure 18-9.  $\overline{UHPI\_HRDY}$  Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles)**



**18.2.6.9.2 UHPI\_HRDY Behavior During Multiplexed-Mode Write Operations**

Figure 18-10 shows an HPIC (UHPI\_HCNTL[1:0] = 00b) write cycle operation. An HPIC write cycle does not cause UHPI\_HRDY to go high and the state of UHPI\_HHWIL is ignored. Firmware is not required to perform a dual access to access HPIC.

**Figure 18-10. UHPI\_HRDY Behavior During an HPIC Write Cycle in the Multiplexed Mode**

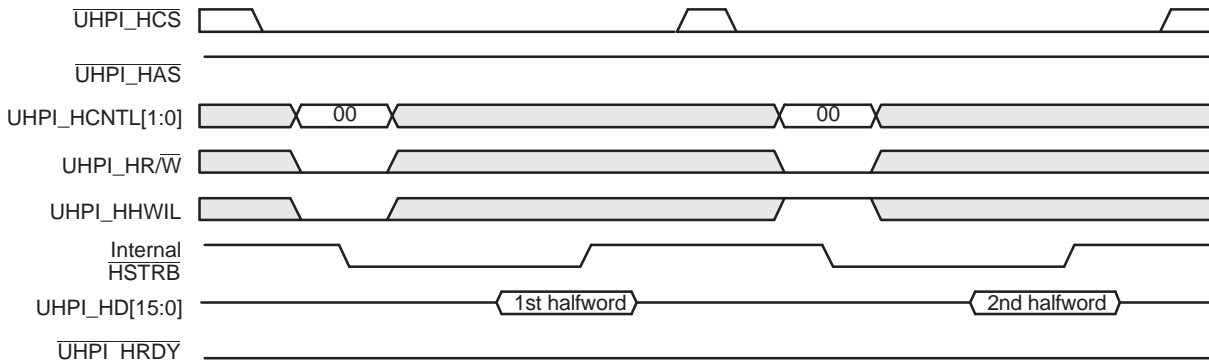


Figure 18-11 includes a HPID write cycle without autoincrementing. The host writes the memory address while UHPI\_HCNTL[1:0] = 10b and writes the data while UHPI\_HCNTL[1:0] = 11b. During the HPID write cycle, UHPI\_HRDY goes high only for the second halfword access.

**Figure 18-11. UHPI\_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing)**

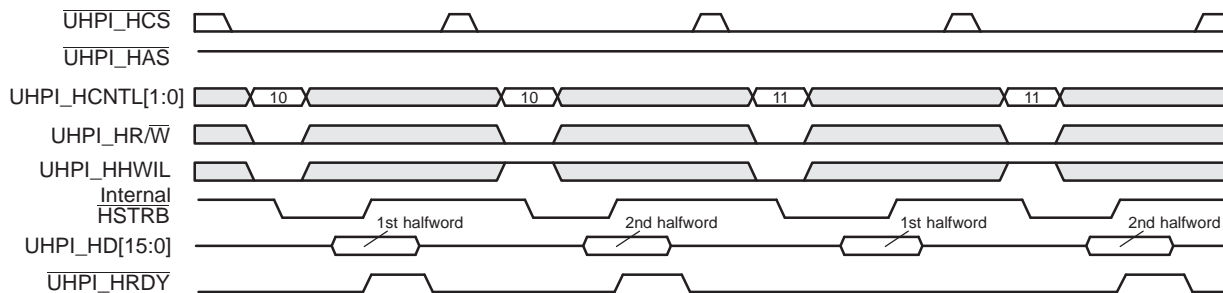


Figure 18-12 shows autoincrement HPID write cycles when the write FIFO is empty prior to the HPIA write. The host writes the memory address while UHPI\_HCNTL[1:0] = 10b and writes the data while UHPI\_HCNTL[1:0] = 01b. UHPI\_HRDY does not go high during any of the HPID write cycles until the FIFO is full.

**Figure 18-12. UHPI\_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write)**

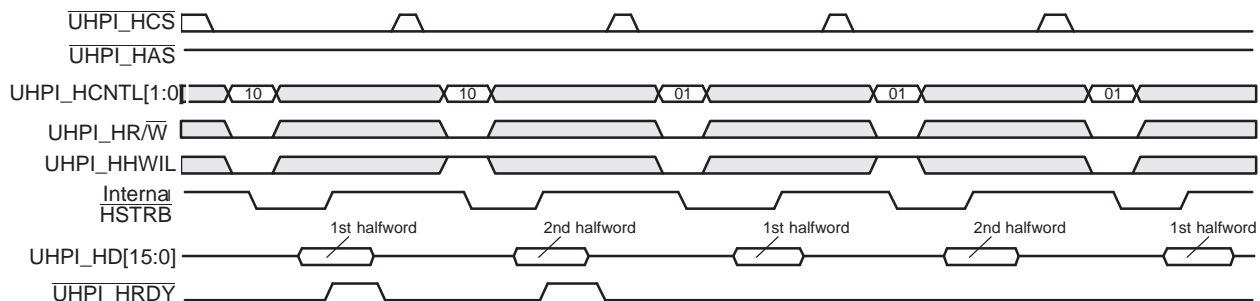
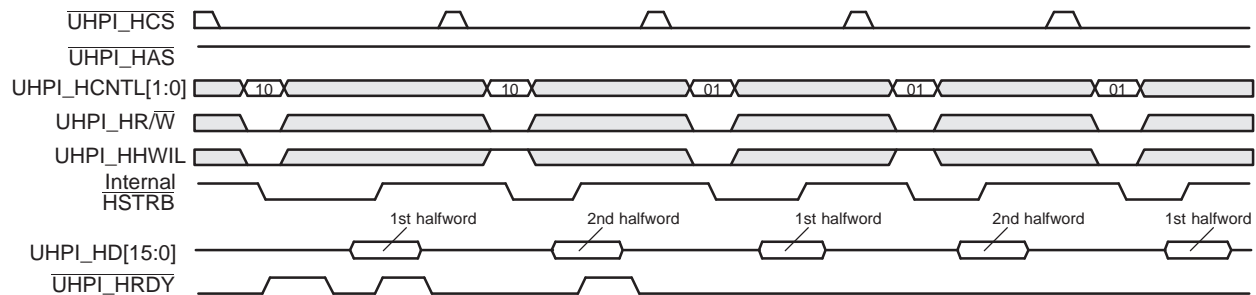




Figure 18-13 shows a case similar to that of Figure 18-12. However, in the case of Figure 18-13, the write FIFO is not empty when the HPIA access is made.  $\overline{\text{UHPI\_HRDY}}$  goes high twice for the first halfword access of the HPIA write cycle. The first  $\overline{\text{UHPI\_HRDY}}$  high period is due to the nonempty FIFO. The data currently in the FIFO must first be written to the memory. This results in  $\overline{\text{UHPI\_HRDY}}$  going high immediately after the falling edge of the data strobe ( $\overline{\text{HSTRB}}$ ). The second and third  $\overline{\text{UHPI\_HRDY}}$  high periods occur for the writes to the HPIA.  $\overline{\text{UHPI\_HRDY}}$  remains low for the HPID accesses.

**Figure 18-13.  $\overline{\text{UHPI\_HRDY}}$  Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write)**

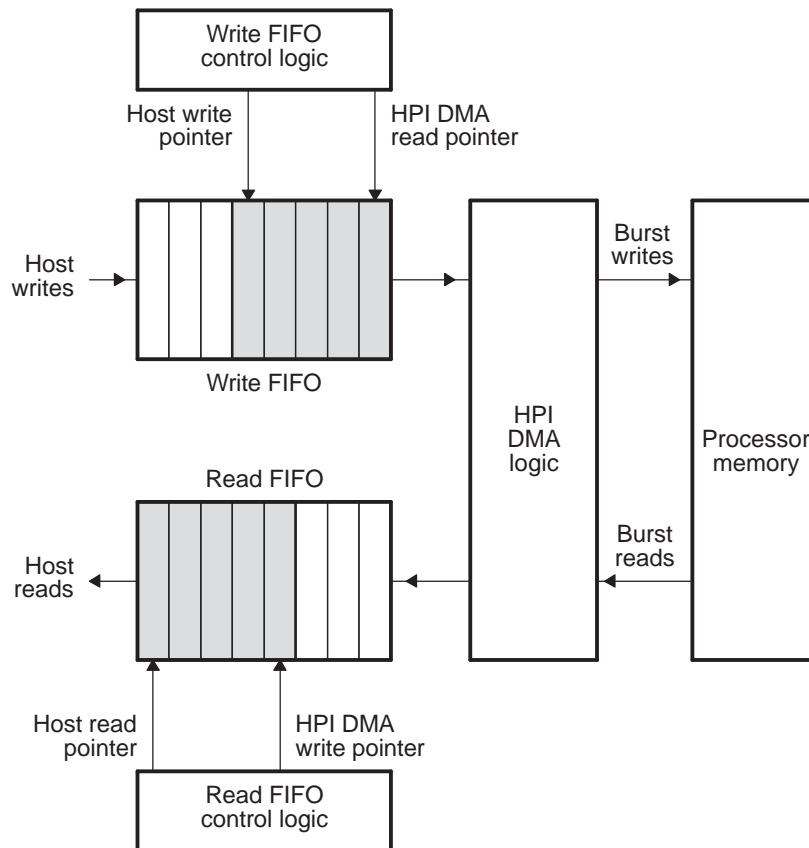


### 18.2.6.10 FIFOs and Bursting

The HPI data register (HPID) is a port through which the host accesses two first-in, first-out buffers (FIFOs). As shown in Figure 18-14, a read FIFO supports host read cycles, and a write FIFO supports host write cycles. Both read and write FIFOs are 8-words deep (each word is 32 bits). If the host is performing multiple reads or writes to consecutive memory addresses (autoincrement HPID cycles), the FIFOs are used for bursting. The HPI DMA logic reads or writes a burst of four words at a time when accessing one of the FIFOs.

Bursting is essentially invisible to the host because the host interface signaling is not affected. Its benefit to the host is that the  $\overline{\text{UHPI\_HRDY}}$  signal is deasserted less often when there are multiple reads or writes to consecutive addresses.

Figure 18-14. FIFOs in the HPI



### 18.2.6.10.1 Read Bursting

When the host writes to the read address register (HPIAR), the read FIFO is flushed. Any host read data that was in the read FIFO is discarded (the read FIFO pointers are reset). If an HPI DMA write to the read FIFO is in progress at the time of a flush request, the HPI allows this write to complete and then performs the flush.

Read bursting can begin in one of two ways: the host initiates an HPID read cycle with autoincrementing, or the host initiates issues a FETCH command (writes 1 to the FETCH bit in HPIC).

If the host initiates an HPID read cycle with autoincrementing, the HPI DMA logic performs two 4-word burst operations to fill the read FIFO. The host is initially held off by the deassertion of the  $\overline{\text{UHPI\_HRDY}}$  signal until data is available to be read from the read FIFO. Once data is available in the read FIFO, the host can read data from the read FIFO by performing subsequent reads of HPID with autoincrementing. Once the initial read has been performed, the HPI DMA logic continues to perform 4-word burst operations to consecutive memory addresses every time there are four empty word locations in the read FIFO. The HPI DMA logic continues to prefetch data to keep the read FIFO full, until the occurrence of an event that causes a read FIFO flush (see [Section 18.2.6.10.3](#)).

As mentioned, the second way that read bursting may begin is with a FETCH command. The host should always precede the FETCH command with the initialization of the HPIAR register or a nonautoincrement access, so that the read FIFO is flushed beforehand. When the host initiates a FETCH command, the HPI DMA logic begins to prefetch data to keep the read FIFO full, as described in the previous paragraph. The FETCH bit in HPIC does not actually store the value that is written to it; rather, the decoding of a host write of 1 to this bit is considered a FETCH command.

The FETCH command can be helpful if the host wants to minimize a stall condition on the interface. The host can initiate prefetching by writing 1 to the FETCH bit and later perform a read. The host can make use of the time it takes to load the read FIFO with read data, during which the HPI was not ready, by using the CPU to service other tasks.

Both types of continuous or burst reads described in the previous paragraphs begin with a write to the HPI address register, which causes a read FIFO flush. This is the typical way of initiating read cycles, because the initial read address needs to be specified.

---

**NOTE:** An HPID read cycle without autoincrementing does not initiate any prefetching activity. Instead, it causes the read FIFO to be flushed and causes the HPI DMA logic to perform a single-word read from the processor memory. As soon as the host activates a read cycle without autoincrementing, prefetching activity ceases until the occurrence of a FETCH command or an autoincrement read cycle.

---

### 18.2.6.10.2 Write Bursting

A write to the write address register (HPIAW) causes the write FIFO to be flushed. This means that any write data in the write FIFO is forced to its destination in the processor memory (the HPI DMA logic performs burst operations until the write FIFO is empty). When the FIFO has been flushed, the only action that will cause the HPI DMA logic to perform burst writes is a host write to HPID with autoincrementing. The initial host-write data is stored in the write FIFO. An HPI DMA write is not requested until there are four words in the write FIFO. As soon as four words have been written to the FIFO via HPID write cycles with autoincrementing, the HPI DMA logic performs a 4-word burst operation to the processor memory. The burst operations continue as long as there are at least four words in the FIFO. If the FIFO becomes full (eight words are waiting in the FIFO), the HPI holds off the host by deasserting  $\overline{\text{UHPI\_HRDY}}$  until at least one empty word location is available in the FIFO.

Because excessive time might pass between consecutive burst operations, the HPI has a time-out counter. If there are fewer than four words in the write FIFO and the time-out counter expires, the HPI DMA logic empties the FIFO immediately by performing a 2-word or 3-word burst, or a single-word write, as necessary. Every time new data is written to the write FIFO, the time-out counter is automatically reset to begin its count again. The time-out period is set to a value of 160. For more detailed information about the time-out period, see your device-specific data manual.

---

**NOTE:** An HPID write cycle without autoincrementing does not initiate any bursting activity. Instead, it causes the write FIFO to be flushed and causes the HPI DMA logic to perform a single-word write to the processor memory. As soon as the host activates a write cycle without autoincrementing, bursting activity ceases until the occurrence of an autoincrement write cycle. A nonautoincrement write cycle always should be preceded by the initialization of HPIAW or by another nonautoincrement access, so that the write FIFO is flushed beforehand.

---

### 18.2.6.10.3 FIFO Flush Conditions

When specific conditions occur within the HPI, the read or write FIFO must be flushed to prevent the reading of stale data from the FIFOs. When a read FIFO flush condition occurs, all current host accesses and direct memory accesses (DMAs) to the read FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. The read FIFO pointers are then reset, causing any read data to be discarded.

Similarly, when a write FIFO flush condition occurs, all current host accesses and DMAs to the write FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. All posted writes in the FIFO are then forced to completion with a final burst or single-word write, as necessary.

If the host initiates an HPID host cycle during a FIFO flush, the cycle is held off with the deassertion of `UHPI_HRDY` until the flush is complete and the FIFO is ready to be accessed.

The following conditions cause the read and write FIFOs to be flushed:

- Read FIFO flush conditions:
  - A value from the host is written to the read address register (HPIAR).
  - The host performs an HPID read cycle without autoincrementing.
- Write FIFO flush conditions:
  - A value from the host is written to the write address register (HPIAW).
  - The host performs an HPID write cycle without autoincrementing.
  - The write-burst time-out counter expires.

When operating with `DUALHPIA = 0`, any read or write flush condition causes both read and write FIFOs to be flushed. In addition, the following scenarios cause both FIFOs to be flushed when `DUALHPIA = 0`:

- The host performs a write to the HPIA register.
- The host performs an HPID write cycle with autoincrementing while the read FIFO is not empty (the read FIFO still contains data from prefetching or an HPID read cycle with autoincrementing).
- The host performs an HPID read cycle with autoincrementing while the write FIFO is not empty (there is still posted write data in the write FIFO).

This is useful in providing protection against reading stale data by reading a memory address when a previous write cycle has not been completed at the same address. Similarly, this protects against overwriting data at a memory address when a previous read cycle has not been completed at the same address.

When operating with `DUALHPIA = 1` (HPIAR and HPIAW are independent), there is no such protection. However, when `DUALHPIA = 1`, data flow can occur in both directions without flushing both FIFOs simultaneously, thereby improving HPI bandwidth.

### 18.2.6.10.4 FIFO Behavior When a Hardware Reset or Software Reset Occurs

A hardware reset (RESET pin driven low) or an HPI software reset causes the FIFOs to be reset. The FIFO pointers are cleared, so that all data in the FIFOs are discarded. In addition, all associated FIFO logic is reset.

If a host cycle is active when a hardware or HPI software reset occurs, the `UHPI_HRDY` signal is asserted (driven low), allowing the host to complete the cycle. When the cycle is complete, `UHPI_HRDY` is deasserted (driven high). Any access interrupted by a reset may result in corrupted read data or a lost write data (if the write does not actually update the intended memory or register). Although data may be lost, the host interface protocol is not violated. While either of reset condition is true, and the host is idle (internal `HSTRB` is held high), the FIFOs are held in reset, and host transactions are held off with an inactive `UHPI_HRDY` signal.

## 18.2.7 Reset Considerations

The HPI has two reset sources: software reset and hardware reset.

### 18.2.7.1 Software Reset Considerations

The HPI is not affected by a software reset issued by the emulator.

### 18.2.7.2 Hardware Reset Considerations

When the entire processor is reset with the RESET pin:

- If the internal strobe signal, internal  $\overline{\text{HSTRB}}$ , is high (host is inactive),  $\overline{\text{UHPI\_HRDY}}$  is driven low and remains low until the reset condition is over.
- If internal  $\overline{\text{HSTRB}}$  is low (host cycle is active),  $\overline{\text{UHPI\_HRDY}}$  is driven high, allowing the host to complete the cycle. When internal  $\overline{\text{HSTRB}}$  goes high (cycle is complete),  $\overline{\text{UHPI\_HRDY}}$  is driven low and remains low until the reset condition is over. If the active cycle was a write cycle, the memory or register may not have been correctly updated. If the active cycle was a read cycle, the fetched value may not be valid.
- The HPI registers are reset to their default values (see [Section 18.3](#)).
- The read and write FIFOs and the associated FIFO logic are reset (this includes a flush of the FIFOs).
- Host-to-CPU and CPU-to-host interrupts are cleared.

## 18.2.8 Initialization

The following steps are required to configure the HPI after a hardware reset:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Configure the HPIENA and HPIBYTEAD bits in the chip configuration 1 register (CFGCHIP1) in the *System Configuration (SYSCFG) Module* chapter.
3. Choose how HPIAR and HPIAW will be controlled by configuring the DUALHPIA bit in HPIC.
4. Choose how halfword ordering will be handled by configuring the HWOB bit in HPIC.
5. Choose how the HPI will respond to emulation suspend events by configuring the FREE and SOFT bits in PWREMU\_MGMT.
6. Choose the desired initial addresses and write the addresses to HPIAW and HPIAR, appropriately.
7. Release the HPI logic from reset by clearing the HPIRST bit in HPIC.

The HPI is now ready to perform data transactions.

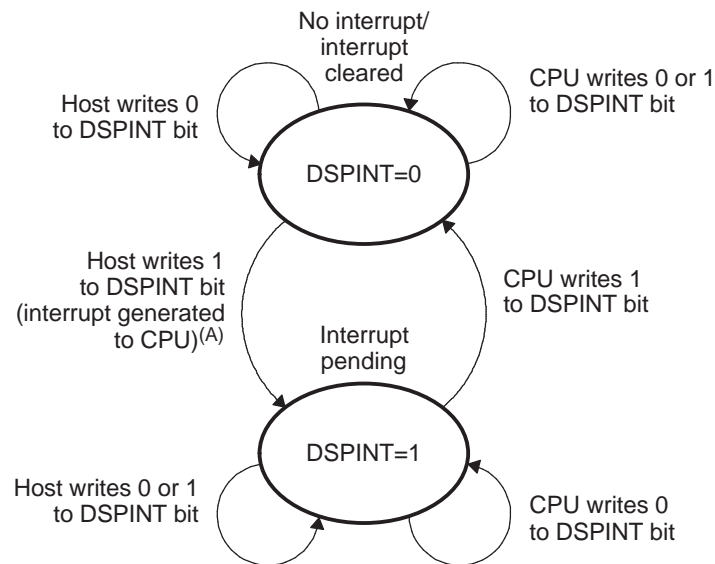
## 18.2.9 Interrupt Support

The host can interrupt the CPU via the DSPINT bit in HPIC, as described in [Section 18.2.9.1](#). The CPU can send an interrupt to the host by using the HINT bit in HPIC, as described in [Section 18.2.9.2](#).

### 18.2.9.1 DSPINT Bit: Host-to-CPU Interrupts

The DSPINT bit in HPIC allows the host to send an interrupt request to the CPU. The use of the DSPINT bit is summarized in [Figure 18-15](#).

**Figure 18-15. Host-to-CPU Interrupt State Diagram**



- A When the DSPINT bit transitions from 0 to 1, an interrupt is generated to the CPU. No new interrupt can be generated until the CPU has cleared the bit (DSPINT = 0).

To interrupt the CPU, the host must:

1. Drive both UHPI\_HCNTL1 and UHPI\_HCNTL0 low to request a write to HPIC.
2. Write 1 to the DSPINT bit in HPIC.

When the host sets the DSPINT bit, the HPI generates an interrupt pulse to the CPU. If this maskable interrupt is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (ISR).

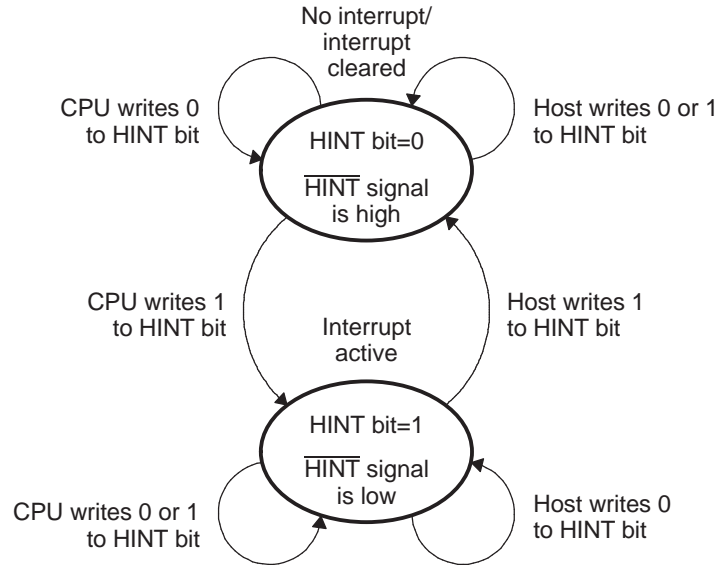
Before the host can use DSPINT to generate a subsequent interrupt to the CPU, the CPU must acknowledge the current interrupt by writing a 1 to the DSPINT bit. When the CPU writes 1, DSPINT is forced to 0. The host should verify that DSPINT = 0 before generating subsequent interrupts. While DSPINT = 1, host writes to the DSPINT bit do not generate an interrupt pulse.

Writes of 0 have no effect. A hardware reset immediately clears DSPINT and thus clears an active host-to-CPU interrupt.

### 18.2.9.2 HINT Bit: CPU-to-Host Interrupts

The HINT bit in HPIC allows the CPU to send an interrupt request to the host. The use of the HINT bit is summarized in [Figure 18-16](#).

**Figure 18-16. CPU-to-Host Interrupt State Diagram**



If the CPU writes 1 to the HINT bit of HPIC, the HPI drives the  $\overline{\text{UHPI\_HINT}}$  signal low, indicating an interrupt condition to the host. Before the CPU can use the HINT bit to generate a subsequent interrupt to the host, the host must acknowledge the current interrupt by writing 1 to the HINT bit. When the host does this, the HPI clears the HINT bit (HINT = 0), and this drives the  $\overline{\text{UHPI\_HINT}}$  signal high. The CPU should read HPIC and make sure HINT = 0 before generating subsequent interrupts.

Writes of 0 have no effect. A hardware reset immediately clears the HINT bit and thus clears an active CPU-to-host interrupt.

### 18.2.10 EDMA Event Support

The HPI does not provide synchronization events to the EDMA system. Memory accesses from the HPI are handled automatically, independent of the EDMA controller. The HPI controller has its own dedicated DMA and its operation and configuration are transparent.

### 18.2.11 Power Management

The HPI peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the HPI peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.



### 18.2.12 Emulation Considerations

The FREE and SOFT bits in the power and emulation management register (PWREMU\_MGMT) determine the response of the HPI to an emulation suspend condition. If FREE = 1, the HPI is not affected, and the SOFT bit has no effect. If FREE = 0 and SOFT = 0, the HPI is not affected. If FREE = 0 and SOFT = 1:

- The HPI DMA logic halts after the current host and HPI DMA operations are completed.
- The external host interface functions as normal throughout the emulation suspend condition. The host may access the control register (HPIC). The host may also access the HPIA registers and may perform data reads until the read FIFO is empty or data writes until the write FIFO is full. As in normal operation,  $\overline{\text{UHPI\_HRDY}}$  is driven low during a host cycle that cannot be completed due to the write FIFO being full or the read FIFO being empty. If this occurs,  $\overline{\text{UHPI\_HRDY}}$  continues to be driven low, holding off the host, until the emulation suspend condition is over, and the FIFOs are serviced by the HPI DMA logic, allowing the host cycle to complete.
- When the emulation suspend condition is over, the appropriate requests by the HPI DMA logic are made to process any posted host writes in the write FIFO or to fill the read FIFO as necessary. HPI operation then continues as normal.

## 18.3 Registers

Table 18-6 lists the memory-mapped registers for the HPI. See your device-specific data manual for the memory addresses of these registers.

**Table 18-6. HPI Registers**

Offset	Acronym	Register Description	Section
0	REVID	Revision Identification Register	<a href="#">Section 18.3.1</a>
4h	PWREMU_MGMT	Power and Emulation Management Register	<a href="#">Section 18.3.2</a>
Ch	GPIO_EN	GPIO Enable Register	<a href="#">Section 18.3.3</a>
10h	GPIO_DIR1	GPIO Direction 1 Register	<a href="#">Section 18.3.4</a>
14h	GPIO_DAT1	GPIO Data 1 Register	<a href="#">Section 18.3.5</a>
18h	GPIO_DIR2	GPIO Direction 2 Register	<a href="#">Section 18.3.6</a>
1Ch	GPIO_DAT2	GPIO Data 2 Register	<a href="#">Section 18.3.7</a>
30h	HPIC	Host Port Interface Control Register	<a href="#">Section 18.3.8</a>
34h	HPIAW	Host Port Interface Write Address Register	<a href="#">Section 18.3.9</a>
38h	HPIAR	Host Port Interface Read Address Register	<a href="#">Section 18.3.10</a>

### 18.3.1 Revision Identification Register (REVID)

The revision identification register (REVID) contains identification data for the peripheral. REVID is shown in [Figure 18-17](#) and described in [Table 18-7](#).

**Figure 18-17. Revision Identification Register (REVID)**



LEGEND: R = Read only; -n = value after reset

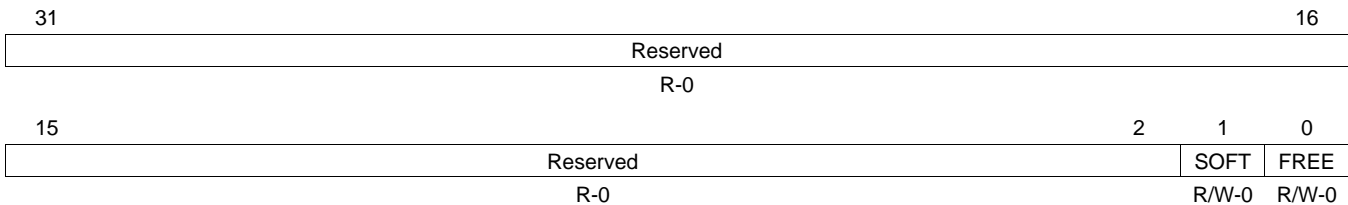
**Table 18-7. Revision Identification Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	0-4421 210Ah	Revision identification

### 18.3.2 Power and Emulation Management Register (PWREMU\_MGMT)

The power and emulation management register (PWREMU\_MGMT) determines the emulation mode of the HPI. PWREMU\_MGMT is shown in [Figure 18-18](#) and described in [Table 18-8](#).

**Figure 18-18. Power and Emulation Management Register (PWREMU\_MGMT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

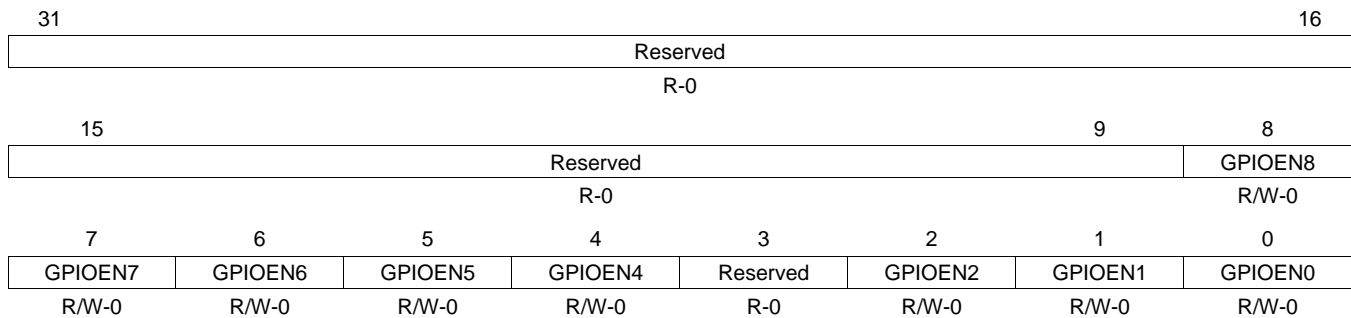
**Table 18-8. Power and Emulation Management Register (PWREMU\_MGMT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT	0	Determines emulation mode functionality of the HPI. When the FREE bit is cleared to 0, the SOFT bit selects the HPI mode. Upon emulation suspend, the HPI operation is not affected.
		1	In response to an emulation suspend event, the HPI logic halts after the current HPI transaction is completed.
0	FREE	0	Free run emulation control. Determines emulation mode functionality of the HPI. When the FREE bit is cleared to 0, the SOFT bit selects the HPI mode. The SOFT bit selects the HPI mode.
		1	The HPI runs free regardless of the SOFT bit.

### 18.3.3 GPIO Enable Register (GPIO\_EN)

The GPIO enable register (GPIO\_EN) enables the pin for general-purpose I/O. GPIO\_EN is shown in Figure 18-19 and described in Table 18-9.

**Figure 18-19. GPIO Enable Register (GPIO\_EN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 18-9. GPIO Enable Register (GPIO\_EN) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	GPIOEN8	0	Enable as GPIO for UHPI_HD[15:8] pins
		1	Disable pins for GPIO. Pins functions as HPI signal. Enable pins for GPIO.
7	GPIOEN7	0	Enable as GPIO for UHPI_HD[7:0] pins.
		1	Disable pins for GPIO. Pins functions as HPI signal. Enable pins for GPIO.
6	GPIOEN6	0	Enable as GPIO for UHPI_HINT pin.
		1	Disable pin for GPIO. Pin functions as HPI signal. Enable pin for GPIO.
5	GPIOEN5	0	Enable as GPIO for UHPI_HRDY pin.
		1	Disable pin for GPIO. Pin functions as HPI signal. Enable pin for GPIO.
4	GPIOEN4	0	Enable as GPIO for UHPI_HHWIL pin.
		1	Disable pin for GPIO. Pin functions as HPI signal. Enable pin for GPIO.
3	Reserved	0	Reserved
2	GPIOEN2	0	Enable as GPIO for UHPI_HAS pin.
		1	Disable pin for GPIO. Pin functions as HPI signal. Enable pin for GPIO.
1	GPIOEN1	0	Enable as GPIO for UHPI_HCNTL[1:0] pins.
		1	Disable pins for GPIO. Pins functions as HPI signal. Enable pins for GPIO.
0	GPIOEN0	0	Enable as GPIO for UHPI_HCS, UHPI_HDS1, UHPI_HDS2, UHPI_HR/W pins.
		1	Disable pins for GPIO. Pins functions as HPI signal. Enable pins for GPIO.

### 18.3.4 GPIO Direction 1 Register (GPIO\_DIR1)

The GPIO direction 1 register (GPIO\_DIR1) determines if the UHPI\_HD $n$  pin is an input or an output. GPIO\_DIR1 is shown in [Figure 18-20](#) and described in [Table 18-10](#).

**Figure 18-20. GPIO Direction 1 Register (GPIO\_DIR1)**



**Table 18-10. GPIO Direction 1 Register (GPIO\_DIR1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	HD $n$	0	Direction control for UHPI_HD $n$ pin. UHPI_HD $n$ pin is an input.
		1	UHPI_HD $n$ pin is an output.

### 18.3.5 GPIO Data 1 Register (GPIO\_DAT1)

The GPIO data 1 register (GPIO\_DAT1) determines the value driven on the corresponding UHPI\_HD $n$  pin, if the pin is configured as an output (GPIO\_DIR1.HD $n$  = 1). Writes do not affect pins not configured as GPIO outputs. The bits in GPIO\_DAT1 are set or cleared by writing directly to this register. A read of GPIO\_DAT1 returns the value of the register bit (HD $n$ ) not the value at the UHPI\_HD $n$  pin (that might be configured as an input). GPIO\_DAT1 is shown in [Figure 18-21](#) and described in [Table 18-11](#).

**Figure 18-21. GPIO Data 1 Register (GPIO\_DAT1)**



**Table 18-11. GPIO Data 1 Register (GPIO\_DAT1) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	HD $n$	0-1	Data read from/written to UHPI_HD $n$ pin.

### 18.3.6 GPIO Direction 2 Register (GPIO\_DIR2)

The GPIO direction 2 register (GPIO\_DIR2) determines if the HPI pin is an input or an output. GPIO\_DIR2 is shown in [Figure 18-22](#) and described in [Table 18-12](#).

**Figure 18-22. GPIO Direction 2 Register (GPIO\_DIR2)**

	Reserved	
	R-0	
15	Reserved	10      9      8
	R-0	HRDY      HINTZ
		R/W-0      R/W-0
7	6      5      4      3      2	1      0
HCNTL0	HCNTL1      HHWIL      HRW      HDS2Z      HDS1Z	HCSZ      HASZ
R/W-0	R/W-0      R/W-0      R/W-0      R/W-0      R/W-0	R/W-0      R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 18-12. GPIO Direction 2 Register (GPIO\_DIR2) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	HRDY	0	Direction control for UHPI_HRDY pin. UHPI_HRDY pin is an input.
		1	UHPI_HRDY pin is an output.
8	HINTZ	0	Direction control for UHPI_HINT pin. UHPI_HINT pin is an input.
		1	UHPI_HINT pin is an output.
7	HCNTL0	0	Direction control for UHPI_HCNTL0 pin. UHPI_HCNTL0 pin is an input.
		1	UHPI_HCNTL0 pin is an output.
6	HCNTL1	0	Direction control for UHPI_HCNTL1 pin. UHPI_HCNTL1 pin is an input.
		1	UHPI_HCNTL1 pin is an output.
5	HHWIL	0	Direction control for UHPI_HHWIL pin. UHPI_HHWIL pin is an input.
		1	UHPI_HHWIL pin is an output.
4	HRW	0	Direction control for UHPI_HR/W pin. UHPI_HR/W pin is an input.
		1	UHPI_HR/W pin is an output.
3	HDS2Z	0	Direction control for UHPI_HDS2 pin. UHPI_HDS2 pin is an input.
		1	UHPI_HDS2 pin is an output.
2	HDS1Z	0	Direction control for UHPI_HDS1 pin. UHPI_HDS1 pin is an input.
		1	UHPI_HDS1 pin is an output.
1	HCSZ	0	Direction control for UHPI_HCS pin. UHPI_HCS pin is an input.
		1	UHPI_HCS pin is an output.
0	HASZ	0	Direction control for UHPI_HAS pin. UHPI_HAS pin is an input.
		1	UHPI_HAS pin is an output.

### 18.3.7 GPIO Data 2 Register (GPIO\_DAT2)

The GPIO data 2 register (GPIO\_DAT2) determines the value driven on the corresponding HPI pin, if the pin is configured as an output (GPIO\_DIR2 bit = 1). Writes do not affect pins not configured as GPIO outputs. The bits in GPIO\_DAT2 are set or cleared by writing directly to this register. A read of GPIO\_DAT2 returns the value of the register bit not the value at the HPI pin (that might be configured as an input). GPIO\_DAT2 is shown in [Figure 18-23](#) and described in [Table 18-13](#).

**Figure 18-23. GPIO Data 2 Register (GPIO\_DAT2)**

	Reserved	
	R-0	
15	Reserved	10      9      8
	R-0	HRDY      HINTZ
		R/W-0      R/W-0
7	6      5      4      3      2	1      0
HCNTL0	HCNTL1      HHWIL      HRW      HDS2Z      HDS1Z	HCSZ      HASZ
R/W-0	R/W-0      R/W-0      R/W-0      R/W-0      R/W-0	R/W-0      R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 18-13. GPIO Data 2 Register (GPIO\_DAT2) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	HRDY	0-1	Data read from/written to UHPI_HRDY pin.
8	HINTZ	0-1	Data read from/written to UHPI_HINT pin.
7	HCNTL0	0-1	Data read from/written to UHPI_HCNTL0 pin.
6	HCNTL1	0-1	Data read from/written to UHPI_HCNTL1 pin.
5	HHWIL	0-1	Data read from/written to UHPI_HHWIL pin.
4	HRW	0-1	Data read from/written to UHPI_HRW pin.
3	HDS2Z	0-1	Data read from/written to UHPI_HDS2 pin.
2	HDS1Z	0-1	Data read from/written to UHPI_HDS1 pin.
1	HCSZ	0-1	Data read from/written to UHPI_HCS pin.
0	HASZ	0-1	Data read from/written to UHPI_HAS pin.

### 18.3.8 Host Port Interface Control Register (HPIC)

The host port interface control register (HPIC) stores configuration and control information for the HPI. As shown in Figure 18-24 and Figure 18-25 and described in Table 18-14, the host and CPU do not have the same access permissions. The host has full read/write access; the CPU has primarily read-only access, but with the exception that the CPU can write 1 to the HINT bit to generate an interrupt to the host.

**Figure 18-24. Host Port Interface Control Register (HPIC)—Host Access Permissions**

31							16								
Reserved															
R-0															
15			12			11		10		9		8			
Reserved			HPIASEL			Reserved		DUALHPIA		HWOBSTAT					
R-0			R/W-0			R/W-0		R/W-0		R-0					
7		6		5		4		3		2		1		0	
HPIRST		Reserved		FETCH		Reserved		HINT		DSPINT		HWOB			
R-1		R-2h		R/W-0		R-1		R/W-1 (Host) R/W1C-0 (CPU)		R/W-0		R/W-0			

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Figure 18-25. Host Port Interface Control Register (HPIC)—CPU Access Permissions**

31							16								
Reserved															
R-0															
15			12			11		10		9		8			
Reserved			HPIASEL			Reserved		DUALHPIA		HWOBSTAT					
R-0			R-0			R-0		R-0		R-0					
7		6		5		4		3		2		1		0	
HPIRST		Reserved		FETCH		Reserved		HINT		DSPINT		HWOB			
R/W-1		R-2h		R-0		R-1		R/W-1 (Host) R/W1C-0 (CPU)		R/W-0		R-0			

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 18-14. Host Port Interface Control Register (HPIC) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	HPIASEL	0 1	HPI address register select bit. When DUALHPIA = 1, the HPIASEL bit is used to select the HPI address register to be accessed. 0 Selects the HPI write address register (HPIAW). 1 Selects the HPI read address register (HPIAR).
10	Reserved	0	Reserved. Always write 0 to this bit.
9	DUALHPIA	0 1	Dual HPIA mode configuration bit. The CPU can access both HPI address registers separately, regardless of the DUALHPIA setting. (Regardless of this bit, dual HPIA mode is implied when the CPU has ownership of the HPI address registers). 0 The two HPI address registers (HPIAW and HPIAR) operate as a single HPI address register in terms of host accesses. 1 Dual HPIA mode operation is enabled.
8	HWOBSTAT	0 1	HWOB status. The value of the HWOB bit is also stored in this bit position. A write to the HWOB bit also updates HWOBSTAT. 0 HWOB bit is logic 0. 1 HWOB bit is logic 1.
7	HPIRST	0 1	HPI reset. Some HPI logic is held in reset when the HPIRST bit is set. The HPIRST bit must be cleared to 0 before data transactions can take place. 0 HPI is released from reset. 1 HPI is held in reset.
6-5	Reserved	2h	Reserved
4	FETCH		Host data fetch request bit. Only the host may write to FETCH. When a host writes a 1 to FETCH, a request is posted in the HPI to prefetch data into the read FIFO. Host and CPU reads of FETCH return a 0.
3	Reserved	1	Reserved
2	HINT	0 1	Processor-to-host interrupt. The CPU writes a 1 to HINT to generate a host interrupt. HINT has an inverted logic level to the $\overline{\text{UHPI\_HINT}}$ pin. The host must write a 1 to HINT to clear the $\overline{\text{UHPI\_HINT}}$ pin; writing a 0 to HINT by the host or processor has no effect. 0 No effect. 1 A CPU write generates a host interrupt ( $\overline{\text{UHPI\_HINT}}$ signal goes low). A host write sets the $\overline{\text{UHPI\_HINT}}$ signal high (clears the interrupt).
1	DSPINT	0 1	Host-to-processor interrupt. The host writes a 1 to DSPINT to generate a processor interrupt; writing a 0 to DSPINT by the host or processor has no effect. 0 No effect. 1 A host write generates a processor interrupt.
0	HWOB	0 1	Halfword ordering bit. HWOB affects both data and address transfers. HWOB must be initialized before the first data or address register access. 0 First halfword is most significant. 1 First halfword is least significant.

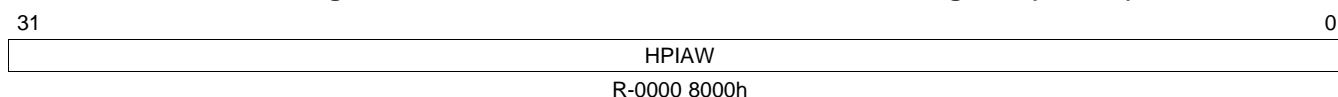


### 18.3.9 Host Port Interface Write Address Register (HPIAW)

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). The host port interface write address register (HPIAW) is shown in [Figure 18-26](#) and described in [Table 18-15](#). The HPI can be configured such that HPIAR and HPIAW act as a single 32-bit HPIA (single-HPIA mode) or as two separate 32-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see [Section 18.2.6.1](#).

Note that the addresses loaded into the HPI address registers can be configured by the HPIBYTEAD bit in the chip configuration 1 register (CFGCHIP1) of the system configuration module. If byte address is selected (HPIBYTEAD = 1), the address must be 32-bit word aligned (with the least-significant two bits equal to zero).

**Figure 18-26. Host Port Interface Write Address Register (HPIAW)**



LEGEND: R = Read only; -n = value after reset

**Table 18-15. Host Port Interface Write Address Register (HPIAW) Field Descriptions**

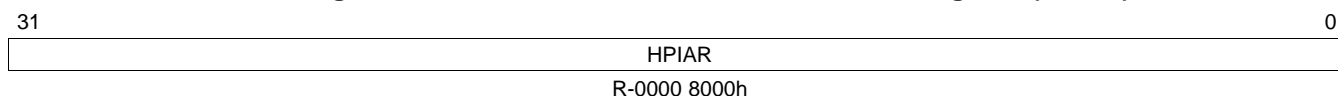
Bit	Field	Value	Description
31-0	HPIAW	0-FFFF FFFFh	Host port interface write address.

### 18.3.10 Host Port Interface Read Address Register (HPIAR)

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). The host port interface read address register (HPIAR) is shown in [Figure 18-27](#) and described in [Table 18-16](#). The HPI can be configured such that HPIAR and HPIAW act as a single 32-bit HPIA (single-HPIA mode) or as two separate 32-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see [Section 18.2.6.1](#).

Note that the addresses loaded into the HPI address registers can be configured by the HPIBYTEAD bit in the chip configuration 1 register (CFGCHIP1) of the system configuration module. If byte address is selected (HPIBYTEAD = 1), the address must be 32-bit word aligned (with the least-significant two bits equal to zero).

**Figure 18-27. Host Port Interface Read Address Register (HPIAR)**



LEGEND: R = Read only; -n = value after reset

**Table 18-16. Host Port Interface Read Address Register (HPIAR) Field Descriptions**

Bit	Field	Value	Description
31-0	HPIAR	0-FFFF FFFFh	Host port interface read address.

---

---

## ***Inter-Integrated Circuit (I2C) Module***

---

---

This chapter describes the inter-integrated circuit (I2C) peripheral. The scope of this chapter assumes that you are familiar with the Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1.

<b>Topic</b>	<b>Page</b>
<b>19.1 Introduction</b> .....	<b>736</b>
<b>19.2 Architecture</b> .....	<b>738</b>
<b>19.3 Registers</b> .....	<b>750</b>

## 19.1 Introduction

### 19.1.1 Purpose of the Peripheral

The I2C peripheral provides an interface between the SoC and other devices that are compliant with the I2C-bus specification and connected by way of an I2C-bus. External components that are attached to this two-wire serial bus can transmit and receive data that is up to eight bits wide both to and from the SoC through the I2C peripheral.

### 19.1.2 Features

The I2C peripheral has the following features:

- Compliance with the Philips Semiconductors I2C-bus specification (version 2.1):
  - Support for byte format transfer
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers mode
  - Support for multiple slave-transmitters and master-receivers mode
  - Combined master transmit/receive and receive/transmit mode
  - I2C data transfer rate of from 10 kbps up to 400 kbps (Philips I2C rate)
- 2-bit to 8-bit format transfer
- Free data format mode
- One read DMA event and one write DMA event that the DMA can use
- Seven interrupts that the CPU can use
- Peripheral enable/disable capability

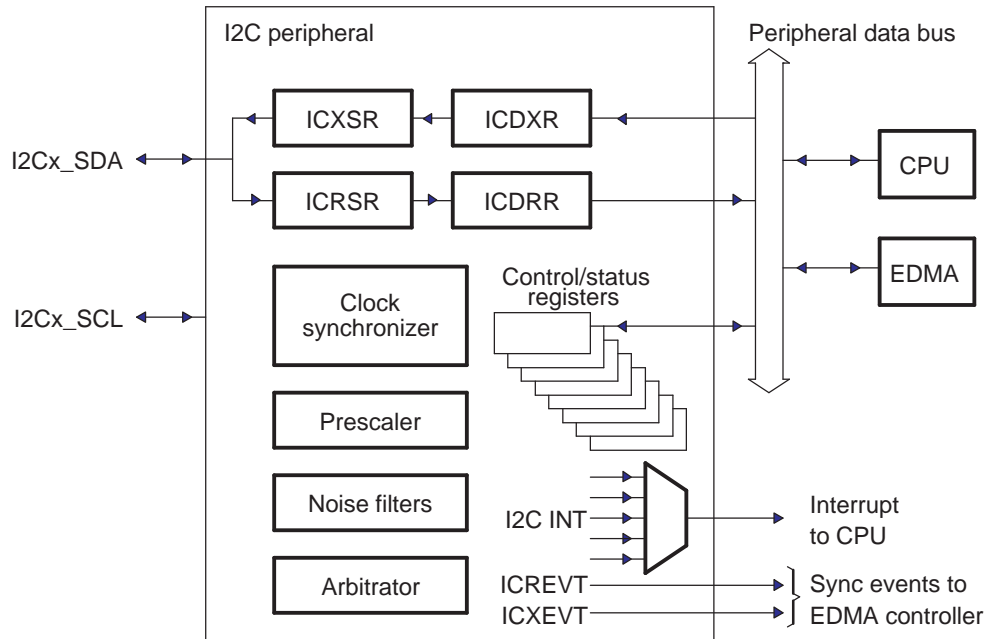
#### 19.1.2.1 Features Not Supported

- High-speed mode
- CBUS-compatibility mode
- The combined format in 10-bit addressing mode (the I2C sends the slave address the second byte every time it sends the slave address the first byte).

### 19.1.3 Functional Block Diagram

A block diagram of the I2C peripheral is shown in [Figure 20-1](#). Refer to [Section 19.2](#) for detailed information about the architecture of the I2C peripheral.

**Figure 19-1. I2C Peripheral Block Diagram**



### 19.1.4 Industry Standard(s) Compliance Statement

The I2C peripheral is compliant with the Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1.

## 19.2 Architecture

The I2C peripheral consists of the following primary blocks:

- A serial interface: one data pin (I2Cx\_SDA) and one clock pin (I2Cx\_SCL)
- Data registers to temporarily hold receive data and transmit data traveling between the I2Cx\_SDA pin and the CPU or the EDMA controller
- Control and status registers
- A peripheral data bus interface to enable the CPU and the EDMA controller to access the I2C peripheral registers
- A clock synchronizer to synchronize the I2C input clock (from the processor clock generator) and the clock on the I2Cx\_SCL pin, and to synchronize data transfers with masters of different clock speeds
- A prescaler to divide down the input clock that is driven to the I2C peripheral
- A noise filter on each of the two pins, I2Cx\_SDA and I2Cx\_SCL
- An arbitrator to handle arbitration between the I2C peripheral (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- EDMA event generation logic, so that activity in the EDMA controller can be synchronized to data reception and data transmission in the I2C peripheral

Figure 20-1 shows the four registers used for transmission and reception. The CPU or the EDMA controller writes data for transmission to ICDXR and reads received data from ICDRR. When the I2C peripheral is configured as a transmitter, data written to ICDXR is copied to ICXSR and shifted out on the I2Cx\_SDA pin one bit at a time. When the I2C peripheral is configured as a receiver, received data is shifted into ICRSR and then copied to ICDRR.

### 19.2.1 Bus Structure

Figure 20-1 shows how the I2C peripheral is connected to the I2C bus. The I2C bus is a multi-master bus that supports a multi-master mode. This allows more than one device capable of controlling the bus that is connected to it. A unique address recognizes each I2C device. Each I2C device can operate as either transmitter or receiver, depending on the function of the device. Devices that are connected to the I2C bus can be considered a master or slave when performing data transfers, in addition to being a transmitter or receiver.

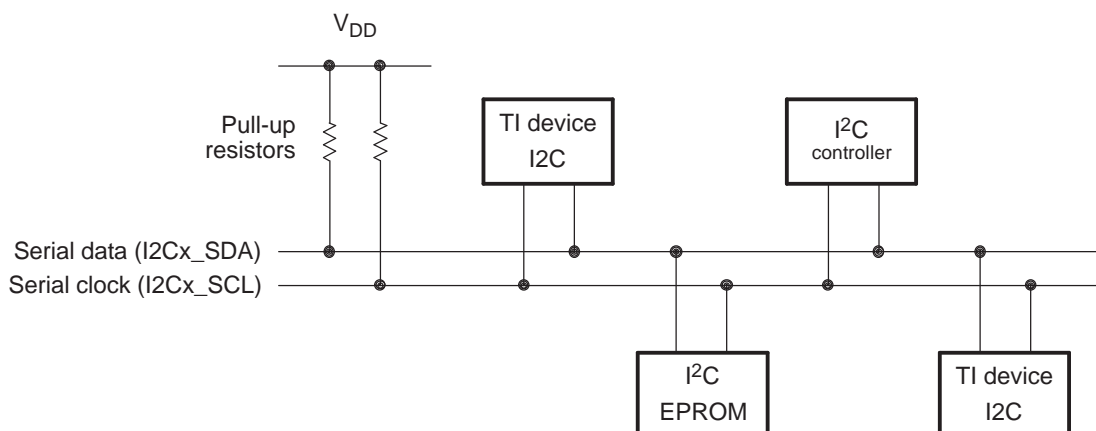
---

**NOTE:** A master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device that is addressed by this master is considered a slave during this transfer.

---

An example of multiple I2C modules that are connected for a two-way transfer from one device to other devices is shown in Figure 19-2.

**Figure 19-2. Multiple I2C Modules Connected**

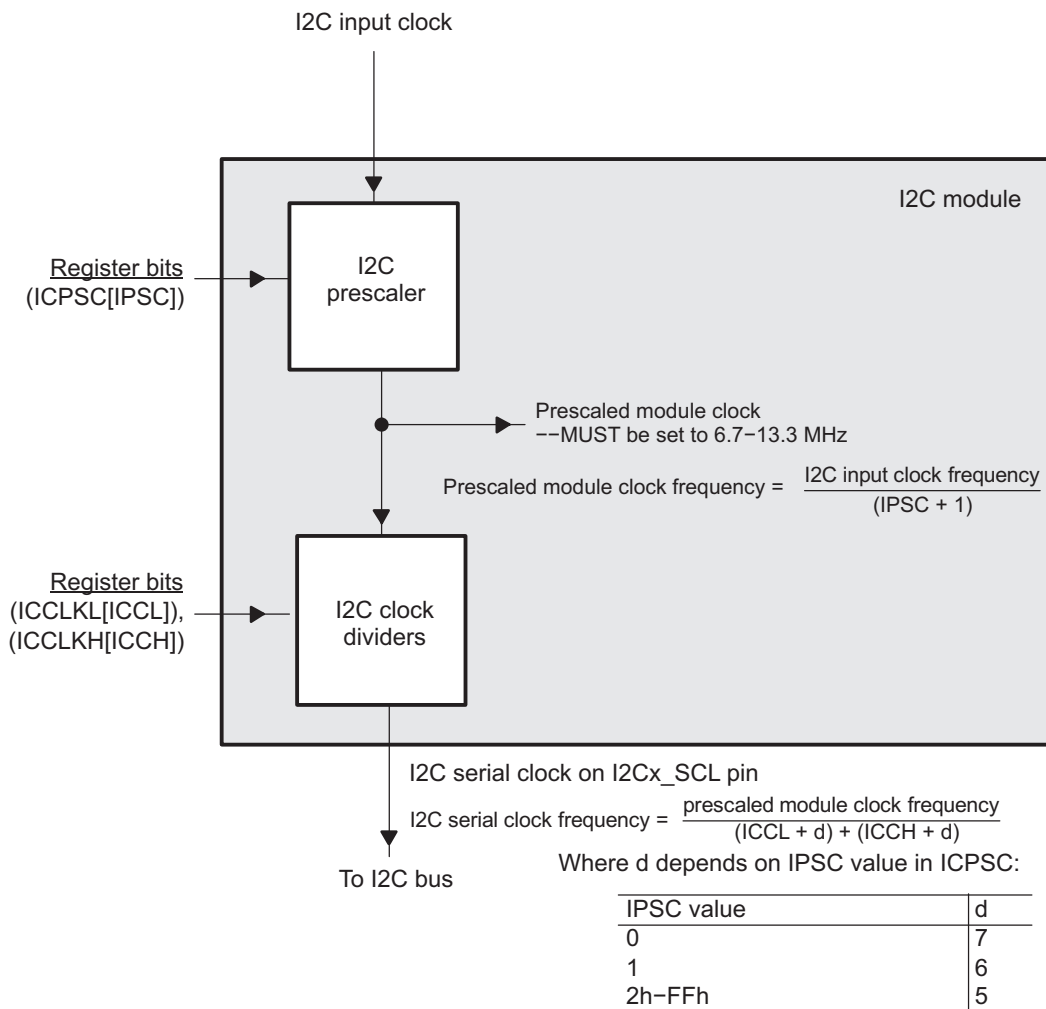


### 19.2.2 Clock Generation

As shown in Figure 19-3, I2C input clock is fed to the I2C module. A programmable prescaler (IPSC bit in ICPSC) in the I2C module divides down the I2C input clock to produce a prescaled module clock. The prescaled module clock must be operated within the range of 6.7 to 13.3 MHz. The I2C clock dividers divide-down the high (ICCH bit in ICCLKH) and low portions (ICCL bit in ICCLKL) of the prescaled module clock signal to produce the I2C serial clock, which appears on the I2Cx\_SCL pin when the I2C module is configured to be a master on the I2C bus.

The prescaler (IPSC bit in ICPSC) must only be initialized while the I2C module is in the reset state (IRS = 0 in ICMR). The prescaled frequency only takes effect when the IRS bit in ICMR is changed to 1. Changing the IPSC bit in ICPSC while IRS = 1 in ICMR has no effect. Likewise, you must configure the I2C clock dividers (ICCH bit in ICCLKH and ICCL bit in ICCLKL) while the I2C module is still in reset (IRS = 0 in ICMR).

**Figure 19-3. Clocking Diagram for the I2C Peripheral**



**CAUTION**

**Prescaled Module Clock Frequency Range:**

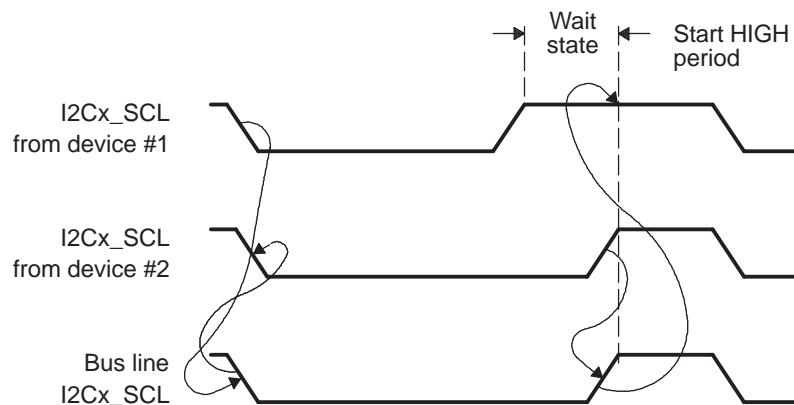
The I2C module must be operated with a prescaled module clock frequency of 6.7 to 13.3 MHz. The I2C prescaler register (ICPSC) must be configured to this frequency range.

### 19.2.3 Clock Synchronization

Only one master device generates the clock signal (I2Cx\_SCL) under normal conditions. However, there are two or more masters during the arbitration procedure; and, you must synchronize the clock so that you can compare the data output. Figure 19-4 illustrates the clock synchronization. The wired-AND property of I2Cx\_SCL means that a device that first generates a low period on I2Cx\_SCL (device #1) overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The I2Cx\_SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for I2Cx\_SCL to be released before starting their high periods. A synchronized signal on I2Cx\_SCL is obtained, where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. This way, a slave slows down a fast master and the slow device creates enough time to store a received data word or to prepare a data word that you are going to transmit.

**Figure 19-4. Synchronization of Two I2C Clock Generators During Arbitration**



### 19.2.4 Signal Descriptions

The I2C peripheral has a serial data pin (I2Cx\_SDA) and a serial clock pin (I2Cx\_SCL) for data communication, as shown in Figure 20-1. These two pins carry information between the device and other devices that are connected to the I2C-bus. The I2Cx\_SDA and I2Cx\_SCL pins both are bi-directional. They each must be connected to a positive supply voltage using a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the required wired-AND function.

See your device-specific data manual for additional timing and electrical specifications for these pins.

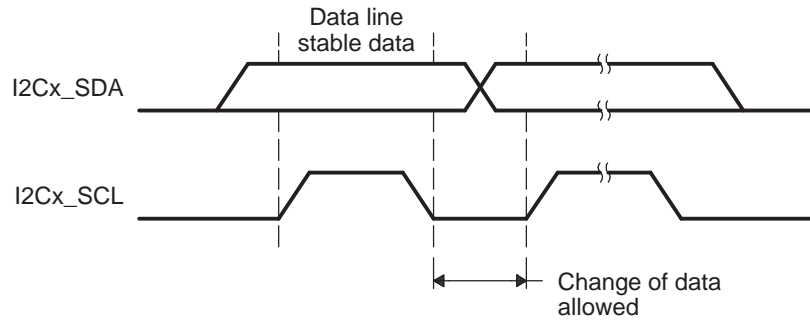
#### 19.2.4.1 Input and Output Voltage Levels

The master device generates one clock pulse for each data bit that is transferred. Due to a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated power supply level. See your device-specific data manual for more information.

### 19.2.4.2 Data Validity

The data on I2Cx\_SDA must be stable during the high period of the clock (see Figure 19-5). The high or low state of the data line, I2Cx\_SDA, can change only when the clock signal on I2Cx\_SCL is low.

Figure 19-5. Bit Transfer on the I2C-Bus



### 19.2.5 START and STOP Conditions

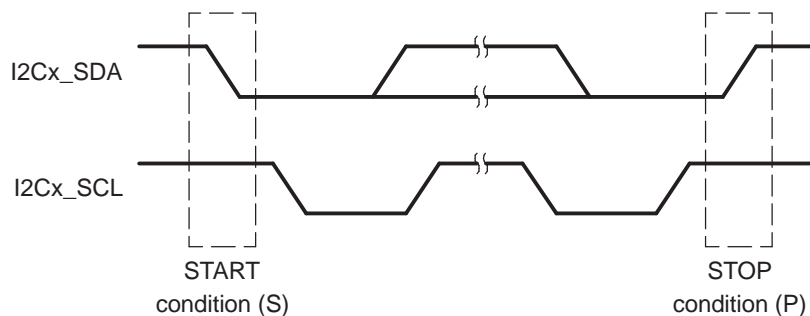
The I2C peripheral can generate START and STOP conditions when the peripheral is configured to be a master on the I2C-bus, as shown in Figure 19-6:

- The START condition is defined as a high-to-low transition on the I2Cx\_SDA line while I2Cx\_SCL is high. A master drives this condition to indicate the start of a data transfer.
- The STOP condition is defined as a low-to-high transition on the I2Cx\_SDA line while I2Cx\_SCL is high. A master drives this condition to indicate the end of a data transfer.

The I2C-bus is considered busy after a START condition and before a subsequent STOP condition. The bus busy (BB) bit of ICSTR is 1. The bus is considered free between a STOP condition and the next START condition. The BB is 0.

The master mode (MST) bit and the START condition (STT) bit in ICMDR must both be 1 for the I2C peripheral to start a data transfer with a START condition. The STOP condition (STP) bit must be set to 1 for the I2C peripheral to end a data transfer with a STOP condition. A repeated START condition generates when BB is set to 1 and STT is also set to 1. See Section 19.3.9 for a description of ICMDR (including the MST, STT, and STP bits).

Figure 19-6. I2C Peripheral START and STOP Conditions





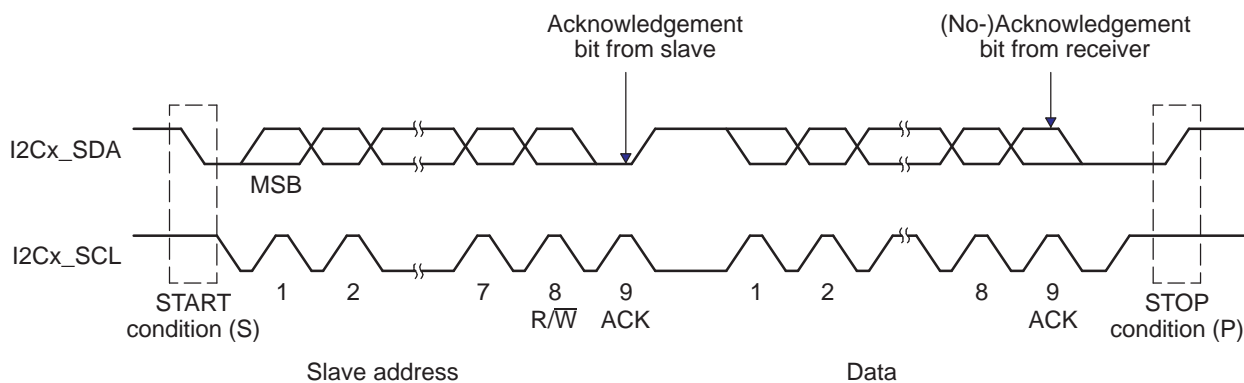
### 19.2.6 Serial Data Formats

Figure 19-7 shows an example of a data transfer on the I2C-bus. The I2C peripheral supports 1-bit to 8-bit data values. Figure 19-7 is shown in an 8-bit data format (BC = 000 in ICM DR). Each bit put on the I2Cx\_SDA line is equivalent to one pulse on the I2Cx\_SCL line. The data is always transferred with the most-significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted; however, the transmitters and receivers must agree on the number of data values being transferred.

The I2C peripheral supports the following data formats:

- 7-bit addressing mode
- 10-bit addressing mode
- Free data format mode

**Figure 19-7. I2C Peripheral Data Transfer**



#### 19.2.6.1 7-Bit Addressing Format

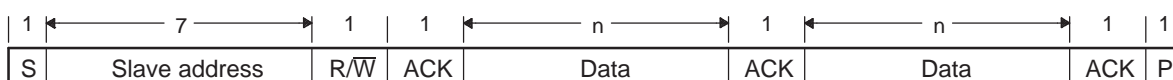
In the 7-bit addressing format (Figure 19-8), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/W bit. The R/W bit determines the direction of the data.

- $R/\overline{W} = 0$ : The master writes (transmits) data to the addressed slave.
- $R/\overline{W} = 1$ : The master reads (receives) data from the slave.

An extra clock cycle dedicated for acknowledgment (ACK) is inserted after the R/W bit. If the slave inserts the ACK bit,  $n$  bits of data from the transmitter (master or slave, depending on the R/W bit) follow it.  $n$  is a number from 1 to 8 that the bit count (BC) bits of ICM DR determine. The receiver inserts an ACK bit after the data bits have been transferred.

Write a 0 to the expanded address enable (XA) bit of ICM DR to select the 7-bit addressing format.

**Figure 19-8. I2C Peripheral 7-Bit Addressing Format (FDF = 0, XA = 0 in ICM DR)**



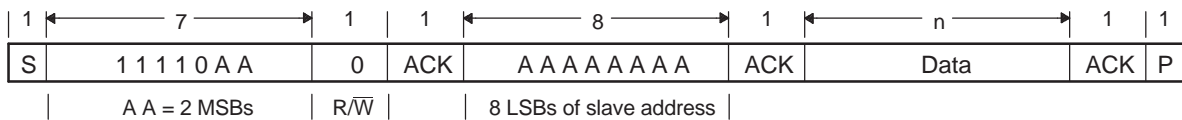
$n$  = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICM DR.

### 19.2.6.2 10-Bit Addressing Format

The 10-bit addressing format (Figure 19-9) is like the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and  $R/\overline{W} = 0$  (write). The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgment (ACK) after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. (For more information about using 10-bit addressing, see the Philips Semiconductors I2C-bus specification.)

Write 1 to the XA bit of ICMDR to select the 10-bit addressing format.

**Figure 19-9. I2C Peripheral 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in ICMDR)**



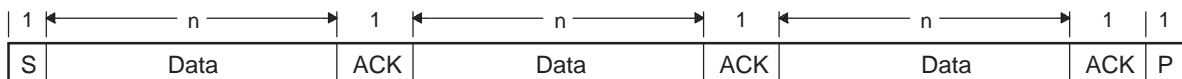
n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICMDR.

### 19.2.6.3 Free Data Format

In the free data format (Figure 19-10), the first bits after a START condition (S) are a data word. An ACK bit is inserted after each data word. The data word can be from 1 to 8 bits, depending on the bit count (BC) bits of ICMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.

To select the free data format, write 1 to the free data format (FDF) bit of ICMDR.

**Figure 19-10. I2C Peripheral Free Data Format (FDF = 1 in ICMDR)**

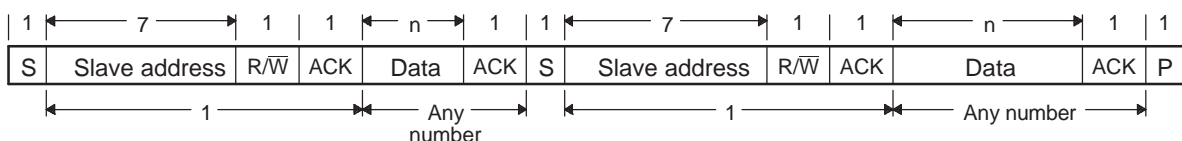


n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICMDR.

### 19.2.6.4 Using a Repeated START Condition

The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. The 7-bit addressing format using a repeated START condition (S) is shown in Figure 19-11. At the end of each data word, the master can drive another START condition. Using this capability, a master can transmit/receive any number of data words before driving a STOP condition. The length of a data word can be from 1 to 8 bits and is selected with the bit count (BC) bits of ICMDR.

**Figure 19-11. I2C Peripheral 7-Bit Addressing Format With Repeated START Condition (FDF = 0, XA = 0 in ICMDR)**



n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICMDR.

## 19.2.7 Operating Modes

The I2C peripheral has four basic operating modes to support data transfers as a master and as a slave. See [Table 19-1](#) for the names and descriptions of the modes.

If the I2C peripheral is a master, it begins as a master-transmitter and, typically, transmits an address for a particular slave. When giving data to the slave, the I2C peripheral must remain a master-transmitter. In order to receive data from a slave, the I2C peripheral must be changed to the master-receiver mode.

If the I2C peripheral is a slave, it begins as a slave-receiver and, typically, sends acknowledgment when it recognizes its slave address from a master. If the master will be sending data to the I2C peripheral, the peripheral must remain a slave-receiver. If the master has requested data from the I2C peripheral, the peripheral must be changed to the slave-transmitter mode.

**Table 19-1. Operating Modes of the I2C Peripheral**

Operating Mode	Description
Slave-receiver mode	The I2C peripheral is a slave and receives data from a master. All slave modules begin in this mode. In this mode, serial data bits received on I2Cx_SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C peripheral does not generate the clock signal, but it can hold I2Cx_SCL low while the intervention of the processor is required (RSFULL = 1 in ICSTR) after data has been received.
Slave-transmitter mode	The I2C peripheral is a slave and transmits data to a master. This mode can only be entered from the slave-receiver mode; the I2C peripheral must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C peripheral enters its slave-transmitter mode if the slave address is the same as its own address (in ICOAR) and the master has transmitted $R/\overline{W} = 1$ . As a slave-transmitter, the I2C peripheral then shifts the serial data out on I2Cx_SDA with the clock pulses that are generated by the master. While a slave, the I2C peripheral does not generate the clock signal, but it can hold I2Cx_SCL low while the intervention of the processor is required (XSMT = 0 in ICSTR) after data has been transmitted.
Master-receiver mode	The I2C peripheral is a master and receives data from a slave. This mode can only be entered from the master-transmitter mode; the I2C peripheral must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C peripheral enters its master-receiver mode after transmitting the slave address and $R/\overline{W} = 1$ . Serial data bits on I2Cx_SDA are shifted into the I2C peripheral with the clock pulses generated by the I2C peripheral on I2Cx_SCL. The clock pulses are inhibited and I2Cx_SCL is held low when the intervention of the processor is required (RSFULL = 1 in ICSTR) after data has been received.
Master-transmitter mode	The I2C peripheral is a master and transmits control information and data to a slave. All master modules begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on I2Cx_SDA. The bit shifting is synchronized with the clock pulses generated by the I2C peripheral on I2Cx_SCL. The clock pulses are inhibited and I2Cx_SCL is held low when the intervention of the processor is required (XSMT = 0 in ICSTR) after data has been transmitted.

### 19.2.8 NACK Bit Generation

When the I2C peripheral is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C peripheral must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. [Table 19-2](#) summarizes the various ways the I2C peripheral sends a NACK bit.

**Table 19-2. Ways to Generate a NACK Bit**

I2C Peripheral Condition	NACK Bit Generation	
	Basic	Optional
Slave-receiver mode	<ul style="list-style-type: none"> <li>• Disable data transfers (STT = 0 in ICSTR).</li> <li>• Allow an overrun condition (RSFULL = 1 in ICSTR).</li> <li>• Reset the peripheral (IRS = 0 in ICMDR)</li> </ul>	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Repeat mode (RM = 1 in ICMDR)	<ul style="list-style-type: none"> <li>• Generate a STOP condition (STOP = 1 in ICMDR).</li> <li>• Reset the peripheral (IRS = 0 in ICMDR).</li> </ul>	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Nonrepeat mode (RM = 0 in ICMDR)	<ul style="list-style-type: none"> <li>• If STP = 1 in ICMDR, allow the internal data counter to count down to 0 and force a STOP condition.</li> <li>• If STP = 0, make STP = 1 to generate a STOP condition.</li> <li>• Reset the peripheral (IRS = 0 in ICMDR).</li> </ul>	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.

### 19.2.9 Arbitration

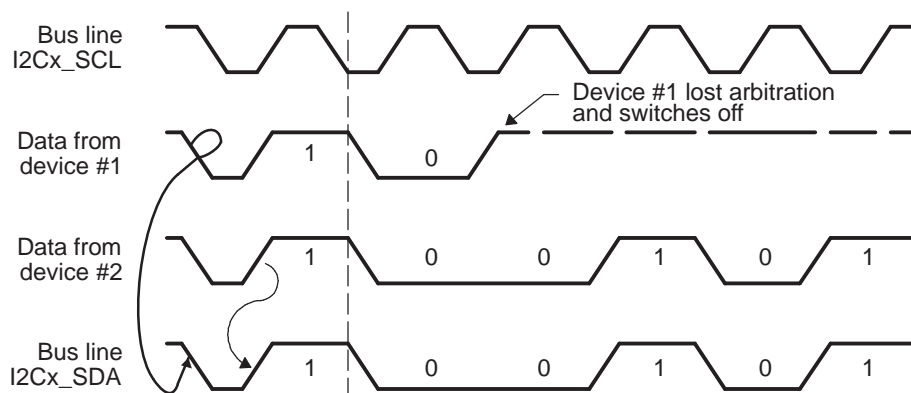
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (I2Cx\_SDA) by the competing transmitters. Figure 19-12 illustrates the arbitration procedure between two devices. The first master-transmitter, which drives I2Cx\_SDA high, is overruled by another master-transmitter that drives I2Cx\_SDA low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

If the I2C peripheral is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to I2Cx\_SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

**Figure 19-12. Arbitration Procedure Between Two Master-Transmitters**



### 19.2.10 Reset Considerations

The I2C peripheral has two reset sources: software reset and hardware reset.

#### 19.2.10.1 Software Reset Considerations

To reset the I2C peripheral, write 0 to the I2C reset (IRS) bit in the I2C mode register (ICMDR). All status bits in the I2C interrupt status register (ICSTR) are forced to their default values, and the I2C peripheral remains disabled until IRS is changed to 1. The I2Cx\_SDA and I2Cx\_SCL pins are in the high-impedance state.

---

**NOTE:** If the IRS bit is cleared to 0 during a transfer, this can cause the I2C bus to hang (I2Cx\_SDA and I2Cx\_SCL are in the high-impedance state).

---

#### 19.2.10.2 Hardware Reset Considerations

When a hardware reset occurs, all the registers of the I2C peripheral are set to their default values and the I2C peripheral remains disabled until the I2C reset (IRS) bit in the I2C mode register (ICMDR) is changed to 1.

---

**NOTE:** The IRS bit must be cleared to 0 while you configure/reconfigure the I2C peripheral. Forcing IRS to 0 can be used to save power and to clear error conditions.

---

### 19.2.11 Initialization

Proper I2C initialization is required prior to starting communication with other I2C device(s). Unless a fully fledged driver is in place, you need to determine the required I2C configuration needed (for example, Master Receiver, etc.) and configure the I2C controller with the desired settings. Enabling the I2C clock should be the first task. Then the I2C controller is placed in reset. You now are ready to configure the I2C controller. Once configuration is done, you need to enable the I2C controller by releasing the controller from reset. Prior to starting communication, you need to make sure that all status bits are cleared and no pending interrupts exist. Once the bus is determined to be available (the bus is not busy), the I2C is ready to proceed with the desired communication.

### 19.2.11.1 Configuring the I2C in Master Receiver Mode and Servicing Receive Data via CPU

The following initialization procedure is for the I2C controller configured in Master Receiver mode. The CPU is used to move data from the I2C receive register to CPU memory (memory accessible by the CPU).

1. Enable I2C clock from the Power and Sleep Controller, if it is driven by the Power and Sleep Controller (see the *Power and Sleep Controller (PSC)* chapter).
2. Place I2C in reset (clear IRS = 0 in ICMDR).
3. Configure ICMDR:
  - Configure I2C as Master (MST = 1).
  - Indicate the I2C configuration to be used; for example, Data Receiver (TRX = 0)
  - Indicate 7-bit addressing is to be used (XA = 0).
  - Disable repeat mode (RM = 0).
  - Disable loopback mode (DLB = 0).
  - Disable free data format (FDF = 0).
  - Optional: Disable start byte mode if addressing a fully fledged I2C device (STB = 0).
  - Set number of bits to transfer to be 8 bits (BC = 0).
4. Configure Slave Address: the I2C device this I2C master would be addressing (ICSAR = 7BIT ADDRESS).
5. Configure the peripheral clock operation frequency (ICPSC). This value should be selected in such a way that the frequency is between 6.7 and 13.3 MHz.
6. Configure I2C master clock frequency:
  - Configure the low-time divider value (ICCLKL).
  - Configure the high-time divider value (ICCLKH).
7. Make sure the interrupt status register (ICSTR) is cleared:
  - Read ICSTR and write it back (write 1 to clear) ICSTR = ICSTR
  - Read ICIVR until it is 0.
8. Take I2C controller out of reset: enable I2C controller (set IRS bit = 1 in ICMDR).
9. Wait until bus busy bit is cleared (BB = 0 in ICSTR).
10. Generate a START event, followed by Slave Address, etc. (set STT = 1 in ICMDR).
11. Wait until data is received (ICRRDY = 1 in ICSTR).
12. Read data:
  - If ICRRDY = 1 in ICSTR, then read ICDRR.
  - Perform the previous two steps until receiving one byte short of the entire byte expecting to receive.
13. Configure the I2C controller not to generate an ACK on the next/final byte reception: set NACKMOD bit for the I2C to generate a NACK on the last byte received (set NACKMOD = 1 in ICMDR).
14. End transfer/release bus when transfer is done. Generate a STOP event (set STP = 1 in ICMDR).

### 19.2.12 Interrupt Support

The I2C peripheral is capable of interrupting the CPU. The CPU can determine which I2C events caused the interrupt by reading the I2C interrupt vector register (ICIVR). ICIVR contains a binary-coded interrupt vector type to indicate which interrupt has occurred. Reading ICIVR clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there is more than one pending interrupt flag, reading ICIVR clears the highest-priority interrupt flag.

### 19.2.12.1 Interrupt Events and Requests

The I2C peripheral can generate the interrupts described in [Table 19-3](#). Each interrupt has a flag bit in the I2C interrupt status register (ICSTR) and a mask bit in the interrupt mask register (ICIMR). When one of the specified events occurs, its flag bit is set. If the corresponding mask bit is 0, the interrupt request is blocked; if the mask bit is 1, the request is forwarded to the CPU as an I2C interrupt.

**Table 19-3. Descriptions of the I2C Interrupt Events**

I2C Interrupt	Initiating Event
Arbitration-lost interrupt (AL)	Generated when the I2C arbitration procedure is lost or illegal START/STOP conditions occur
No-acknowledge interrupt (NACK)	Generated when the master I2C does not receive any acknowledge from the receiver
Registers-ready-for-access interrupt (ARDY)	Generated by the I2C when the previously programmed address, data and command have been performed and the status bits have been updated. This interrupt is used to let the controlling processor know that the I2C registers are ready to be accessed.
Receive interrupt/status (ICRINT and ICRRDY)	Generated when the received data in the receive-shift register (ICRSR) has been copied into the ICDRR. The ICRRDY bit can also be polled by the CPU to read the received data in the ICDRR.
Transmit interrupt/status (ICXINT and ICXRDY)	Generated when the transmitted data has been copied from ICDXR to the transmit-shift register (ICXSR) and shifted out on the I2Cx_SDA pin. This bit can also be polled by the CPU to write the next transmitted data into the ICDXR.
Stop-Condition-Detection interrupt (SCD)	Generated when a STOP condition has been detected
Address-as-Slave interrupt (AAS)	Generated when the I2C has recognized its own slave address or an address of all (8) zeros.

### 19.2.13 DMA Events Generated by the I2C Peripheral

For the EDMA controller to handle transmit and receive data, the I2C peripheral generates the following two EDMA events. Activity in EDMA channels can be synchronized to these events.

- **Receive event (ICREVT):** When receive data has been copied from the receive shift register (ICRSR) to the data receive register (ICDRR), the I2C peripheral sends an REVT signal to the EDMA controller. In response, the EDMA controller can read the data from ICDRR.
- **Transmit event (ICXEVT):** When transmit data has been copied from the data transmit register (ICDXR) to the transmit shift register (ICXSR), the I2C peripheral sends an XEVT signal to the EDMA controller. In response, the EDMA controller can write the next transmit data value to ICDXR.

### 19.2.14 Power Management

The I2C peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the I2C peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

### 19.2.15 Emulation Considerations

The response of the I2C events to emulation suspend events (such as halts and breakpoints) is controlled by the FREE bit in the I2C mode register (ICMDR). The I2C peripheral either stops exchanging data (FREE = 0) or continues to run (FREE = 1) when an emulation suspend event occurs. How the I2C peripheral terminates data transactions is affected by whether the I2C peripheral is acting as a master or a slave. For more information, see the description of the FREE bit in ICMDR (see [Section 19.3.9](#)).



## 19.3 Registers

Table 23-8 lists the memory-mapped registers for the inter-integrated circuit (I2C) peripheral. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 23-8 should be considered as reserved locations and the register contents should not be modified.

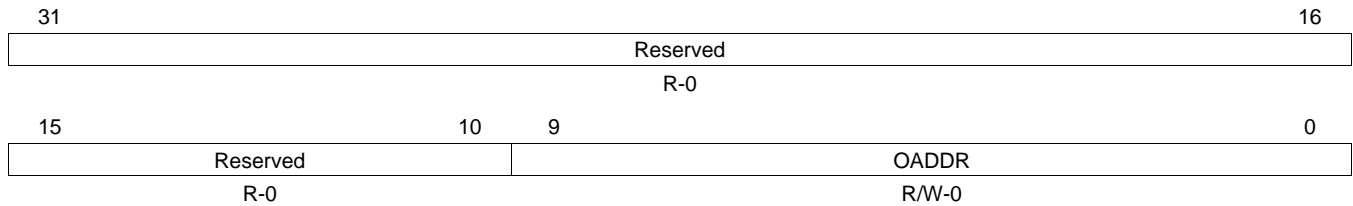
**Table 19-4. Inter-Integrated Circuit (I2C) Registers**

Offset	Acronym	Register Description	Section
0h	ICOAR	I2C Own Address Register	<a href="#">Section 19.3.1</a>
4h	ICIMR	I2C Interrupt Mask Register	<a href="#">Section 19.3.2</a>
8h	ICSTR	I2C Interrupt Status Register	<a href="#">Section 19.3.3</a>
Ch	ICCLKL	I2C Clock Low-Time Divider Register	<a href="#">Section 19.3.4</a>
10h	ICCLKH	I2C Clock High-Time Divider Register	<a href="#">Section 19.3.4</a>
14h	ICCNT	I2C Data Count Register	<a href="#">Section 19.3.5</a>
18h	ICDRR	I2C Data Receive Register	<a href="#">Section 19.3.6</a>
1Ch	ICSAR	I2C Slave Address Register	<a href="#">Section 19.3.7</a>
20h	ICDXR	I2C Data Transmit Register	<a href="#">Section 19.3.8</a>
24h	ICMDR	I2C Mode Register	<a href="#">Section 19.3.9</a>
28h	ICIVR	I2C Interrupt Vector Register	<a href="#">Section 19.3.10</a>
2Ch	ICEMDR	I2C Extended Mode Register	<a href="#">Section 19.3.11</a>
30h	ICPSC	I2C Prescaler Register	<a href="#">Section 19.3.12</a>
34h	REVID1	I2C Revision Identification Register 1	<a href="#">Section 19.3.13</a>
38h	REVID2	I2C Revision Identification Register 2	<a href="#">Section 19.3.13</a>
3Ch	ICDMAC	I2C DMA Control Register	<a href="#">Section 19.3.15</a>
48h	ICPFUNC	I2C Pin Function Register	<a href="#">Section 19.3.16</a>
4Ch	ICPDIR	I2C Pin Direction Register	<a href="#">Section 19.3.17</a>
50h	ICPDIN	I2C Pin Data In Register	<a href="#">Section 19.3.18</a>
54h	ICPDOUT	I2C Pin Data Out Register	<a href="#">Section 19.3.19</a>
58h	ICPDSET	I2C Pin Data Set Register	<a href="#">Section 19.3.20</a>
5Ch	ICPDCLR	I2C Pin Data Clear Register	<a href="#">Section 19.3.21</a>

### 19.3.1 I2C Own Address Register (ICOAR)

The I2C own address register (ICOAR) is used to specify its own slave address, which distinguishes it from other slaves connected to the I2C-bus. If the 7-bit addressing mode is selected (XA = 0 in ICMDR), only bits 6-0 are used; bits 9-7 are ignored. ICOAR is shown in [Figure 19-13](#) and described in [Table 19-5](#).

**Figure 19-13. I2C Own Address Register (ICOAR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

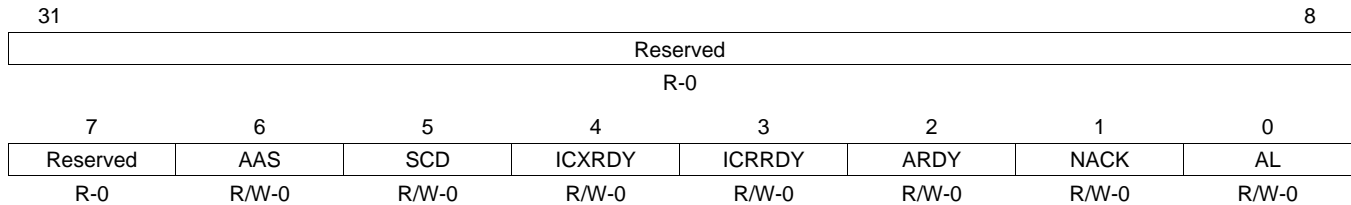
**Table 19-5. I2C Own Address Register (ICOAR) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	OADDR	0-3FFh	Own slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address of the I2C. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): bits 9-0 provide the 10-bit slave address of the I2C.

### 19.3.2 I2C Interrupt Mask Register (ICIMR)

The I2C interrupt mask register (ICIMR) is used to individually enable or disable I2C interrupt requests. ICIMR is shown in [Figure 19-14](#) and described [Table 19-6](#).

**Figure 19-14. I2C Interrupt Mask Register (ICIMR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-6. I2C Interrupt Mask Register (ICIMR) Field Descriptions**

Bit	Field	Value	Description
31-7	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
6	AAS	0 1	Address-as-slave interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
5	SCD	0 1	Stop condition detected interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
4	ICXRDY	0 1	Transmit-data-ready interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
3	ICRRDY	0 1	Receive-data-ready interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
2	ARDY	0 1	Register-access-ready interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
1	NACK	0 1	No-acknowledgment interrupt enable bit. Interrupt request is disabled. Interrupt request is enabled.
0	AL	0 1	Arbitration-lost interrupt enable bit Interrupt request is disabled. Interrupt request is enabled.

### 19.3.3 I2C Interrupt Status Register (ICSTR)

The I2C interrupt status register (ICSTR) is used to determine which interrupt has occurred and to read status information. ICSTR is shown in [Figure 19-15](#) and described in [Table 19-7](#).

**Figure 19-15. I2C Interrupt Status Register (ICSTR)**

Reserved							
R-0							
15	14	13	12	11	10	9	8
Reserved	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0	R/W1C-0	R/W1C-0	R/W1C-0	R-0	R-1	R-0	R-0
7	6	5	4	3	2	1	0
Reserved		SCD	ICXRDY	ICRRDY	ARDY	NACK	AL
R-0		R/W1C-0	R/W1C-1	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

**Table 19-7. I2C Interrupt Status Register (ICSTR) Field Descriptions**

Bit	Field	Value	Description
31-15	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
14	SDIR	0	Slave direction bit. In digital-loopback mode (DLB), the SDIR bit is cleared to 0. I2C is acting as a master-transmitter/receiver or a slave-receiver. SDIR is cleared by one of the following events:
		1	I2C is acting as a slave-transmitter.
13	NACKSNT	0	No-acknowledgment sent bit. NACKSNT bit is used when the I2C is in the receiver mode. One instance in which NACKSNT is affected is when the NACK mode is used (see the description for NACKMOD in <a href="#">Section 19.3.9</a> ). NACK is not sent. NACKSNT is cleared by one of the following events:
		1	NACK is sent. A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB	0	Bus busy bit. BB bit indicates whether the I2C-bus is busy or is free for another data transfer. In the master mode, BB is controlled by the software. Bus is free. BB is cleared by one of the following events:
		1	Bus is busy. When the STT bit in ICMDR is set to 1, a restart condition is generated. BB is set by one of the following events:
11	RSFULL	0	Receive shift register full bit. RSFULL indicates an overrun condition during reception. Overrun occurs when the receive shift register (ICRSR) is full with new data but the previous data has not been read from the data receive register (ICDRR). The new data will not be copied to ICDRR until the previous data is read. As new bits arrive from the I2Cx_SDA pin, they overwrite the bits in ICRSR. No overrun is detected. RSFULL is cleared by one of the following events:
		1	Overrun is detected.

**Table 19-7. I2C Interrupt Status Register (ICSTR) Field Descriptions (continued)**

Bit	Field	Value	Description
10	XSMT	0 1	<p>Transmit shift register empty bit. XSMT indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (ICXSR) is empty but the data transmit register (ICDXR) has not been loaded since the last ICDXR-to-ICXSR transfer. The next ICDXR-to-ICXSR transfer will not occur until new data is in ICDXR. If new data is not transferred in time, the previous data may be re-transmitted on the I2Cx_SDA pin.</p> <p>0 Underflow is detected.</p> <p>1 No underflow is detected. XSMT is set by one of the following events:</p> <ul style="list-style-type: none"> <li>Data is written to ICDXR.</li> <li>The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).</li> </ul>
9	AAS	0 1	<p>Addressed-as-slave bit.</p> <p>0 The AAS bit has been cleared by a repeated START condition or by a STOP condition.</p> <p>1 AAS is set by one of the following events:</p> <ul style="list-style-type: none"> <li>I2C has recognized its own slave address or an address of all zeros (general call).</li> <li>The first data word has been received in the free data format (FDF = 1 in ICMDR).</li> </ul>
8	AD0	0 1	<p>Address 0 bit.</p> <p>0 AD0 has been cleared by a START or STOP condition.</p> <p>1 An address of all zeros (general call) is detected.</p>
7-6	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
5	SCD	0 1	<p>Stop condition detected bit. SCD indicates when a STOP condition has been detected on the I2C bus. The STOP condition could be generated by the I2C or by another I2C device connected to the bus.</p> <p>0 No STOP condition has been detected. SCD is cleared by one of the following events:</p> <ul style="list-style-type: none"> <li>By reading the INTCODE bits in ICIVR as 110b.</li> <li>SCD is manually cleared. To clear this bit, write a 1 to it.</li> </ul> <p>1 A STOP condition has been detected.</p>
4	ICXRDY	0 1	<p>Transmit-data-ready interrupt flag bit. ICXRDY indicates that the data transmit register (ICDXR) is ready to accept new data because the previous data has been copied from ICDXR to the transmit shift register (ICXSR). The CPU can poll ICXRDY or use the XRDY interrupt request.</p> <p>0 ICDXR is not ready. ICXRDY is cleared by one of the following events:</p> <ul style="list-style-type: none"> <li>Data is written to ICDXR.</li> <li>ICXRDY is manually cleared. To clear this bit, write a 1 to it.</li> </ul> <p>1 ICDXR is ready. Data has been copied from ICDXR to ICXSR. ICXRDY is forced to 1 when the I2C is reset.</p>
3	ICRRDY	0 1	<p>Receive-data-ready interrupt flag bit. ICRRDY indicates that the data receive register (ICDRR) is ready to be read because data has been copied from the receive shift register (ICRSR) to ICDRR. The CPU can poll ICRRDY or use the RRDY interrupt request.</p> <p>0 ICDRR is not ready. ICRRDY is cleared by one of the following events:</p> <ul style="list-style-type: none"> <li>ICDRR is read.</li> <li>ICRRDY is manually cleared. To clear this bit, write a 1 to it.</li> <li>The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).</li> </ul> <p>1 ICDRR is ready. Data has been copied from ICRSR to ICDRR.</p>
2	ARDY	0 1	<p>Register-access-ready interrupt flag bit (only applicable when the I2C is in the master mode). ARDY indicates that the I2C registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request.</p> <p>0 The registers are not ready to be accessed. ARDY is cleared by one of the following events:</p> <ul style="list-style-type: none"> <li>The I2C starts using the current register contents.</li> <li>ARDY is manually cleared. To clear this bit, write a 1 to it.</li> <li>The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).</li> </ul> <p>1 The registers are ready to be accessed. This bit is set after the slave address appears on the I2C bus.</p> <ul style="list-style-type: none"> <li>In the nonrepeat mode (RM = 0 in ICMDR): If STP = 0 in ICMDR, ARDY is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C generates a STOP condition when the counter reaches 0).</li> <li>In the repeat mode (RM = 1): ARDY is set at the end of each data word transmitted from ICDXR.</li> </ul>

**Table 19-7. I2C Interrupt Status Register (ICSTR) Field Descriptions (continued)**

Bit	Field	Value	Description
1	NACK	<p>0</p> <p>1</p>	<p>No-acknowledgment interrupt flag bit. NACK applies when the I2C is a transmitter (master or slave). NACK indicates whether the I2C has detected an acknowledge bit (ACK) or a no-acknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request.</p> <p>ACK received/NACK is not received. NACK is cleared by one of the following events:</p> <ul style="list-style-type: none"> <li>• An acknowledge bit (ACK) has been sent by the receiver.</li> <li>• NACK is manually cleared. To clear this bit, write a 1 to it.</li> <li>• The CPU reads the interrupt vector register (ICIVR) when the register contains the code for a NACK interrupt.</li> <li>• The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).</li> </ul> <p>NACK bit is received. The hardware detects that a no-acknowledge (NACK) bit has been received.  <b>Note:</b> While the I2C performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment.</p>
0	AL	<p>0</p> <p>1</p>	<p>Arbitration-lost interrupt flag bit (only applicable when the I2C is a master-transmitter). AL primarily indicates when the I2C has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request.</p> <p>Arbitration is not lost. AL is cleared by one of the following events:</p> <ul style="list-style-type: none"> <li>• AL is manually cleared. To clear this bit, write a 1 to it.</li> <li>• The CPU reads the interrupt vector register (ICIVR) when the register contains the code for an AL interrupt.</li> <li>• The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).</li> </ul> <p>Arbitration is lost. AL is set by one of the following events:</p> <ul style="list-style-type: none"> <li>• The I2C senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously.</li> <li>• The I2C attempts to start a transfer while the BB (bus busy) bit is set to 1.</li> </ul> <p>When AL is set to 1, the MST and STP bits of ICMDR are cleared, and the I2C becomes a slave-receiver.</p>

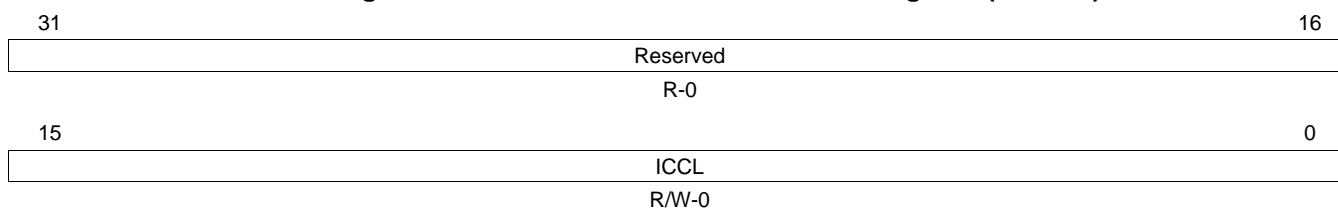
### 19.3.4 I2C Clock Divider Registers (ICCLKL and ICCLKH)

When the I2C is a master, the prescaled module clock is divided down for use as the I2C serial clock on the I2Cx\_SCL pin. The shape of the I2C serial clock depends on two divide-down values, ICCL and ICCH. For detailed information on how these values are programmed, see [Section 19.2.2](#).

#### 19.3.4.1 I2C Clock Low-Time Divider Register (ICCLKL)

For each I2C serial clock cycle, ICCL in the I2C clock low-time divider register (ICCLKL) determines the amount of time the signal is low. ICCLKL must be configured while the I2C is still in reset (IRS = 0 in ICMDR). ICCLKL is shown in [Figure 19-16](#) and described in [Table 19-8](#).

**Figure 19-16. I2C Clock Low-Time Divider Register (ICCLKL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

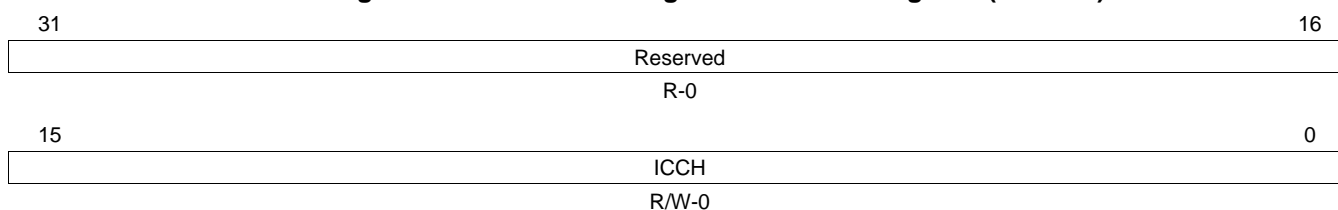
**Table 19-8. I2C Clock Low-Time Divider Register (ICCLKL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICCL	0-FFFFh	Clock low-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCL + d) to produce the low-time duration of the I2C serial on the I2Cx_SCL pin.

#### 19.3.4.2 I2C Clock High-Time Divider Register (ICCLKH)

For each I2C serial clock cycle, ICCH in the I2C clock high-time divider register (ICCLKH) determines the amount of time the signal is high. ICCLKH must be configured while the I2C is still in reset (IRS = 0 in ICMDR). ICCLKH is shown in [Figure 19-17](#) and described in [Table 19-9](#).

**Figure 19-17. I2C Clock High-Time Divider Register (ICCLKH)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-9. I2C Clock High-Time Divider Register (ICCLKH) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICCH	0-FFFFh	Clock high-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCH + d) to produce the high-time duration of the I2C serial on the I2Cx_SCL pin.

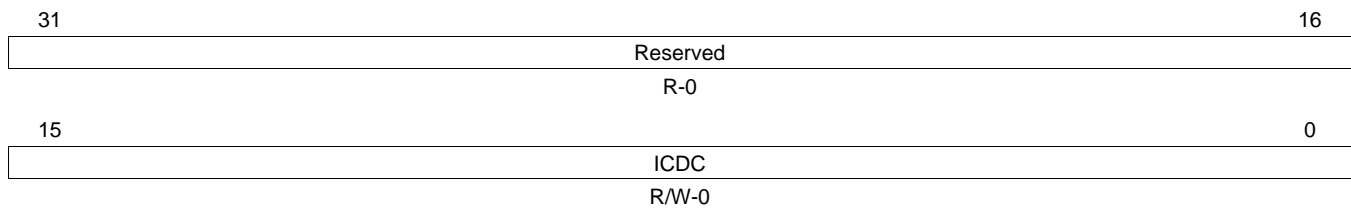
### 19.3.5 I2C Data Count Register (ICCNT)

The I2C data count register (ICCNT) is used to indicate how many data words to transfer when the I2C is configured as a master-transmitter-receiver (MST = 1 and TRX = 1/0 in ICMDR) and the repeat mode is off (RM = 0 in ICMDR). In the repeat mode (RM = 1), ICCNT is not used.

The value written to ICCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each data word transferred (ICCNT remains unchanged). If a STOP condition is requested (STP = 1 in ICMDR), the I2C terminates the transfer with a STOP condition when the countdown is complete (that is, when the last data word has been transferred).

ICCNT is shown in [Figure 19-18](#) and described in [Table 19-10](#).

**Figure 19-18. I2C Data Count Register (ICCNT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-10. I2C Data Count Register (ICCNT) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICDC	0-FFFFh	Data count value. When RM = 0 in ICMDR, ICDC indicates the number of data words to transfer in the nonrepeat mode. When RM = 1 in ICMDR, the value in ICCNT is a don't care. If STP = 1 in ICMDR, a STOP condition is generated when the internal data counter counts down to 0.
		0	The start value loaded to the internal data counter is 65536.
		1h-FFFFh	The start value loaded to internal data counter is 1-65535.

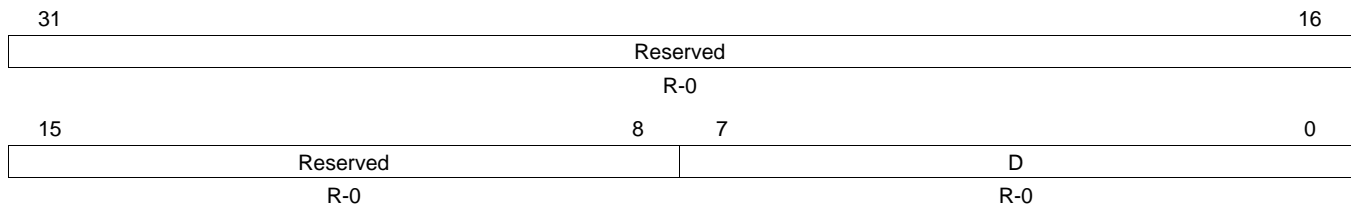


### 19.3.6 I2C Data Receive Register (ICDRR)

The I2C data receive register (ICDRR) is used to read the receive data. The ICDRR can receive a data value of up to 8 bits; data values with fewer than 8 bits are right-aligned in the D bits and the remaining D bits are undefined. The number of data bits is selected by the bit count bits (BC) of ICMDR. The I2C receive shift register (ICRSR) shifts in the received data from the I2Cx\_SDA pin. Once data is complete, the I2C copies the contents of ICRSR into ICDRR. The CPU and the EDMA controller cannot access ICRSR.

ICDRR is shown in [Figure 19-19](#) and described in [Table 19-11](#).

**Figure 19-19. I2C Data Receive Register (ICDRR)**



LEGEND: R = Read only; -n = value after reset

**Table 19-11. I2C Data Receive Register (ICDRR) Field Descriptions**

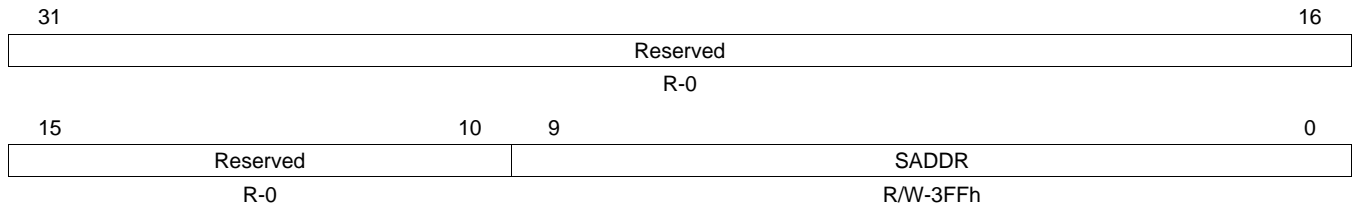
Bit	Field	Value	Description
31-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	D	0-FFh	Receive data.

### 19.3.7 I2C Slave Address Register (ICSAR)

The I2C slave address register (ICSAR) contains a 7-bit or 10-bit slave address. When the I2C is not using the free data format (FDF = 0 in ICMDR), it uses this address to initiate data transfers with a slave or slaves. When the address is nonzero, the address is for a particular slave. When the address is 0, the address is a general call to all slaves. If the 7-bit addressing mode is selected (XA = 0 in ICMDR), only bits 6-0 of ICSAR are used; bits 9-7 are ignored.

ICSAR is shown in [Figure 19-20](#) and described in [Table 19-12](#).

**Figure 19-20. I2C Slave Address Register (ICSAR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-12. I2C Slave Address Register (ICSAR) Field Descriptions**

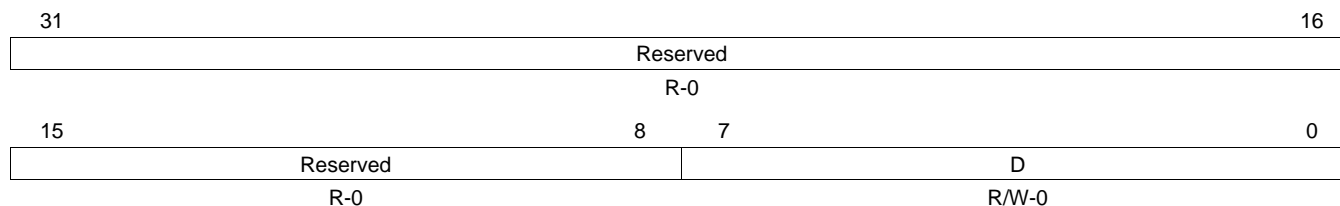
Bit	Field	Value	Description
31-10	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	SADDR	0-3FFh	Slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address that the I2C transmits when it is in the master-transmitter mode. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): Bits 9-0 provide the 10-bit slave address that the I2C transmits when it is in the master-transmitter mode.

### 19.3.8 I2C Data Transmit Register (ICDXR)

The CPU or EDMA writes transmit data to the I2C data transmit register (ICDXR). The ICDXR can accept a data value of up to 8 bits. When writing a data value with fewer than 8 bits, the written data must be right-aligned in the D bits. The number of data bits is selected by the bit count bits (BC) of ICMR. Once data is written to ICDXR, the I2C copies the contents of ICDXR into the I2C transmit shift register (ICXSR). The ICXSR shifts out the transmit data from the I2Cx\_SDA pin. The CPU and the EDMA controller cannot access ICXSR.

ICDXR is shown in [Figure 19-21](#) and described in [Table 19-13](#).

**Figure 19-21. I2C Data Transmit Register (ICDXR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-13. I2C Data Transmit Register (ICDXR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	D	0-FFh	Transmit data.

### 19.3.9 I2C Mode Register (ICMDR)

The I2C mode register (ICMDR) contains the control bits of the I2C. ICMDR is shown in shown in [Figure 19-22](#) and described in [Table 19-14](#).

**Figure 19-22. I2C Mode Register (ICMDR)**

Reserved							
R-0							
15	14	13	12	11	10	9	8
NACKMOD	FREE	STT	Reserved	STP	MST	TRX	XA
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	0	
RM	DLB	IRS	STB	FDF	BC		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-14. I2C Mode Register (ICMDR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15	NACKMOD	0	No-acknowledge (NACK) mode bit (only applicable when the I2C is a receiver). In slave-receiver mode: The I2C sends an acknowledge (ACK) bit to the transmitter during the each acknowledge cycle on the bus. The I2C only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit. In master-receiver mode: The I2C sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. When the counter reaches 0, the I2C sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you must set the NACKMOD bit.
		1	In either slave-receiver or master-receiver mode: The I2C sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared. To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.
14	FREE	0	This emulation mode bit is used to determine the state of the I2C when a breakpoint is encountered in the high-level language debugger. When I2C is master: If I2Cx_SCL is low when the breakpoint occurs, the I2C stops immediately and keeps driving I2Cx_SCL low, whether the I2C is the transmitter or the receiver. If I2Cx_SCL is high, the I2C waits until I2Cx_SCL becomes low and then stops. When I2C is slave: A breakpoint forces the I2C to stop when the current transmission/reception is complete.
		1	The I2C runs free; that is, it continues to operate when a breakpoint occurs.
13	STT	0	START condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see <a href="#">Table 19-15</a> ). Note that the STT and STP bits can be used to terminate the repeat mode. In master mode, STT is automatically cleared after the START condition has been generated. In slave mode, if STT is 0, the I2C does not monitor the bus for commands from a master. As a result, the I2C performs no data transfers.
		1	In master mode, setting STT to 1 causes the I2C to generate a START condition on the I2C-bus. In slave mode, if STT is 1, the I2C monitors the bus and transmits/receives data in response to commands from a master.
12	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
11	STP	0	STOP condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see <a href="#">Table 19-15</a> ). Note that the STT and STP bits can be used to terminate the repeat mode. STP is automatically cleared after the STOP condition has been generated.
		1	STP has been set to generate a STOP condition when the internal data counter of the I2C counts down to 0.

**Table 19-14. I2C Mode Register (ICMDR) Field Descriptions (continued)**

Bit	Field	Value	Description
10	MST	0 1	<p>Master mode bit. MST determines whether the I2C is in the slave mode or the master mode. MST is automatically changed from 1 to 0 when the I2C master generates a STOP condition. See <a href="#">Table 19-16</a>.</p> <p>0 Slave mode. The I2C is a slave and receives the serial clock from the master.</p> <p>1 Master mode. The I2C is a master and generates the serial clock on the I2Cx_SCL pin.</p>
9	TRX	0 1	<p>Transmitter mode bit. When relevant, TRX selects whether the I2C is in the transmitter mode or the receiver mode. <a href="#">Table 19-16</a> summarizes when TRX is used and when it is a don't care.</p> <p>0 Receiver mode. The I2C is a receiver and receives data on the I2Cx_SDA pin.</p> <p>1 Transmitter mode. The I2C is a transmitter and transmits data on the I2Cx_SDA pin.</p>
8	XA	0 1	<p>Expanded address enable bit.</p> <p>0 7-bit addressing mode (normal address mode). The I2C transmits 7-bit slave addresses (from bits 6-0 of ICSAR), and its own slave address has 7 bits (bits 6-0 of ICOAR).</p> <p>1 10-bit addressing mode (expanded address mode). The I2C transmits 10-bit slave addresses (from bits 9-0 of ICSAR), and its own slave address has 10 bits (bits 9-0 of ICOAR).</p>
7	RM	0 1	<p>Repeat mode bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see <a href="#">Table 19-15</a>). If the I2C is configured in slave mode, the RM bit is don't care.</p> <p>0 Nonrepeat mode. The value in the data count register (ICCNT) determines how many data words are received/transmitted by the I2C.</p> <p>1 Repeat mode. Data words are continuously received/transmitted by the I2C until the STP bit is manually set to 1, regardless of the value in ICCNT.</p>
6	DLB	0 1	<p>Digital loopback mode bit (only applicable when the I2C is a master-transmitter). This bit disables or enables the digital loopback mode of the I2C. The effects of this bit are shown in <a href="#">Figure 19-23</a>. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported.</p> <p>0 Digital loopback mode is disabled.</p> <p>1 Digital loopback mode is enabled. In this mode, the MST bit must be set to 1 and data transmitted out of ICDXR is received in ICDRR after n clock cycles by an internal path, where:</p> $n = ((I2C \text{ input clock frequency} / \text{prescaled module clock frequency}) \times 8)$ <p>The transmit clock is also the receive clock. The address transmitted on the I2Cx_SDA pin is the address in ICOAR.</p>
5	IRS	0 1	<p>I2C reset bit. Note that if IRS is reset during a transfer, it can cause the I2C bus to hang (I2Cx_SDA and I2Cx_SCL are in a high-impedance state).</p> <p>0 The I2C is in reset/disabled. When this bit is cleared to 0, all status bits (in ICSTR) are set to their default values.</p> <p>1 The I2C is enabled.</p>
4	STB	0 1	<p>START byte mode bit (only applicable when the I2C is a master). As described in version 2.1 of the Philips I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C is a slave, the I2C ignores a START byte from a master, regardless of the value of the STB bit.</p> <p>0 The I2C is not in the START byte mode.</p> <p>1 The I2C is in the START byte mode. When you set the START condition bit (STT), the I2C begins the transfer with more than just a START condition. Specifically, it generates:</p> <ol style="list-style-type: none"> <li>1. A START condition</li> <li>2. A START byte (0000 0001b)</li> <li>3. A dummy acknowledge clock pulse</li> <li>4. A repeated START condition</li> </ol> <p>The I2C sends the slave address that is in ICSAR.</p>
3	FDF	0 1	<p>Free data format mode bit. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported. See <a href="#">Table 19-16</a>.</p> <p>0 Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.</p> <p>1 Free data format mode is enabled.</p>

**Table 19-14. I2C Mode Register (ICMDR) Field Descriptions (continued)**

Bit	Field	Value	Description
2-0	BC	0-7h	Bit count bits. BC defines the number of bits (1 to 8) in the next data word that is to be received or transmitted by the I2C. The number of bits selected with BC must match the data size of the other device. Note that when BC = 0, a data word has 8 bits.  If the bit count is less than 8, receive data is right aligned in the D bits of ICRR and the remaining D bits are undefined. Also, transmit data written to ICXR must be right aligned.
		0	8 bits per data word
		1h	1 bit per data word
		2h	2 bits per data word
		3h	3 bits per data word
		4h	4 bits per data word
		5h	5 bits per data word
		6h	6 bits per data word
		7h	7 bits per data word

**Table 19-15. Master-Transmitter/Receiver Bus Activity Defined by RM, STT, and STP Bits**

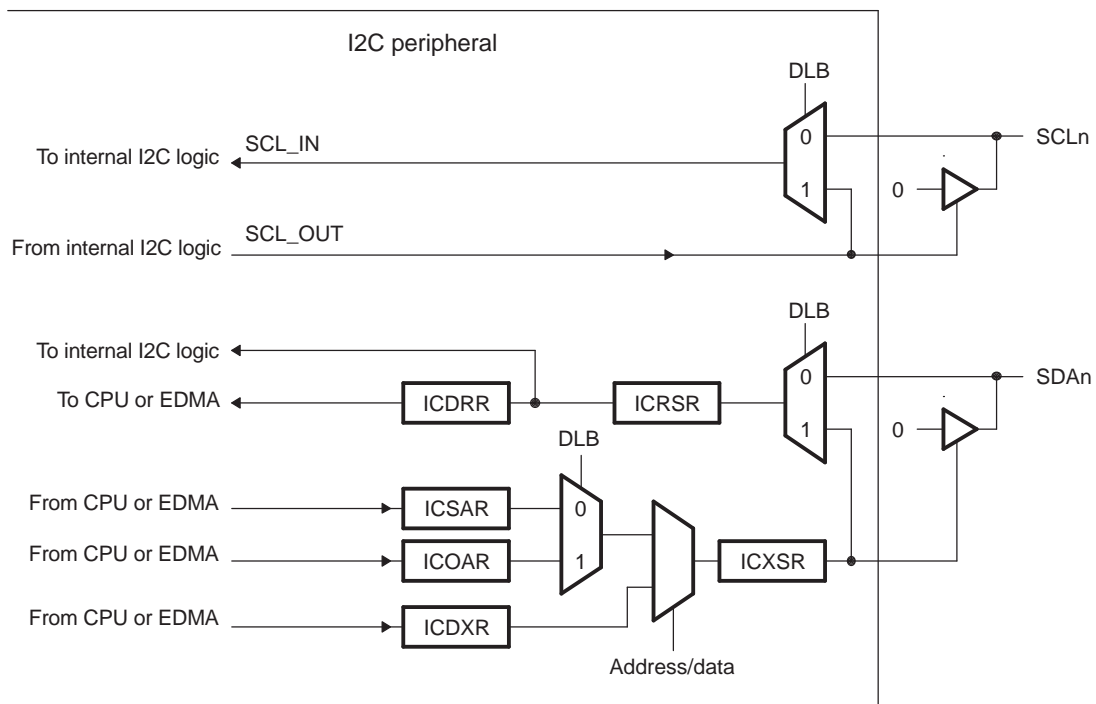
ICMDR Bit			Bus Activity <sup>(1)</sup>	Description
RM	STT	STP		
0	0	0	None	No activity
0	0	1	P	STOP condition
0	1	0	S-A-D..(n)..D	START condition, slave address, <i>n</i> data words ( <i>n</i> = value in ICCNT)
0	1	1	S-A-D..(n)..D-P	START condition, slave address, <i>n</i> data words, STOP condition ( <i>n</i> = value in ICCNT)
1	0	0	None	No activity
1	0	1	P	STOP condition
1	1	0	S-A-D-D-D..	Repeat mode transfer: START condition, slave address, continuous data transfers until STOP condition or next START condition
1	1	1	None	Reserved bit combination (No activity)

<sup>(1)</sup> A = Address; D = Data word; P = STOP condition; S = START condition

**Table 19-16. How the MST and FDF Bits Affect the Role of TRX Bit**

ICMDR Bit			I2C State	Function of TRX Bit
MST	FDF			
0	0		In slave mode but not free data format mode	TRX is a don't care. Depending on the command from the master, the I2C responds as a receiver or a transmitter.
0	1		In slave mode and free data format mode	The free data format mode requires that the transmitter and receiver be fixed. TRX identifies the role of the I2C:  TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.
1	0		In master mode but not free data format mode	TRX identifies the role of the I2C:  TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.
1	1		In master mode and free data format mode	The free data format mode requires that the transmitter and receiver be fixed. TRX identifies the role of the I2C:  TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.

**Figure 19-23. Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit**

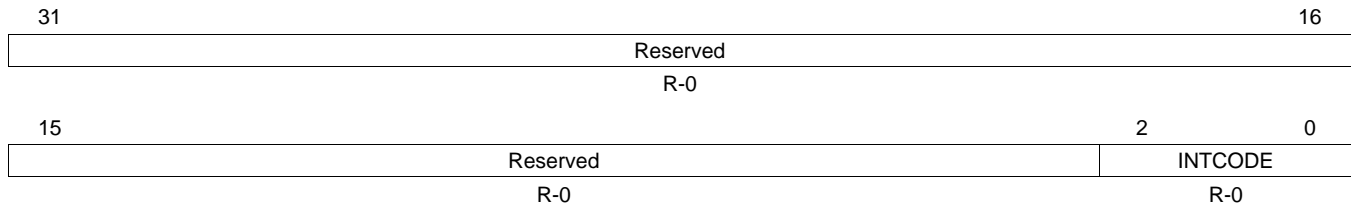


### 19.3.10 I2C Interrupt Vector Register (ICIVR)

The I2C interrupt vector register (ICIVR) is used by the CPU to determine which event generated the I2C interrupt. Reading ICIVR clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there are more than one interrupt flag, reading ICIVR clears the highest priority interrupt flag. Note that you must read (clear) ICIVR before doing another start; otherwise, ICIVR could contain an incorrect (old interrupt flags) value.

ICIVR is shown in [Figure 19-24](#) and described in [Table 19-17](#).

**Figure 19-24. I2C Interrupt Vector Register (ICIVR)**



LEGEND: R= Read only; -n = value after reset

**Table 19-17. I2C Interrupt Vector Register (ICIVR) Field Descriptions**

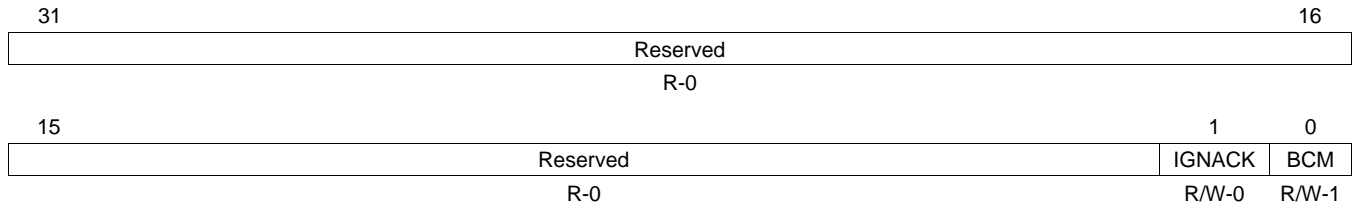
Bit	Field	Value	Description
31-3	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
2-0	INTCODE	0-7h	Interrupt code bits. The binary code in INTCODE indicates which event generated an I2C interrupt.
		0	None
		1h	Arbitration-lost interrupt (AL). Highest priority if multiple I2C interrupts are pending.
		2h	No-acknowledgment interrupt (NACK)
		3h	Register-access-ready interrupt (ARDY)
		4h	Receive-data-ready interrupt (ICRRDY)
		5h	Transmit-data-ready interrupt (ICXRDY)
		6h	Stop condition detected interrupt (SCD)
		7h	Address-as-slave interrupt (AAS). Lowest priority if multiple I2C interrupts are pending.



### 19.3.11 I2C Extended Mode Register (ICEMDR)

The I2C extended mode register (ICEMDR) is used to indicate which condition generates a transmit data ready interrupt. ICEMDR is shown in [Figure 19-25](#) and described in [Table 19-18](#).

**Figure 19-25. I2C Extended Mode Register (ICEMDR)**



LEGEND: R/W = Read/Write; R= Read only; -n = value after reset

**Table 19-18. I2C Extended Mode Register (ICEMDR) Field Descriptions**

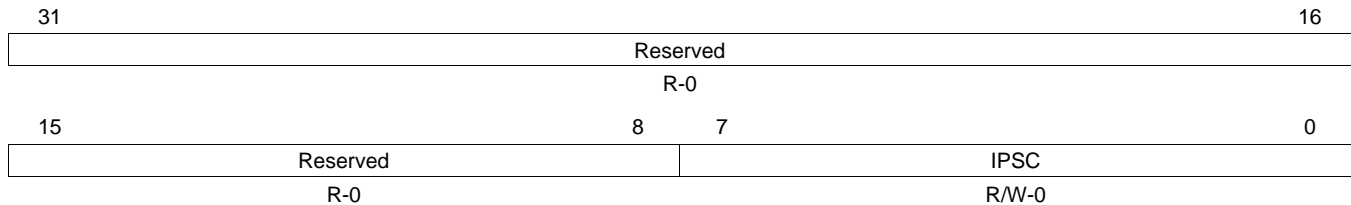
Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	IGNACK	0	Master transmitter operates normally, that is, it discontinues the data transfer and sets the ARDY and NACK bits in ICSTR when receiving a NACK from the slave.
		1	Master transmitter ignores a NACK from the slave.
0	BCM		Backward compatibility mode bit. Determines which condition generates a transmit data ready interrupt. The BCM bit only has an effect when the I2C is operating as a slave-transmitter.
		0	The transmit data ready interrupt is generated when the master requests more data by sending an acknowledge signal after the transmission of the last data.
		1	The transmit data ready interrupt is generated when the data in ICDXR is copied to ICXSR.

### 19.3.12 I2C Prescaler Register (ICPSC)

The I2C prescaler register (ICPSC) is used for dividing down the I2C input clock to obtain the desired prescaled module clock for the operation of the I2C. The IPSC bits must be initialized while the I2C is in reset (IRS = 0 in ICMDR). The prescaled frequency takes effect only when the IRS bit is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

ICPSC is shown in [Figure 19-26](#) and described in [Table 19-19](#).

**Figure 19-26. I2C Prescaler Register (ICPSC)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

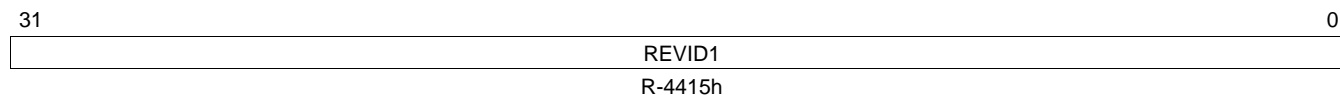
**Table 19-19. I2C Prescaler Register (ICPSC) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	IPSC	0-FFh	I2C prescaler divide-down value. IPSC determines how much the I2C input clock is divided to create the I2C prescaled module clock: $I2C \text{ clock frequency} = I2C \text{ input clock frequency} / (IPSC + 1)$ <b>Note:</b> IPSC must be initialized while the I2C is in reset (IRS = 0 in ICMDR).

### 19.3.13 I2C Revision Identification Register (REVID1)

The I2C revision identification register (REVID1) contains identification data for the peripheral. REVID1 is shown in [Figure 19-27](#) and described in [Table 19-20](#).

**Figure 19-27. I2C Revision Identification Register 1 (REVID1)**



LEGEND: R = Read only; -n = value after reset

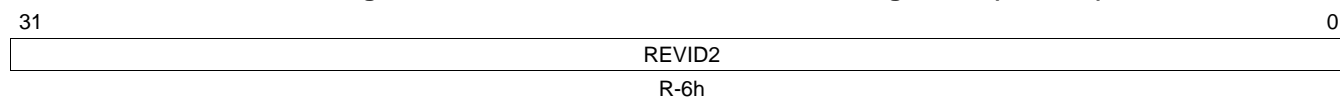
**Table 19-20. I2C Revision Identification Register 1 (REVID1) Field Descriptions**

Bit	Field	Value	Description
31-0	REVID1	4415h	Peripheral Identification Number

### 19.3.14 I2C Revision Identification Register (REVID2)

The I2C revision identification register (REVID2) contains identification data for the peripheral. REVID2 is shown in [Figure 19-28](#) and described in [Table 19-21](#).

**Figure 19-28. I2C Revision Identification Register 2 (REVID2)**



LEGEND: R = Read only; -n = value after reset

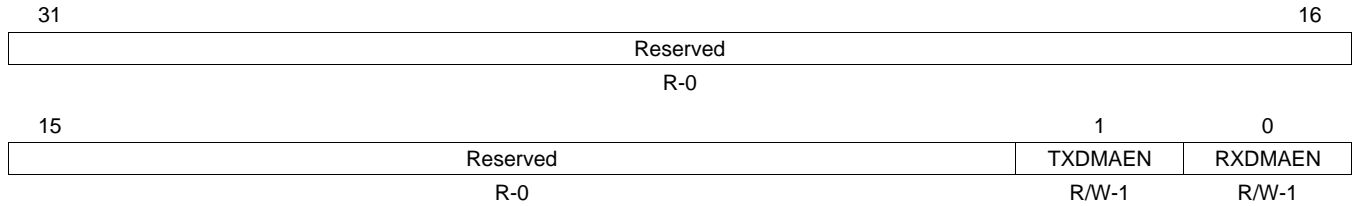
**Table 19-21. I2C Revision Identification Register 2 (REVID2) Field Descriptions**

Bit	Field	Value	Description
31-0	REVID2	6h	Peripheral Identification Number

### 19.3.15 I2C DMA Control Register (ICDMAC)

The I2C DMA control register (ICDMAC) is used to control the transmit DMA event and receive DMA event pin to the system . ICDMAC is shown in [Figure 19-29](#) and described in [Table 19-22](#).

**Figure 19-29. I2C DMA Control Register (ICDMAC)**



LEGEND: R/W = Read/Write; R= Read only; -n = value after reset

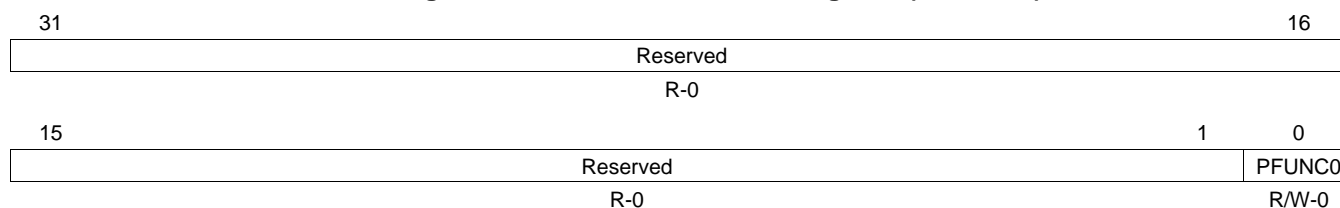
**Table 19-22. I2C DMA Control Register (ICDMAC) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	TXDMAEN	0	Transmit DMA enable. This bit controls the transmit DMA event pin to the system. Always set this bit to 1. DMA transmit event is disabled.
		1	DMA transmit event is enabled.
0	RXDMAEN	0	Receive DMA enable . This bit controls the receive DMA event pin to the system. Always set this bit to 1. DMA receive event is disabled.
		1	DMA receive event is enabled.

### 19.3.16 I2C Pin Function Register (ICPFUNC)

The I2C pin function register (ICPFUNC) is used to configure the external I2C pins (I2Cx\_SDA and I2Cx\_SCL) as a I2C peripheral pin or a GPIO pin. ICPFUNC is shown in [Figure 19-30](#) and described in [Table 19-23](#).

**Figure 19-30. I2C Pin Function Register (ICPFUNC)**



LEGEND: R/W = Read/Write; R= Read only; -n = value after reset

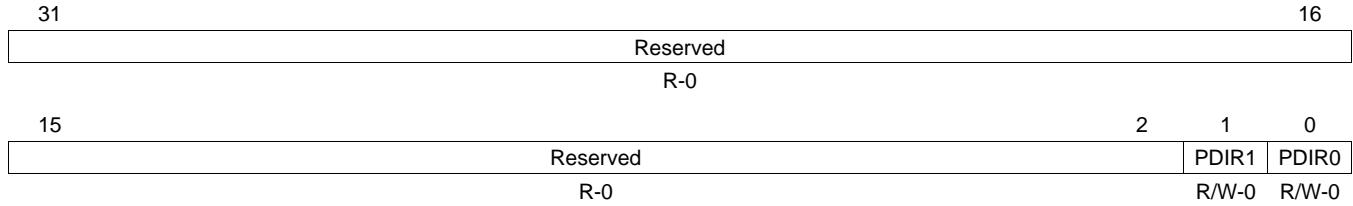
**Table 19-23. I2C Pin Function Register (ICPFUNC) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
0	PFUNC0	0 1	Controls the function of the I2Cx_SCL and I2Cx_SDA pins. 0 Pins function as I2Cx_SCL and I2Cx_SDA. 1 Pins function as GPIO.  Note: No hardware protection is required to disable the I2C function when the PFUNC0 bit and the IRS bit in the I2C mode register (ICMDR) are both set to 1. When PFUNC0 = 1 (GPIO mode), the submodule that controls the I2C function receives the value 1 for I2Cx_SCL and I2Cx_SDA. The IRS bit can be set to 1 regardless of PFUNC0, and the I2C function works whenever the IRS bit is 1. You are expected to hold I2C in reset via the IRS bit when changing to/from GPIO mode via the PFUNC0 bit.

### 19.3.17 I2C Pin Direction Register (ICPDIR)

The I2C pin direction register (ICPDIR) is used to configure each GPIO pin as either an input or an output. ICPDIR is shown in [Figure 19-31](#) and described in [Table 19-24](#).

**Figure 19-31. I2C Pin Direction Register (ICPDIR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

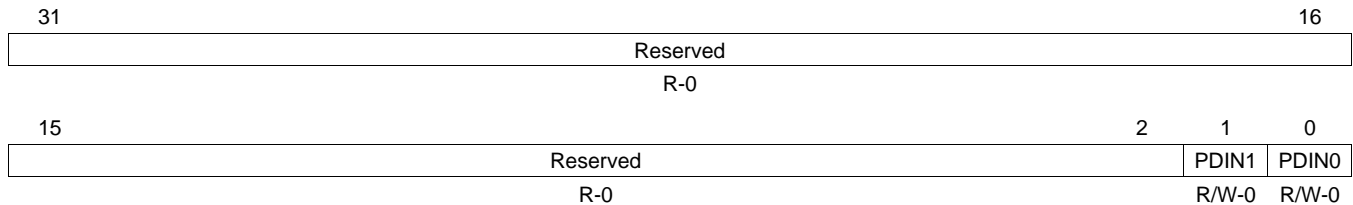
**Table 19-24. I2C Pin Direction Register (ICPDIR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDIR1	0 1	Controls the direction of the I2Cx_SDA pin when configured as GPIO. I2Cx_SDA pin functions as input. I2Cx_SDA pin functions as output.
0	PDIR0	0 1	Controls the direction of the I2Cx_SCL pin when configured as GPIO. I2Cx_SCL pin functions as input. I2Cx_SCL pin functions as output.

### 19.3.18 I2C Pin Data In Register (ICPDIN)

The I2C pin data in register (ICPDIN) holds the I/O state of each of the I2C pins (I2Cx\_SDA and I2Cx\_SCL); and should return the value from the pin's input buffer (with appropriate synchronization/DFT considerations). However, this register allows the actual value of the pin to be read regardless of the state of PFUNC or PDIR bits . ICPDIN is shown in [Figure 19-32](#) and described in [Table 19-25](#).

**Figure 19-32. I2C Pin Data In Register (ICPDIN)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-25. I2C Pin Data In Register (ICPDIN) Field Descriptions**

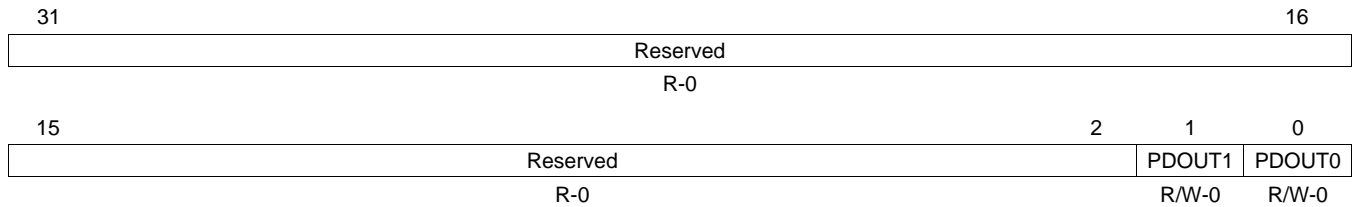
Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDIN1	0 1	Indicates the logic level present on the I2Cx_SDA pin. <b>During reads:</b> 0 Logic-low present at I2Cx_SDA pin, regardless of PFUNC bit setting. 1 Logic-high present at I2Cx_SDA pin, regardless of PFUNC bit setting. <b>During writes:</b> Writes have no effect.
0	PDIN0	0 1	Indicates the logic level present on the I2Cx_SCL pin. <b>During reads:</b> 0 Logic-low present at I2Cx_SCL pin, regardless of PFUNC bit setting. 1 Logic-high present at I2Cx_SCL pin, regardless of PFUNC bit setting. <b>During writes:</b> Writes have no effect.

### 19.3.19 I2C Pin Data Out Register (ICPDOUT)

The I2C pin data out register (ICPDOUT) has one bit for each of the GPIO pins. This bit holds a value for data out at all times, and may be read back at all times. The value held by this register is not affected by writing to the PDIR and PFUNC bits. However, the data value in this register is driven out onto the GPIO pin only if the PFUNC0 bit in ICPFUNC is set to 1 (I2Cx\_SDA and I2Cx\_SCL function as GPIO) and also the corresponding bit in ICPDIR is set to 1 (output).

ICPDOUT is shown in [Figure 19-33](#) and described in [Table 19-26](#).

**Figure 19-33. I2C Pin Data Out Register (ICPDOUT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-26. I2C Pin Data Out Register (ICPDOUT) Field Descriptions**

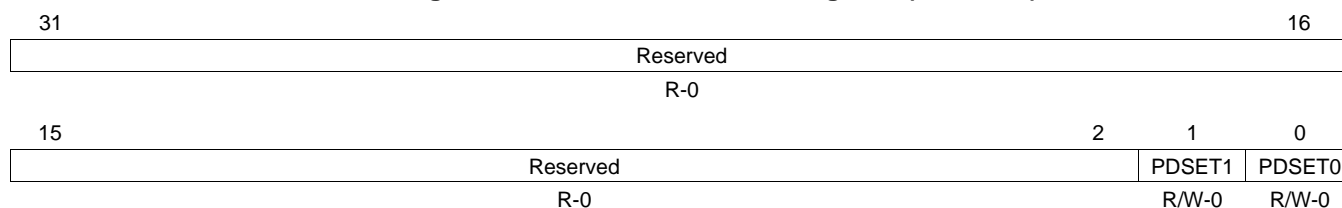
Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDOUT1	0	Controls the level driven on the I2Cx_SDA pin when configured as GPIO output. Note: If I2Cx_SDA is connected to an open-drain buffer at the chip level, the I2C cannot drive I2Cx_SDA to high. <b>During reads:</b> Reads return register values, not GPIO pin levels.
		1	<b>During writes:</b> I2Cx_SDA pin is driven low.
		1	I2Cx_SDA pin is driven high.
0	PDOUT0	0	Controls the level driven on the I2Cx_SCL pin when configured as GPIO output. Note: If I2Cx_SCL is connected to an open-drain buffer at the chip level, the I2C cannot drive I2Cx_SCL to high. <b>During reads:</b> Reads return register values, not GPIO pin levels.
		0	<b>During writes:</b> I2Cx_SCL pin is driven low.
		1	I2Cx_SCL pin is driven high.



### 19.3.20 I2C Pin Data Set Register (ICPDSET)

The I2C pin data set register (ICPDSET) is an alias of the I2C pin data out register (ICPDOUT). Writing a 1 to a bit in ICPDSET sets the corresponding bit in ICPDOUT to a 1, while writing a 0 keeps the bit unchanged. ICPDSET is shown in [Figure 19-34](#) and described in [Table 19-27](#).

**Figure 19-34. I2C Pin Data Set Register (ICPDSET)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

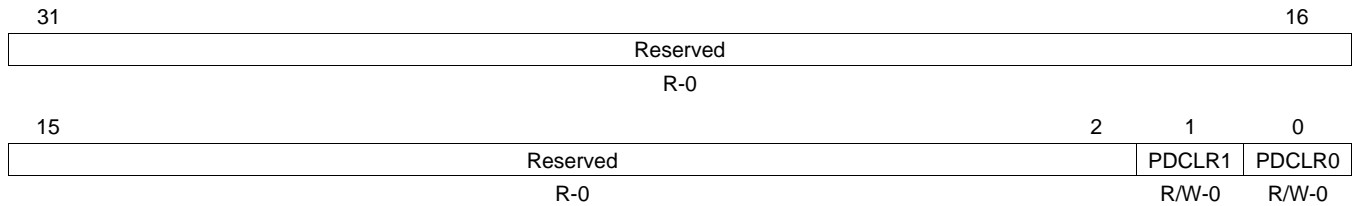
**Table 19-27. I2C Pin Data Set Register (ICPDSET) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDSET1	0	Used to set the PDOUT1 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SDA GPIO pin. <b>During reads:</b> Reads return indeterminate values.
		1	<b>During writes:</b> No effect PDOUT1 bit is set to logic high.
0	PDSET0	0	Used to set the PDOUT0 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SCL GPIO pin. <b>During reads:</b> Reads return indeterminate values.
		1	<b>During writes:</b> No effect PDOUT0 bit is set to logic high.

### 19.3.21 I2C Pin Data Clear Register (ICPDCLR)

The I2C pin data clear register (ICPDCLR) is an alias of the I2C pin data out register (ICPDOUT). Writing a 1 to a bit in ICPDCLR clears the corresponding bit in ICPDOUT to a 0, while writing a 0 keeps the bit unchanged. ICPDCLR is shown in [Figure 19-35](#) and described in [Table 19-28](#).

**Figure 19-35. I2C Pin Data Clear Register (ICPDCLR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-28. I2C Pin Data Clear Register (ICPDCLR) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDCLR1		Used to clear the PDOUT1 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SDA GPIO pin.
			<b>During reads:</b> Reads return indeterminate values.
			<b>During writes:</b>
		0	No effect
		1	PDOUT1 bit is cleared to logic low.
0	PDCLR0		Used to clear the PDOUT0 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SCL GPIO pin.
			<b>During reads:</b> Reads return indeterminate values.
			<b>During writes:</b>
		0	No effect
		1	PDOUT0 bit is cleared to logic low.

---

---

## ***Multichannel Audio Serial Port (McASP)***

---

---

This chapter describes the multichannel audio serial port (McASP). See your device-specific data manual to determine how many McASPs are available on your device.

Topic	Page
<b>20.1 Registers .....</b>	<b>832</b>

## Introduction

### Purpose of the Peripheral

The multichannel audio serial port (McASP) functions as a general-purpose audio serial port optimized for the needs of multichannel audio applications. The McASP is useful for time-division multiplexed (TDM) stream, Inter-IC Sound (I2S) protocols, and intercomponent digital audio interface transmission (DIT).

The McASP consists of transmit and receive sections that may operate synchronized, or completely independently with separate master clocks, bit clocks, and frame syncs, and using different transmit modes with different bit-stream formats. The McASP module also includes up to 16 serializers that can be individually enabled to either transmit or receive. In addition, all of the McASP pins can be configured as general-purpose input/output (GPIO) pins.

### 20.0.22 Features

Features of the McASP include:

- Two independent clock generator modules for transmit and receive
  - Clocking flexibility allows the McASP to receive and transmit at different rates. For example, the McASP can receive data at 48 kHz but output up-sampled data at 96 kHz or 192 kHz.
- Independent transmit and receive modules, each includes:
  - Programmable clock and frame sync generator
  - TDM streams from 2 to 32, and 384 time slots
  - Support for time slot sizes of 8, 12, 16, 20, 24, 28, and 32 bits
  - Data formatter for bit manipulation
- Up to 16 individually assignable serial data pins:
  - McASP0 can have up to 16 serial data pins
- Glueless connection to audio analog-to-digital converters (ADC), digital-to-analog converters (DAC), codec, digital audio interface receiver (DIR), and S/PDIF transmit physical layer components
- Wide variety of Inter-IC Sound (I2S) and similar bit-stream formats
- 384-slot TDM with external digital audio interface receiver (DIR) device
  - For DIR reception, an external DIR receiver integrated circuit should be used with I2S output format and connected to the McASP receive section.
- Extensive error checking and recovery:
  - Transmit underruns and receiver overruns due to the system not meeting real-time requirements
  - Early or late frame sync in TDM mode
  - Out-of-range high-frequency master clock for both transmit and receive
  - External error signal coming into the AMUTEIN input
  - DMA error due to incorrect programming
- McASP Audio FIFO (AFIFO):
  - Provides additional data buffering
  - Provides added tolerance to variations in host/DMA controller response times
  - May be used as a DMA event pacer
  - Independent Read FIFO and Write FIFO
  - 256 bytes of RAM for each FIFO (read and write)
    - 256 bytes = four 32-bit words per serializer in the case of 16 data pins
    - 256 bytes = 64 32-bit words in the case of one data pin
  - Option to bypass Write FIFO and/or Read FIFO independently

### 20.0.23 Protocols Supported

The McASP supports a wide variety of protocols.

- Transmit section supports
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots
  - S/PDIF, IEC60958-1, AES-3 formats
- Receive section supports
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots
  - TDM stream of 384 time slots specifically designed for easy interface to external digital interface receiver (DIR) device transmitting DIR frames to McASP using the I2S protocol (one time slot for each DIR subframe)

The transmit and receive sections may each be individually programmed to support the following options on the basic serial protocol:

- Programmable clock and frame sync polarity (rising or falling edge): ACLKR/X, AHCLKR/X, and AFSR/X
- Slot length (number of bits per time slot): 8, 12, 16, 20, 24, 28, 32 bits supported
- Word length (bits per word): 8, 12, 16, 20, 24, 28, 32 bits; always less than or equal to the time slot length
- First-bit data delay: 0, 1, 2 bit clocks
- Left/right alignment of word inside slot
- Bit order: MSB first or LSB first
- Bit mask/pad/rotate function
  - Automatically aligns data for DSP internally in either Q31 or integer formats
  - Automatically masks nonsignificant bits (sets to 0, 1, or extends value of another bit)

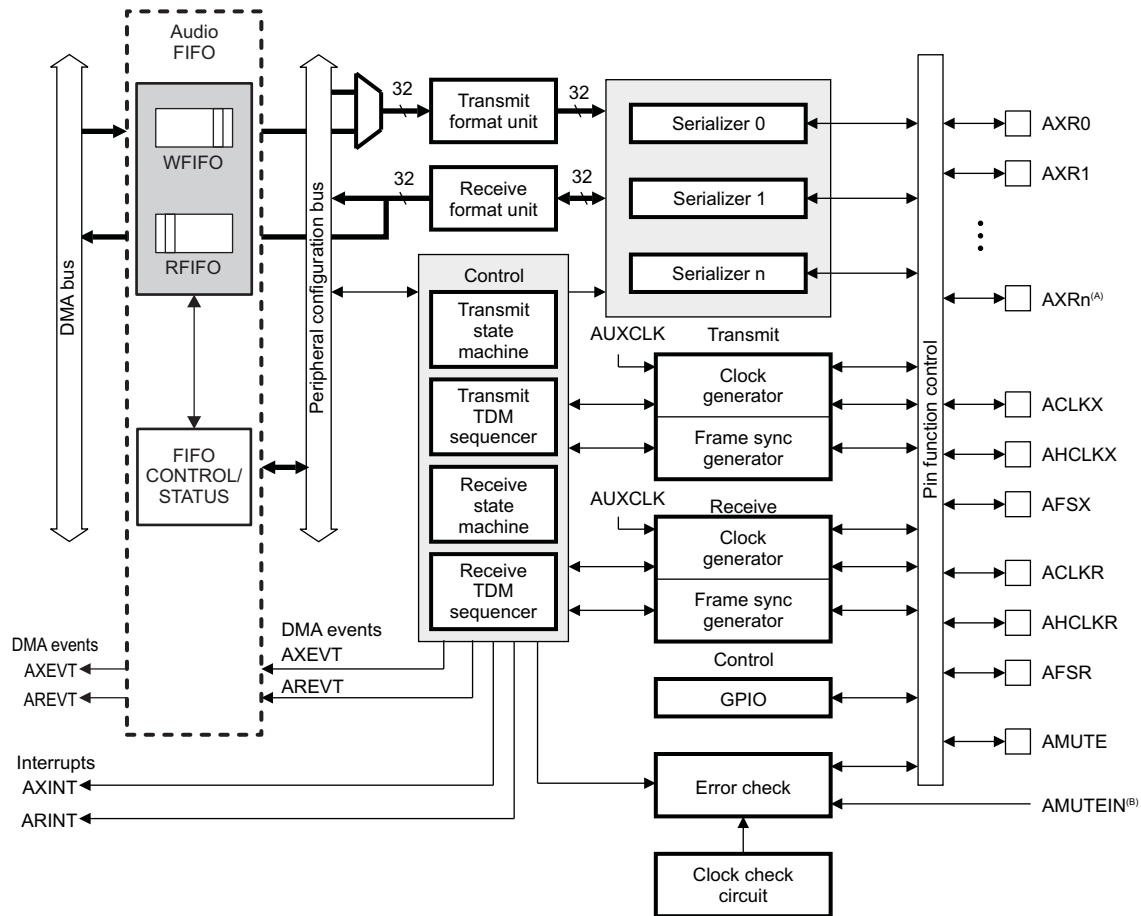
In I2S mode, the transmit and receive sections can support simultaneous transfers on up to all serial data pins operating as 192 kHz stereo channels.

In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to 2 serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for I2S mode, due to the need to generate Biphase Mark Encoded Data).

## 20.0.24 Functional Block Diagram

A block diagram of the McASP is shown in Figure 20-1. The McASP has independent receive/transmit clock generators and frame sync generators.

Figure 20-1. McASP Block Diagram



- A McASP0 has up to 16 serial data pins,  $n = 15$
- B One of the DSP's external pins, see your device-specific data manual.

20.0.24.1 System Level Connections

Figure 20-2 through Figure 20-5 show examples of McASP usage in digital audio encoder/decoder systems.

Figure 20-2. McASP to Parallel 2-Channel DACs

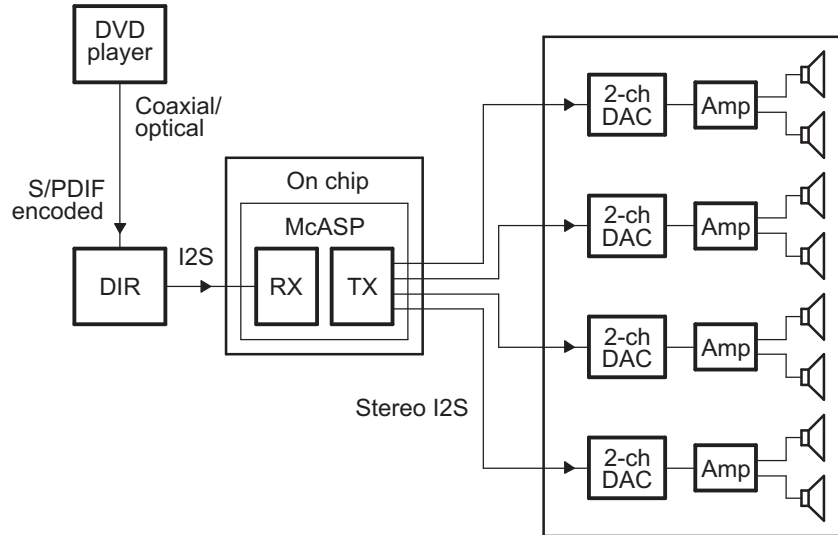
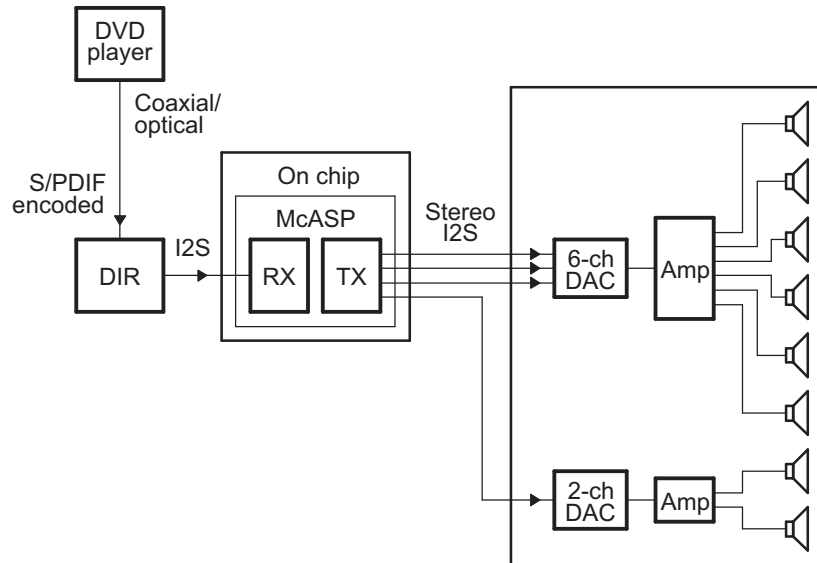
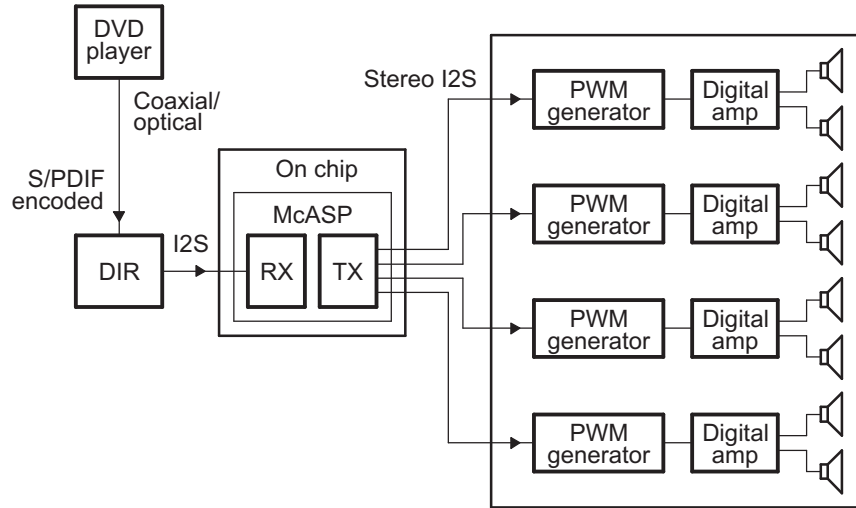


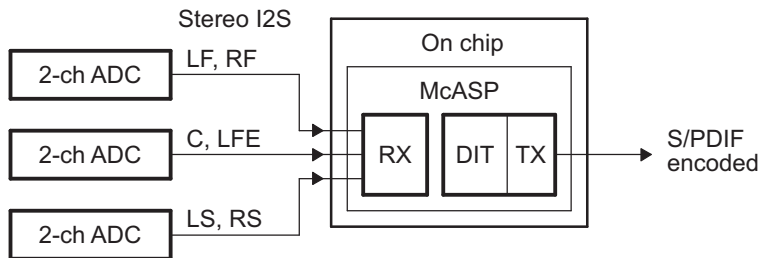
Figure 20-3. McASP to 6-Channel DAC and 2-Channel DAC



**Figure 20-4. McASP to Digital Amplifier**



**Figure 20-5. McASP as Digital Audio Encoder**





## Industry Standard Compliance Statement

The McASP supports the following industry standard interfaces.

### 20.0.24.1 TDM Format

The McASP transmitter and receiver support the multichannel, synchronous time-division-multiplexed (TDM) format via the TDM transfer mode. Within this transfer mode, a wide variety of serial data formats are supported, including formats compatible with devices using the Inter-IC Sound (I2S) protocol. This section briefly discusses the TDM format and the I2S protocol.

#### 20.0.24.1.1 TDM Format

The TDM format is typically used when communicating between integrated circuit devices on the same printed circuit board or on another printed circuit board within the same piece of equipment. For example, the TDM format is used to transfer data between the DSP and one or more analog-to-digital converter (ADC), digital-to-analog converter (DAC), or S/PDIF receiver (DIR) devices.

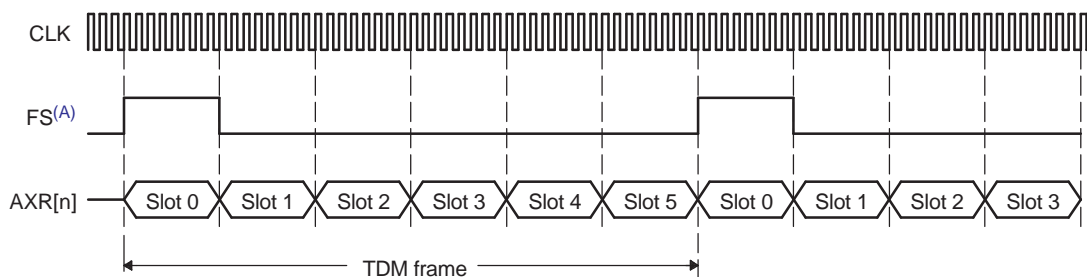
The TDM format consists of three components in a basic synchronous serial transfer: the clock, the data, and the frame sync. In a TDM transfer, all data bits (AXR[n]) are synchronous to the serial clock (ACLKX or ACLKR). The data bits are grouped into words and slots (as defined in Section 20.0.25). The "slots" are also commonly referred to as "time slots" or "channels" in TDM terminology. A frame consists of multiple slots (or channels). Each TDM frame is defined by the frame sync signal (AFSX or AFSR). Data transfer is continuous and periodic, since the TDM format is most commonly used to communicate with data converters that operate at a fixed sample rate.

There are no delays between slots. The last bit of slot N is followed immediately on the next serial clock cycle with the first bit of slot N + 1, and the last bit of the last slot is followed immediately on the next serial clock cycle with the first bit of the first slot. However, the frame sync may be offset from the first bit of the first slot with a 0, 1, or 2-cycle delay.

It is required that the transmitter and receiver in the system agree on the number of bits per slot, since the determination of a slot boundary is not made by the frame sync signal (although the frame sync marks the beginning of slot 0 and the beginning of a new frame).

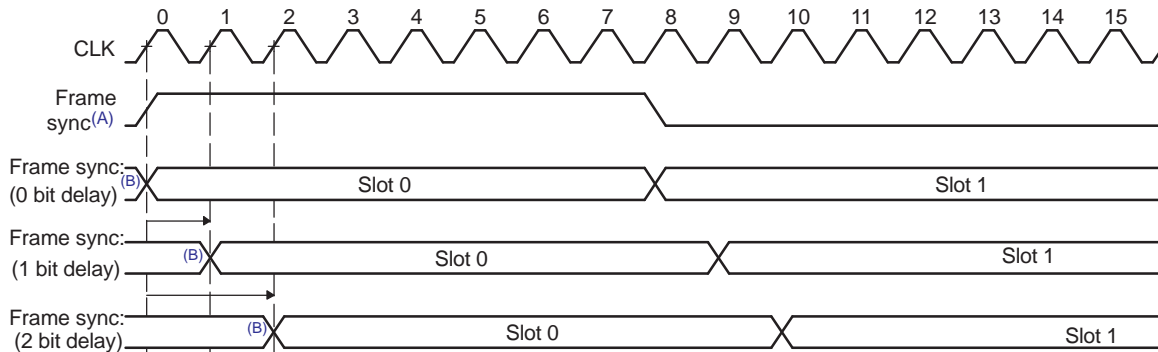
Figure 20-6 shows the TDM format. Figure 20-7 shows the different bit delays from the frame sync.

**Figure 20-6. TDM Format–6 Channel TDM Example**



A FS duration of slot is shown. FS duration of single bit is also supported.

**Figure 20-7. TDM Format Bit Delays from Frame Sync**



- A FS duration of slot is shown. FS duration of single bit is also supported.
- B Last bit of last slot of previous frame. No gap between this bit and the first bit of slot 0 is allowed.

In a typical audio system, one frame of data is transferred during each data converter sample period  $f_s$ . To support multiple channels, the choices are to either include more time slots per frame (thus operating with a higher bit clock rate), or to use additional data pins to transfer the same number of channels (thus operating with a slower bit clock rate).

For example, a particular six channel DAC may be designed to transfer over a single serial data pin AXR[n] as shown in Figure 20-6. In this case the serial clock must run fast enough to transfer a total of 6 channels within each frame period. Alternatively, a similar six channel DAC may be designed to use three serial data pins AXR[0,1,2], transferring two channels of data on each pin during each sample period (Figure 20-8). In the latter case, if the sample period remains the same, the serial clock can run three times slower than the former case. The McASP is flexible enough to support either type of DAC.

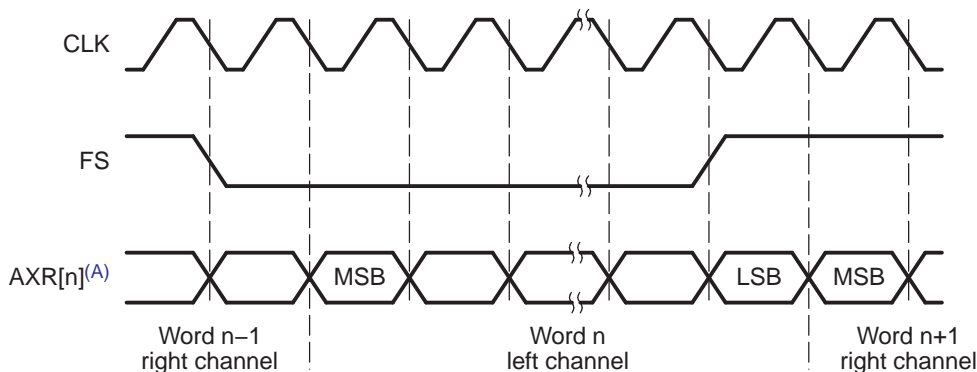
**20.0.24.1.2 Inter-IC Sound (I2S) Format**

The Inter-IC Sound (I2S) format is used extensively in audio interfaces. The TDM transfer mode of the McASP supports the I2S format when configured to 2 slots per frame.

I2S format is specifically designed to transfer a stereo channel (left and right) over a single data pin AXR[n]. "Slots" are also commonly referred to as "channels". The frame width duration in the I2S format is the same as the slot size. The frame signal is also referred to as "word select" in the I2S format. Figure 20-8 shows the I2S protocol.

The McASP supports transfer of multiple stereo channels over multiple AXR[n] pins.

**Figure 20-8. Inter-IC Sound (I2S) Format**



- A 1 to 16 data pins may be supported.

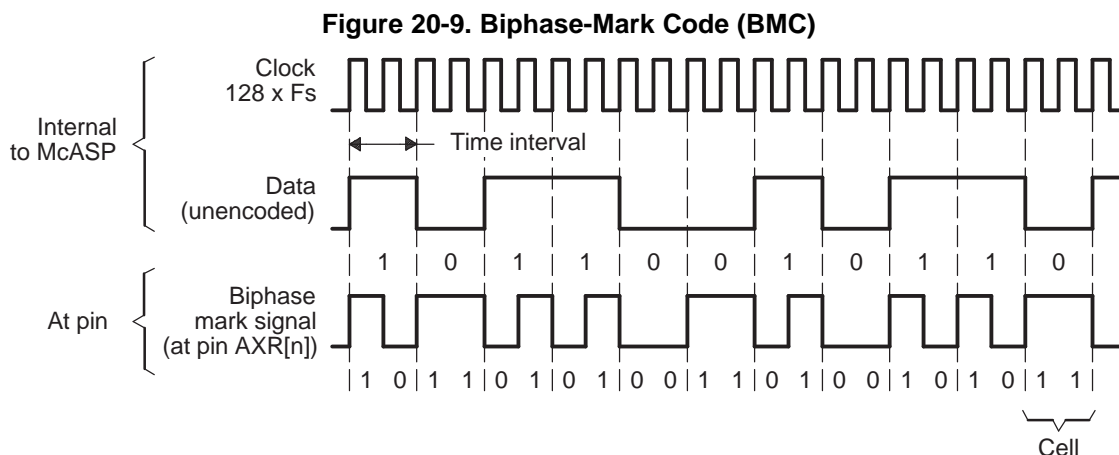
### 20.0.24.2 S/PDIF Coding Format

The McASP transmitter supports the S/PDIF format with 3.3V biphasemark encoded output. The S/PDIF format is supported by the digital audio interface transmit (DIT) transfer mode of the McASP. This section briefly discusses the S/PDIF coding format.

#### 20.0.24.2.1 Biphasemark Code (BMC)

In S/PDIF format, the digital signal is coded using the biphasemark code (BMC). The clock, frame, and data are embedded in only one signal—the data pin AXR[n]. In the BMC system, each data bit is encoded into two logical states (00, 01, 10, or 11) at the pin. These two logical states form a cell. The duration of the cell, which equals to the duration of the data bit, is called a time interval. A logical 1 is represented by two transitions of the signal within a time interval, which corresponds to a cell with logical states 01 or 10. A logical 0 is represented by one transition within a time interval, which corresponds to a cell with logical states 00 or 11. In addition, the logical level at the start of a cell is inverted from the level at the end of the previous cell. Figure 20-9 and Table 20-1 show how data is encoded to the BMC format.

As shown in Figure 20-9, the frequency of the clock is twice the unencoded data bit rate. In addition, the clock is always programmed to  $128 \times f_s$ , where  $f_s$  is the sample rate (see Section 20.0.24.2.3 for details on how this clock rate is derived based on the S/PDIF format). The device receiving in S/PDIF format can recover the clock and frame information from the BMC signal.



**Table 20-1. Biphasemark Encoder**

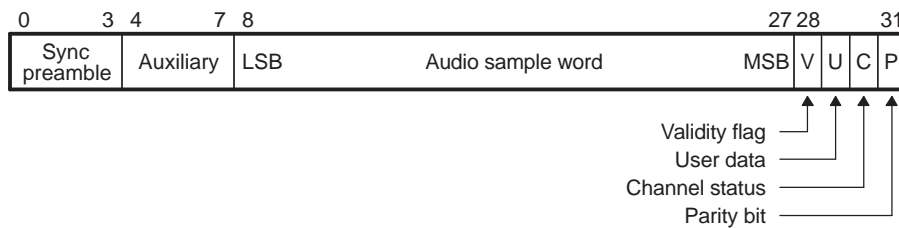
Data (Unencoded)	Previous State at Pin AXR[n]	BMC-Encoded Cell Output at AXR[n]
0	0	11
0	1	00
1	0	10
1	1	01

### 20.0.24.2.2 Subframe Format

Every audio sample transmitted in a subframe consists of 32 S/PDIF time intervals (or cells), numbered from 0 to 31. [Figure 20-10](#) shows a subframe.

- **Time intervals 0-3** carry one of the three permitted preambles to signify the type of audio sample in the current subframe. The preamble is *not* encoded in BMC format, and therefore the preamble code can contain more than two consecutive 0 or 1 logical states in a row. See [Table 20-2](#).
- **Time intervals 4-27** carry the audio sample word in linear 2s-complement representation. The most-significant bit (MSB) is carried by time interval 27. When a 24-bit coding range is used, the least-significant bit (LSB) is in time interval 4. When a 20-bit coding range is used, time intervals 8-27 carry the audio sample word with the LSB in time interval 8. Time intervals 4-7 may be used for other applications and are designated auxiliary sample bits.
- If the source provides fewer bits than the interface allows (either 20 or 24), the unused LSBs are set to logical 0. For a nonlinear PCM audio application or a data application, the main data field may carry any other information.
- **Time interval 28** carries the validity bit (V) associated with the main data field in the subframe.
- **Time interval 29** carries the user data channel (U) associated with the main data field in the subframe.
- **Time interval 30** carries the channel status information (C) associated with the main data field in the subframe. The channel status indicates if the data in the subframe is digital audio or some other type of data.
- **Time interval 31** carries a parity bit (P) such that time intervals 4-31 carry an even number of 1s and an even number of 0s (even parity). As shown in [Table 20-2](#), the preambles (time intervals 0-3) are also defined with even parity.

**Figure 20-10. S/PDIF Subframe Format**



**Table 20-2. Preamble Codes**

Preamble Code <sup>(1)</sup>	Previous Logical State	Logical States on pin AXR[n] <sup>(2)</sup>	Description
B (or Z)	0	1110 1000	Start of a block and subframe 1
M (or X)	0	1110 0010	Subframe 1
W (or Y)	0	1110 0100	Subframe 2

<sup>(1)</sup> Historically, preamble codes are referred to as B, M, W. For use in professional applications, preambles are referred to as Z, X, Y, respectively.

<sup>(2)</sup> The preamble is not BMC encoded. Each logical state is synchronized to the serial clock. These 8 logical states make up time slots (cells) 0 to 3 in the S/PDIF stream.

As shown in [Table 20-2](#), the McASP DIT only generates one polarity of preambles and it assumes the previous logical state to be 0. This is because the McASP assures an even-polarity encoding scheme when transmitting in DIT mode. If an underrun condition occurs, the DIT resynchronizes to the correct logic level on the AXR[n] pin before continuing with the next transmission.

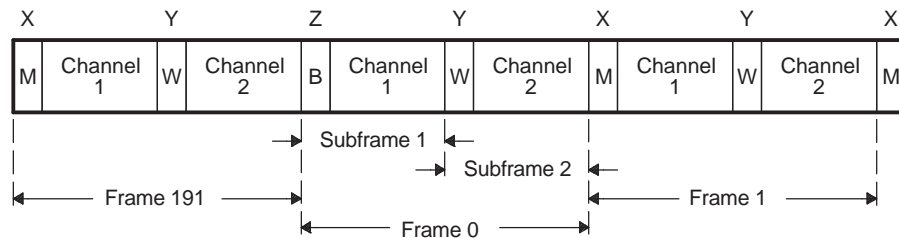
### 20.0.24.2.3 Frame Format

An S/PDIF frame is composed of two subframes (Figure 20-11). For linear coded audio applications, the rate of frame transmission normally corresponds exactly to the source sampling frequency  $f_s$ . The S/PDIF format clock rate is therefore  $128 \times f_s$  ( $128 = 32 \text{ cells/subframe} \times 2 \text{ clocks/cell} \times 2 \text{ subframes/sample}$ ). For example, for an S/PDIF stream at a 192 kHz sampling frequency, the serial clock is  $128 \times 192 \text{ kHz} = 24.58 \text{ MHz}$ .

In 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive subframes. Both subframes contain valid data. The first subframe (**left** or **A** channel in stereophonic operation and **primary** channel in monophonic operation) normally starts with preamble M. However, the preamble of the first subframe changes to preamble B once every 192 frames to identify the start of the block structure used to organize the channel status information. The second subframe (**right** or **B** channel in stereophonic operation and **secondary** channel in monophonic operation) always starts with preamble W.

In single-channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried in the first subframe and may be duplicated in the second subframe. If the second subframe is not carrying duplicate data, time slot 28 (validity bit) is set to logical 1.

**Figure 20-11. S/PDIF Frame Format**



## 20.0.25 Definition of Terms

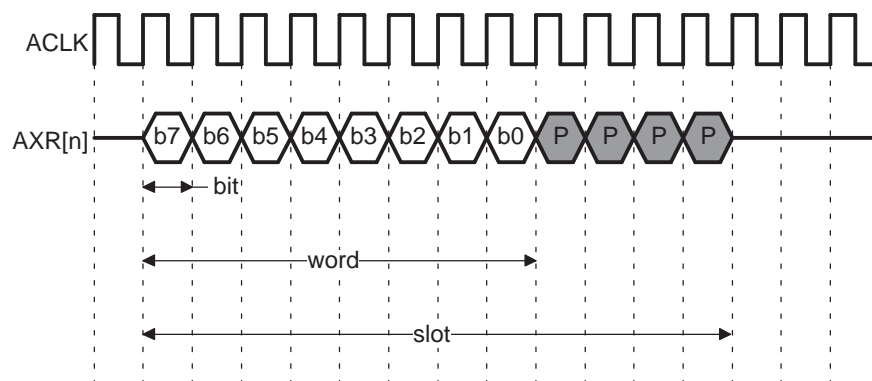
The serial bit stream transmitted or received by the McASP is a long sequence of 1s and 0s, either output or input on one of the audio transmit/receive pins (AXR[n]). However, the sequence has a hierarchical organization that can be described in terms of frames of data, slots, words, and bits.

A basic synchronous serial interface consists of three important components: clock, frame sync, and data. [Figure 20-12](#) shows two of the three basic components—the clock (ACLK) and the data (AXR[n]).

[Figure 20-12](#) does not specify whether the clock is for transmit (ACLKX) or receive (ACLKR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses ACLKX as the serial clock, and the receiver uses ACLKR as the serial clock. Optionally, the receiver can use ACLKX as the serial clock when the transmitter and receiver of the McASP are configured to operate synchronously.

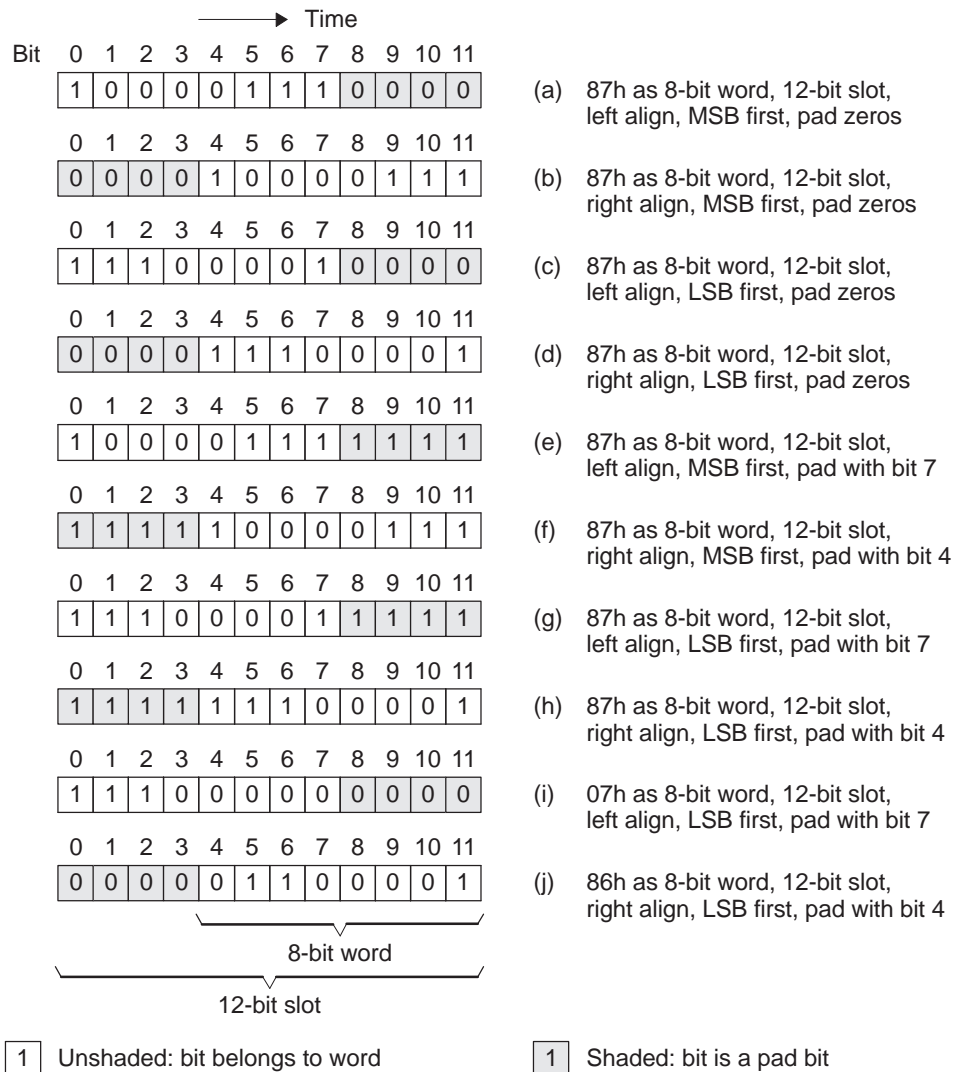
- Bit** A bit is the smallest entity in the serial data stream. The beginning and end of each bit is marked by an edge of the serial clock. The duration of a bit is a serial clock period. A 1 is represented by a logic high on the AXR[n] pin for the entire duration of the bit. A 0 is represented by a logic low on the AXR[n] pin for the entire duration of the bit.
- Word** A word is a group of bits that make up the data being transferred between the DSP and the external device. [Figure 20-12](#) shows an 8-bit word.
- Slot** A slot consists of the bits that make up the word, and may consist of additional bits used to pad the word to a convenient number of bits for the interface between the DSP and the external device. In [Figure 20-12](#), the audio data consists of only 8 bits of useful data (8-bit word), but it is padded with 4 zeros (12-bit slot) to satisfy the desired protocol in interfacing to an external device. Within a slot, the bits may be shifted in/out of the McASP on the AXR[n] pin either MSB or LSB first. When the word size is smaller than the slot size, the word may be aligned to the left (beginning) of the slot or to the right (end) of the slot. The additional bits in the slot not belonging to the word may be padded with 0, 1, or with one of the bits (the MSB or the LSB typically) from the data word. These options are shown in [Figure 20-13](#).

**Figure 20-12. Definition of Bit, Word, and Slot**



- (1) b7:b0 - bits. Bits b7 to b0 form a word.
- (2) P - pad bits. Bits b7 to b0, together with the four pad bits, form a slot.
- (3) In this example, the data is transmitted MSB first, left aligned.

**Figure 20-13. Bit Order and Word Alignment Within a Slot Examples**

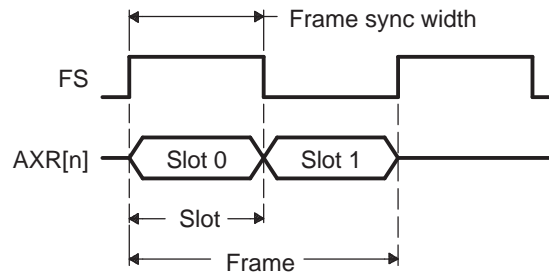


The third basic element of a synchronous serial interface is the frame synchronization signal, also referred to as frame sync in this chapter.

**Frame** A frame contains one or multiple slots, as determined by the desired protocol. [Figure 20-14](#) shows an example frame of data and the frame definitions. [Figure 20-14](#) does not specify whether the frame sync (FS) is for transmit (AFSX) or receive (AFSR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses AFSX and the receiver uses AFSR. Optionally, the receiver can use AFSX as the frame sync when the transmitter and receiver of the McASP are configured to operate synchronously.

This section only shows the generic definition of the frame sync. See [Section 20.0.24.1](#), [Section 20.0.24.2](#), and [Section 20.0.27.2.1](#) for details on the frame sync formats required for the different transfer modes and protocols (burst mode, TDM mode and I2S format, DIT mode and S/PDIF format).

**Figure 20-14. Definition of Frame and Frame Sync Width**



(1) In this example, there are two slots in a frame, and FS duration of slot length is shown.

Other terms used throughout this chapter:

**TDM** Time-division multiplexed. See [Section 20.0.24.1](#) for details on the TDM protocol.

**DIR** Digital audio interface receive. The McASP does not natively support receiving in the S/PDIF format. The McASP supports I2S format output by an external DIR device.

**DIT** Digital audio interface transmit. The McASP supports transmitting in S/PDIF format on up to all data pins configured as outputs.

**I2S** Inter-IC Sound protocol, commonly used on audio interfaces. The McASP supports the I2S protocol as part of the TDM mode (when configured as a 2-slot frame).

**Slot or Time Slot** For TDM format, the term time slot is interchangeable with the term slot defined in this section. For DIT format, a McASP time slot corresponds to a DIT subframe.



## Architecture

### 20.0.26 Overview

Figure 20-1 shows the major blocks of the McASP. The McASP has independent receive/transmit clock generators and frame sync generators, error-checking logic, and up to 16 serial data pins. See your device-specific data manual for the number of data pins available on your device.

All the McASP pins on the device may be individually programmed as general-purpose I/O (GPIO) if they are not used for serial port functions.

The McASP includes the following pins:

- Serializers
  - Data pins AXR[n]: Up to sixteen per McASP
- Transmit clock generator:
  - AHCLKX: McASP transmit high-frequency master clock
  - ACLKX: McASP transmit bit clock
- Transmit Frame Sync Generator
  - AFSX: McASP transmit frame sync or left/right clock (LRCLK)
- Receive clock generator:
  - AHCLKR: McASP receive high-frequency master clock
  - ACLKR: McASP receive bit clock
- Receive Frame Sync Generator
  - AFSR: McASP receive frame sync or left/right clock (LRCLK)
- Mute in/out:
  - AMUTEIN: McASP mute input (from external device)
  - AMUTE: McASP mute output
  - Data pins AXR[n]

### 20.0.27 Clock and Frame Sync Generators

The McASP clock generators are able to produce two independent clock zones: transmit and receive clock zones. The serial clock generators may be programmed independently for the transmit section and the receive section, and may be completely asynchronous to each other. The serial clock (clock at the bit rate) may be sourced:

- **Internally** - by passing through two clock dividers off the internal clock source (AUXCLK)
- **Externally** - directly from ACLKR/X pin
- **Mixed** - an external high-frequency clock is input to the McASP on either the AHCLKX or AHCLKR pins, and divided down to produce the bit rate clock

In the internal/mixed cases, the bit rate clock is generated internally and should be driven out on the ACLKX (for transmit) or ACLKR (for receive) pins. In the internal case, an internally-generated high-frequency clock may be driven out onto the AHCLKX or AHCLKR pins to serve as a reference clock for other components in the system.

The McASP requires a minimum of a bit clock and a frame sync to operate, and provides the capability to reference these clocks from an external high-frequency master clock. In DIT mode, it is possible to use only internally-generated clocks and frame syncs.

### 20.0.27.1 Transmit Clock

The transmit bit clock, ACLKX, (Figure 20-15) may be either externally sourced from the ACLKX pin or internally generated, as selected by the CLKXM bit. If internally generated (CLKXM = 1), the clock is divided down by a programmable bit clock divider (CLKXDIV) from the transmit high-frequency master clock (AHCLKX).

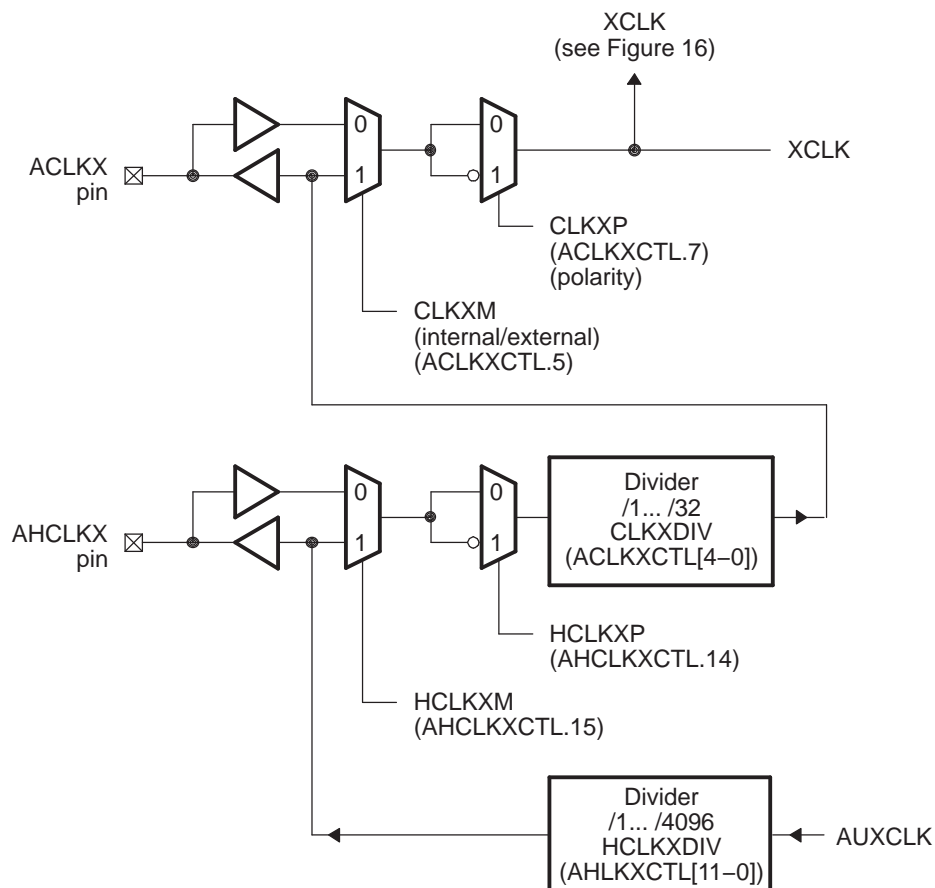
Internally, the McASP always shifts transmit data at the rising edge of the internal transmit clock, XCLK, (Figure 20-15). The CLKXP mux determines if ACLKX needs to be inverted to become XCLK. If CLKXP = 0, the CLKXP mux directly passes ACLKX to XCLK. As a result, the McASP shifts transmit data at the rising edge of ACLKX. If CLKXP = 1, the CLKXP mux passes the inverted version of ACLKX to XCLK. As a result, the McASP shifts transmit data at the falling edge of ACLKX.

The transmit high-frequency master clock, AHCLKX, may be either externally sourced from the AHCLKX pin or internally generated, as selected by the HCLKXM bit. If internally generated (HCLKXM = 1), the clock is divided down by a programmable high clock divider (HCLKXDIV) from McASP internal clock source AUXCLK. The transmit high-frequency master clock may be (but is not required to be) output on the AHCLKX pin where it is available to other devices in the system.

The transmit clock configuration is controlled by the following registers:

- ACLKXCTL
- AHCLKXCTL

**Figure 20-15. Transmit Clock Generator Block Diagram**



### 20.0.27.2 Receive Clock

The receiver has a clock generation circuit identical to (but independent of) that of the transmitter. The receive bit clock, ACLKR, (Figure 20-16) may be either externally sourced from the ACLKR pin or internally generated, as selected by the CLKRM bit. If internally generated (CLKRM = 1), the clock is divided down by a programmable divider (CLKRDIV) from the receive high-frequency master clock (AHCLKR). Regardless if ACLKR is either internally generated or externally sourced, polarity of the clock may be programmed (CLKRP) to be either rising or falling edge.

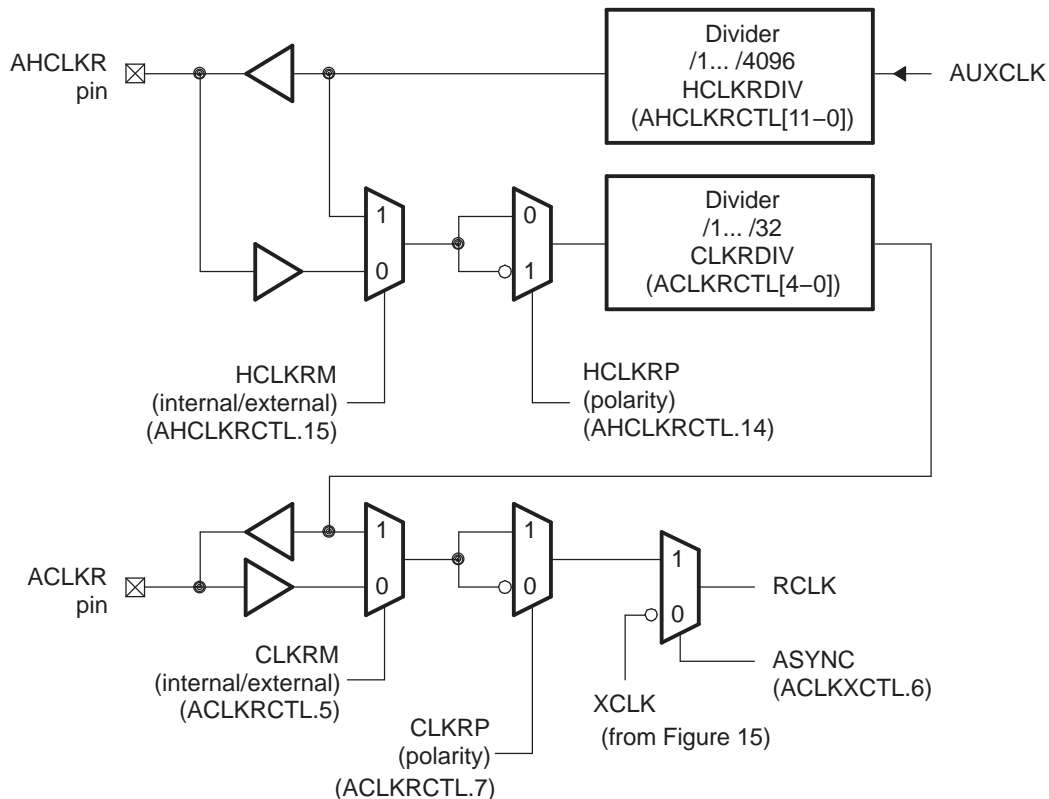
The receive high-frequency master clock, AHCLKR, may be either externally sourced from the AHCLKR pin or internally generated, as selected by the HCLKRM bit. If internally generated (HCLKRM = 1), the clock is divided down by a programmable divider (HCLKRDIV) from AUXCLK. The receive high-frequency master clock may be (but is not required to be) output on the AHCLKR pin where it is available to other devices in the system. Regardless if AHCLKR is either internally generated or externally sourced, polarity of the high-frequency clock may be programmed (HCLKRP) to be either rising or falling edge.

The receiver also has the option to operate synchronously from the ACLKX and AFSX signals. This is achieved when the ASYNC bit in the transmit clock control register (ACLKXCTL) is cleared to 0. See Section 20.0.27.1.5 for details on McASP operation when ACLKXCTL.ASYNC = 0.

The receive clock configuration is controlled by the following registers:

- ACLKRCTL
- AHCLKRCTL

Figure 20-16. Receive Clock Generator Block Diagram



### 20.0.27.3 Frame Sync Generator

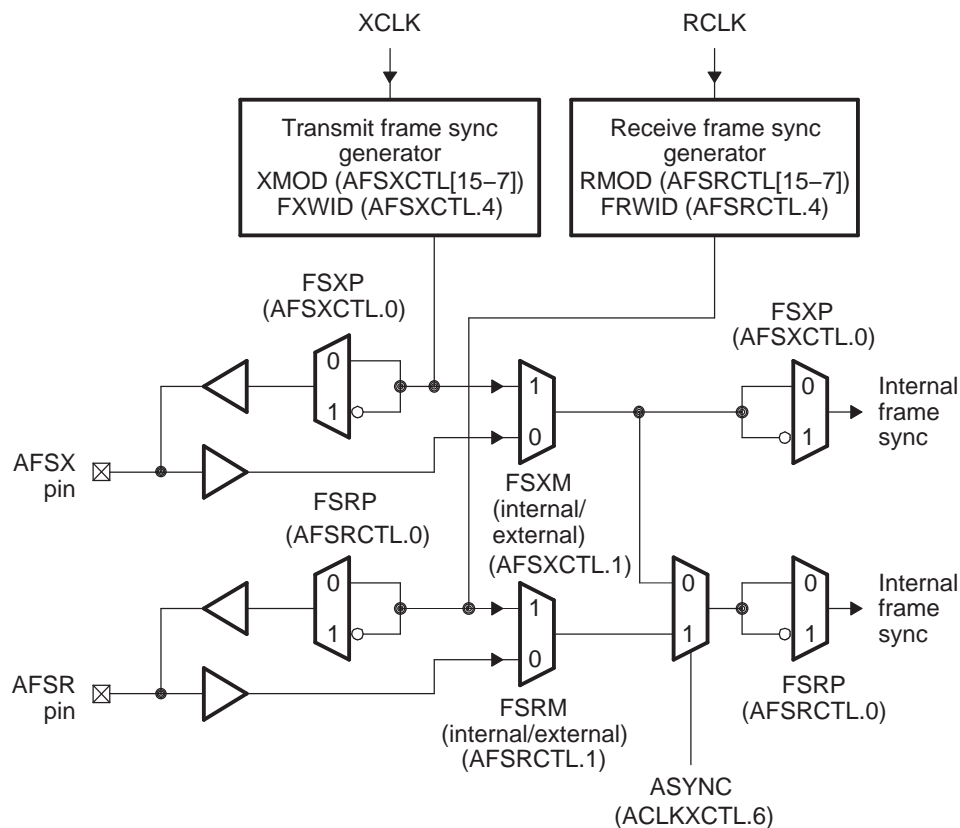
There are two different modes for frame sync: burst and TDM. A block diagram of the frame sync generator is shown in Figure 20-17. The frame sync options are programmed by the receive and transmit frame sync control registers (AFSRCTL and AFSXCTL). The options are:

- Internally-generated or externally-generated
- Frame sync polarity: rising edge or falling edge
- Frame sync width: single bit or single word
- Bit delay: 0, 1, or 2 cycles before the first data bit

The transmit frame sync pin is AFSX and the receive frame sync pin is AFSR. A typical usage for these pins is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

Regardless if the AFSX/AFSR is internally generated or externally sourced, the polarity of AFSX/AFSR is determined by FSXP/FSRP, respectively, to be either rising or falling edge. If FSXP/FSRP = 0, the frame sync polarity is rising edge. If FSXP/FSRP = 1, the frame sync polarity is falling edge.

Figure 20-17. Frame Sync Generator Block Diagram



## 20.0.27.4 Clocking Examples

Some examples of processes using the McASP clocking and frame flexibility are:

- Receive data from a DVD at 48 kHz, but output up-sampled or decoded audio at 96 kHz or 192 kHz. This could be accomplished by inputting a high-frequency master clock (for example,  $512 \times$  receive FS), receiving with an internally-generated bit clock ratio of divide-by-8, and transmitting with an internally-generated bit clock ratio of divide-by-4 or divide-by-2.
- Transmit/receive data based on one sample rate (for example, 44.1 kHz), and transmit/receive data at a different sample rate (for example, 48 kHz).

## General Architecture

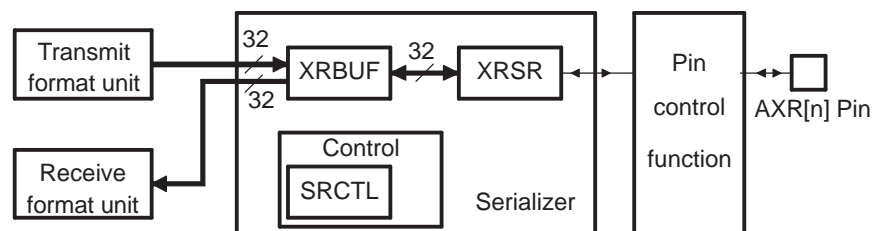
### 20.0.27.1 Serializers

The serializers take care of shifting serial data in and out of the McASP. Each serializer consists of a shift register (XRSR), a data buffer (XRBUF), a control register (SRCTL), and logic to support the data alignment options of the McASP. For each serializer, there is a dedicated serial data pin (AXR[n]) and a dedicated control register (SRCTL[n]). The control register allows the serializer to be configured as a transmitter, receiver, or as inactive. When configured as a transmitter the serializer shifts out data to the serial data pin AXR[n]. When configured as a receiver, the serializer shifts in data from the AXR[n] pin. The serializer is clocked from the transmit/receive section clock (ACLKX/ACLKR) if configured to transmit/receive respectively.

All serializers that are configured to transmit operate in lock-step. Similarly, all serializers that are configured to receive also operate in lock-step. This means that at most there are two zones per McASP, one for transmit and one for receive.

Figure 20-18 shows the block diagram of the serializer and its interface to other units within the McASP.

**Figure 20-18. Individual Serializer and Connections Within McASP**



For receive, data is shifted in through the AXR[n] pin to the shift register XRSR. Once the entire slot of data is collected in the XRSR, the data is copied to the data buffer XRBUF. The data is now ready to be read by the DSP through the RBUF register, which is an alias of the XRBUF for receive. When the DSP reads from the RBUF, the McASP passes the data from RBUF through the receive format unit and returns the formatted data to the DSP.

For transmit, the DSP services the McASP by writing data into the XBUF register, which is an alias of the XRBUF for transmit. The data automatically passes through the transmit format unit before actually reaching the XRBUF in the serializer. The data is then copied from XRBUF to XRSR, and shifted out from the AXR[n] synchronously to the serial clock.

In DIT mode, in addition to the data, the serializer shifts out other DIT-specific information accordingly (preamble, user data, etc.).

The serializer configuration is controlled by SRCTL[n].

### 20.0.27.2 Format Unit

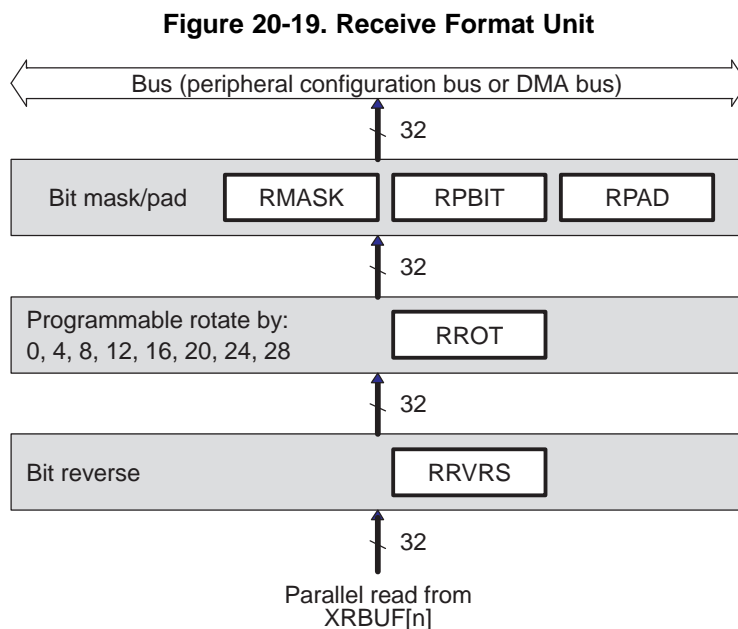
The McASP has two data formatting units, one for transmit and one for receive. These units automatically remap the data bits within the transmitted and received words between a natural format for the DSP (such as a Q31 representation) and the required format for the external serial device (such as "I2S format"). During the remapping process, the format unit also can mask off certain bits or perform sign extension.

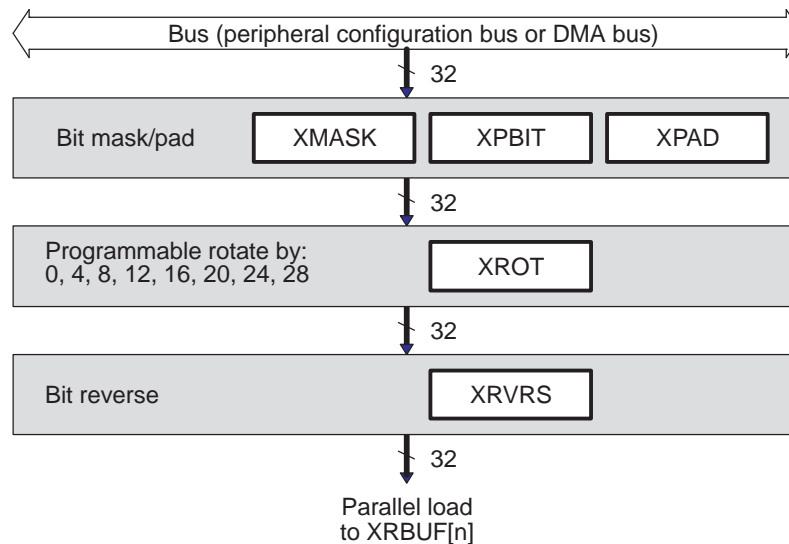
Since all transmitters share the same data formatting unit, the McASP only supports one transmit format at a time. For example, the McASP will not transmit in "I2S format" on serializer 0, while transmitting "Left Justified" on serializer 1. Likewise, the receiver section of the McASP only supports one data format at a time, and this format applies to all receiving serializers. However, the McASP can transmit in one format while receiving in a completely different format.

This formatting unit consists of three stages:

- Bit mask and pad (masks off bits, performs sign extension)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB first or LSB first)

Figure 20-19 shows a block diagram of the receive formatting unit, and Figure 20-20 shows the transmit formatting unit. Note that the order in which data flows through the three stages is different between the transmit and receive formatting units.



**Figure 20-20. Transmit Format Unit**


The bit mask and pad stage includes a full 32-bit mask register, allowing selected individual bits to either pass through the stage unchanged, or be masked off. The bit mask and pad then pad the value of the masked off bits by inserting either a 0, a 1, or one of the original 32 bits as the pad value. The last option allows for sign-extension when the sign bit is selected to pad the remaining bits.

The rotate right stage performs bitwise rotation by a multiple of 4 bits (between 0 and 28 bits), programmable by the (R/X)FMT register. Note that this is a rotation process, not a shifting process, so bit 0 gets shifted back into bit 31 during the rotation.

The bit reversal stage either passes all 32 bits directly through, or swaps them. This allows for either MSB or LSB first data formats. If bit reversal is not enabled, then the McASP will naturally transmit and receive in an LSB first order.

Finally, note that the (R/X)DATDLY bits in (R/X)FMT also determine the data format. For example, the difference between I2S format and left-justified is determined by the delay between the frame sync edge and the first data bit of a given time slot. For I2S format, (R/X)DATDLY should be set to a 1-bit delay, whereas for left-justified format, it should be set to a 0-bit delay.

The combination of all the options in (R/X)FMT means that the McASP supports a wide variety of data formats, both on the serial data lines, and in the internal DSP representation.

[Section 20.0.27.4](#) provides more detail and specific examples. The examples use internal representation in integer and Q31 notation, but other fractional notations are also possible.

### 20.0.27.3 State Machine

The receive and transmit sections have independent state machines. Each state machine controls the interactions between the various units in the respective section. In addition, the state machine keeps track of error conditions and serial port status.

No serial transfers can occur until the respective state machine is released from reset. See initialization sequence for details ([Section 20.0.27.1](#)).

The receive state machine is controlled by the RFMT register, and it reports the McASP status and error conditions in the RSTAT register. Similarly, the transmit state machine is controlled by the XFMT register, and it reports the McASP status and error conditions in the XSTAT register.

#### 20.0.27.4 TDM Sequencer

There are separate TDM sequencers for the transmit section and the receive section. Each TDM sequencer keeps track of the slot count. In addition, the TDM sequencer checks the bits of (R/X)TDM and determines if the McASP should receive/transmit in that time slot.

If the McASP should participate (transmit/receive bit is active) in the time slot, the McASP functions normally. If the McASP should not participate (transmit/receive bit is inactive) in the time slot, no transfers between the XRBUF and XRSR registers in the serializer would occur during that time slot. In addition, the serializers programmed as transmitters place their data output pins in a predetermined state (logic low, high, or high impedance) as programmed by each serializer control register (SRCTL). Refer also to [Section 20.0.27.2.2](#) for details on how DMA event or interrupt generations are handled during inactive time slots in TDM mode.

The receive TDM sequencer is controlled by register RTDM and reports current receive slot to RSLOT. The transmit TDM sequencer is controlled by register XTDM and reports current transmit slot to XSLOT.

#### 20.0.27.5 Clock Check Circuit

A common source of error in audio systems is a serial clock failure due to instabilities in the off-chip DIR circuit. To detect a clock error quickly, a clock-check circuit is included in the McASP for both transmit and receive clocks, since both may be sourced from off chip.

The clock check circuit can detect and recover from transmit and receive clock failures. See [Section 20.0.27.6.6](#) for implementation and programming details.

#### 20.0.27.6 Pin Function Control

All McASP pins except AMUTEIN are bidirectional input/output pins. In addition, these bidirectional pins function either as McASP or general-purpose I/O (GPIO) pins. The following registers control the pin functions:

- Pin function register (PFUNC): selects pin to function as McASP or GPIO
- Pin direction register (PDIR): selects pin to be input or output
- Pin data input register (PDIN): shows data input at the pin
- Pin data output register (PDOUT): data to be output at the pin if the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1). Not applicable when the pin is configured as McASP pin (PFUNC[n] = 0).
- Pin data set register (PDSET): alias of PDOUT. Writing a 1 to PDSET[n] sets the respective PDOUT[n] to 1. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1).
- Pin data clear register (PDCLR): alias of PDOUT. Writing a 1 to PDCLR[n] clears the respective PDOUT[n] to 0. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1).

See the register descriptions in [Section 20.1](#) for details on the mapping of each McASP pin to the register bits. [Figure 20-21](#) shows the pin control block diagram.

##### 20.0.27.6.1 McASP Pin Control-Transmit and Receive

You must correctly set the McASP GPIO registers PFUNC and PDIR, even when McASP pins are used for their serial port (non-GPIO) function.

Serial port functions include:

- Clock pins (ACLKX, ACLKR, AHCLKX, AHCLKR, AFSX, AFSR) used as clock inputs and outputs
- Serializer data pins (AXR[n]) used to transmit or receive
- AMUTE used as a mute output signal

When using these pins in their serial port function, you must clear PFUNC[n] to 0 for each pin, as opposed to PFUNC[n] = 1, which makes the pin a GPIO.



Also, certain outputs require  $PDIR[n] = 1$ , such as clock pins used as clock outputs, serializer data pins used to transmit, and AMUTE used as mute output.

Clock inputs and serializers configured to receive must have  $PDIR[n] = 0$ .

PFUNC and PDIR do not control the AMUTEIN device pin, it is usually tied to a device pin (see your device-specific data manual). If used as a mute input, this pin needs to be configured as an input in the appropriate peripheral.

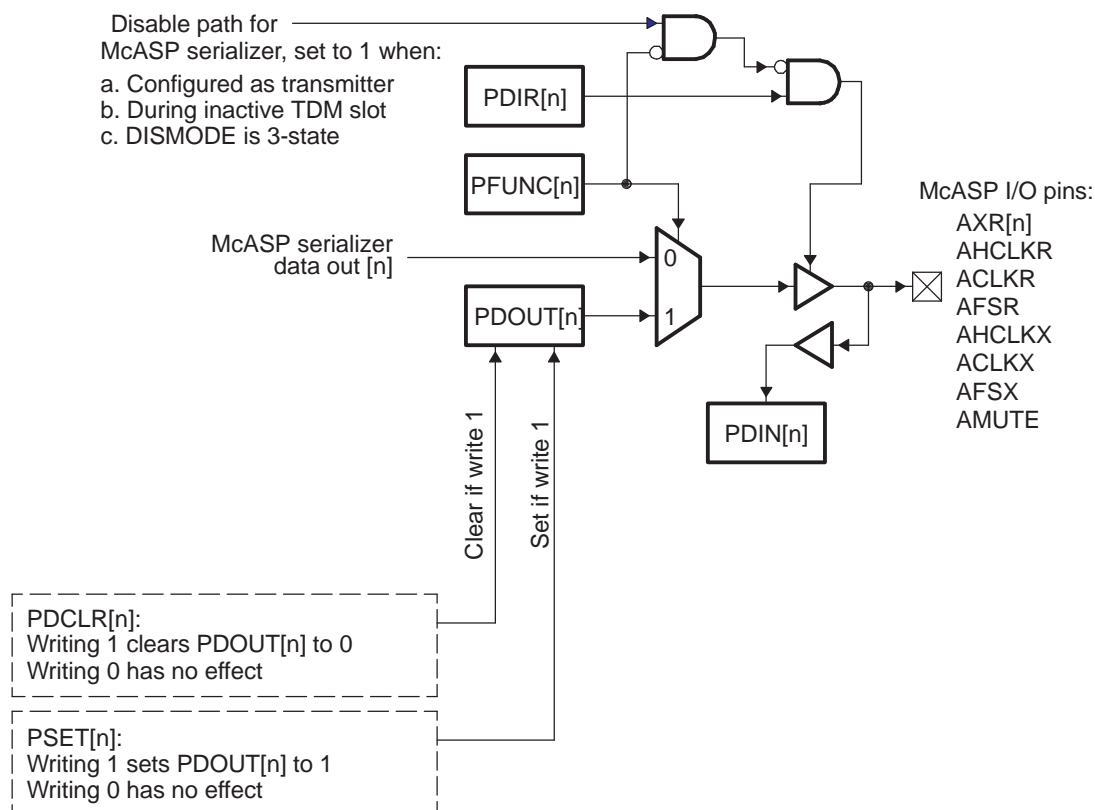
Finally, there is an important advantage to having separate control of pin direction (by PDIR), and the choice of internal versus external clocking (by CLKRM/CLKXM). Depending on the specific device and usage, you might select an external clock ( $CLKRM = 0$ ), while enabling the internal clock divider, and the clock pin as an output in the PDIR register ( $PDIR[ACLKR] = 1$ ). In this case, the bit clock is an output ( $PDIR[ACLKR] = 1$ ) and, therefore, routed to the ACLKR pin. However, because  $CLKRM = 0$ , the bit clock is then routed back to the McASP module as an "external" clock source. This may result in less skew between the clock inside the McASP and the clock in the external device, thus producing more balanced setup and hold times for a particular system. As a result, this may allow a higher serial clock rate interface.

### 20.0.27.6.2 GPIO Pin Control

For GPIO operation, you must set the desired  $PFUNC[n]$  to 1 to indicate GPIO function.  $PDIR[n]$  must be configured to the desired direction. PDOUT, PDSET, PDCLR control the output value on the pin. PDIN always reflects the state at the pin, regardless of the PDIR and PFUNC setting.

Figure 20-21 and Figure 20-22 display the pin descriptions. The examples that follow (Example 20-1 through Example 20-4) show how the pins can be used as general-purpose input or output pins.

Figure 20-21. McASP I/O Pin Control Block Diagram



**Figure 20-22. McASP I/O Pin to Control Register Mapping**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23	Reserved						16
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

### Example 20-1. General-Purpose Input Pin

Because the PDIN register always reflects the state at the pin, you can read the PDIN register to obtain the pin input state. To explicitly set the pin as a general-purpose input pin, you can set the registers as follows:

- $PDIN[n] = 0$  (input)
- $PFUNC[n] = 1$  (GPIO function)

### Example 20-2. General-Purpose Output Pin—Initialization Using PDOUT

All pins default as inputs. To initialize a pin as output, you should follow this sequence:

1.  $PDIN[n] = 0$  (default as input)
2.  $PFUNC[n] = 1$  (GPIO function)
3.  $PDOUT[n] =$  desired output value
4.  $PDIN[n] = 1$  (change to output after desired value is configured in  $PDOUT[n]$ )

### Example 20-3. General-Purpose Output Pin—Change Data from 0 to 1 Using PDSET

If the pin is already configured as a general-purpose output pin driving a 0, and you want to change the output from 0 to 1, the recommended method is to use the PDSET register instead of the PDOUT register. This is because writing to the PDSET register only affects pin(s) in concern. To change a pin from 0 to 1:

- Set  $PDSET[n]$ . This sets the respective  $PDOUT[n]$ .

### Example 20-4. General-Purpose Output Pin—Change Data from 1 to 0 Using PDCLR

If the pin is already configured as a general-purpose output pin driving a 1, and you want to change the output from 1 to 0, the recommended method is to use the PDCLR register instead of the PDOUT register. This is because writing to the PDCLR register only affects pin(s) in concern. To change a pin from 1 to 0:

- Set  $PDCLR[n]$ . This clears the respective  $PDOUT[n]$ .

## McASP Audio FIFO (AFIFO)

The McASP Audio FIFO (AFIFO) provides additional data buffering for the McASP. The time it takes the host CPU or DMA controller to respond to DMA requests from the McASP may vary; the additional buffering provided by the AFIFO allows greater tolerance to such variations.

For convenience, the AFIFO is treated here as a block between McASP and the host/DMA controller (see [Figure 20-1](#)). Details on configuration of the AFIFO are provided in [McASP Audio FIFO \(AFIFO\)](#).

### Operation

This section discusses the operation of the McASP.

#### 20.0.27.1 Setup and Initialization

This section discusses steps necessary to use the McASP module.

##### 20.0.27.1.1 Considerations When Using a McASP

The following is a list of things to be considered for systems using a McASP:

###### 20.0.27.1.1.1 Clocks

For each receive and transmit section:

- External or internal generated bit clock and high frequency clock?
- If internally generated, what is the bit clock speed and the high frequency clock speed?
- Clock polarity?
- External or internal generated frame sync?
- If internally generated, what is frame sync speed?
- Frame sync polarity?
- Frame sync width?
- Transmit and receive sync or asynchronous?

###### 20.0.27.1.1.2 Data Pins

For each pin of each McASP:

- McASP or GPIO?
- Input or output?

###### 20.0.27.1.1.3 Data Format

For each transmit and receive data:

- Internal numeric representation (integer, Q31 fraction)?
- I2S or DIT (transmit only)?
- Time slot delay (0, 1, or 2 bit)?
- Alignment (left or right)?
- Order (MSB first, LSB first)?
- Pad (if yes, pad with what value)?
- Slot size?
- Rotate?
- Mask?

#### 20.0.27.1.1.4 Data Transfers

- Internal: DMA or CPU?
- External: TDM or burst?
- Bus: peripheral configuration bus or DMA bus?

#### 20.0.27.1.2 Transmit/Receive Section Initialization

You must follow the following steps to properly configure the McASP. If external clocks are used, they should be present prior to the following initialization steps.

1. Reset McASP to default values by setting GBLCTL = 0.
2. Configure McASP Audio FIFO. Recall that the Write FIFO and Read FIFO are enabled/disabled independently.
  - (a) Write FIFO:
    - If the Write FIFO will not be enabled, verify that WFIFOCTL.WENA is cleared to 0 (the default value).
    - If the Write FIFO will be enabled, configure WFIFOCTL. Note that WFIFOCTL.WENA should not be set to 1 (enabled) until the other bitfields in this register are configured.
  - (b) Read FIFO:
    - If the Read FIFO will not be enabled, verify that RFIFOCTL.RENA is cleared to 0 (the default value).
    - If the Read FIFO will be enabled, configure RFIFOCTL. Note that RFIFOCTL.RENA should not be set to 1 (enabled) until the other bitfields in this register are configured.
3. Configure all McASP registers except GBLCTL in the following order:
  - (a) Receive registers: RMASK, RFMT, AFSRCTL, ACLKRCTL, AHCLKRCTL, RTDM, RINTCTL, RCLKCHK. If external clocks AHCLKR and/or ACLKR are used, they must be running already for proper synchronization of the GBLCTL register.
  - (b) Transmit registers: XMASK, XFMT, AFSXCTL, ACLKXCTL, AHCLKXCTL, XTDM, XINTCTL, XCLKCHK. If external clocks AHCLKX and/or ACLKX are used, they must be running already for proper synchronization of the GBLCTL register.
  - (c) Serializer registers: SRCTL[n].
  - (d) Global registers: Registers PFUNC, PDIR, DITCTL, DLBCTL, AMUTE. Note that PDIR should only be programmed after the clocks and frames are set up in the steps above. This is because the moment a clock pin is configured as an output in PDIR, the clock pin starts toggling at the rate defined in the corresponding clock control register. Therefore you must ensure that the clock control register is configured appropriately before you set the pin to be an output. A similar argument applies to the frame sync pins.
  - (e) DIT registers: For DIT mode operation, set up registers DITCSRA[n], DITCSRB[n], DITUDRA[n], and DITUDRB[n].
4. Start the respective high-frequency serial clocks AHCLKX and/or AHCLKR. This step is necessary even if external high-frequency serial clocks are used:
  - (a) Take the respective internal high-frequency serial clock divider(s) out of reset by setting the RHCLKRST bit for the receiver and/or the XHCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be held at 0.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

5. Start the respective serial clocks ACLKX and/or ACLKR. This step can be skipped if external serial clocks are used and they are running:
  - (a) Take the respective internal serial clock divider(s) out of reset by setting the RCLKRST bit for the receiver and/or the XCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
6. Setup data acquisition as required:
  - (a) If DMA is used to service the McASP, set up data acquisition as desired and start the DMA in this step, before the McASP is taken out of reset.
  - (b) If CPU interrupt is used to service the McASP, enable the transmit and/ or receive interrupt as required.
  - (c) If CPU polling is used to service the McASP, no action is required in this step.
7. Activate serializers.
  - (a) Before starting, clear the respective transmitter and receiver status registers by writing XSTAT = FFFFh and RSTAT = FFFFh.
  - (b) Take the respective serializers out of reset by setting the RSRCLR bit for the receiver and/or the XSRCLR bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (c) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
8. Verify that all transmit buffers are serviced. Skip this step if the transmitter is not used. Also, skip this step if time slot 0 is selected as inactive (special cases, see [Figure 20-24](#), second waveform). As soon as the transmit serializer is taken out of reset, XDATA in the XSTAT register is set, indicating that XBUF is empty and ready to be serviced. The XDATA status causes a DMA event AXEVT to be generated, and can cause an interrupt AXINT to be generated if it is enabled in the XINTCTL register.
  - (a) If DMA is used to service the McASP, the DMA automatically services the McASP upon receiving AXEVT. Before proceeding in this step, you should verify that the XDATA bit in the XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the DMA.
  - (b) If CPU interrupt is used to service the McASP, interrupt service routine is entered upon the AXINT interrupt. The interrupt service routine should service the XBUF registers. Before proceeding in this step, you should verify that the XDATA bit in XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the CPU.
  - (c) If CPU polling is used to service the McASP, the XBUF registers should be written to in this step.

#### **CAUTION**

The DSP does not support the emulation suspend signal. Therefore, if a data window is open in the Code Composer Studio™ integrated development environment to observe the XRBUF locations, the emulation read from the XRBUF locations causes an undesirable side effect of clearing the RDATA bit in RSTAT. Furthermore, if you write to the XRBUF through the Code Composer Studio™ integrated development environment, the emulation write to the XRBUF locations causes the XDATA bit in XSTAT to be cleared.

9. Release state machines from reset.
  - (a) Take the respective state machine(s) out of reset by setting the RSMRST bit for the receiver and/or the XSMRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

10. Release frame sync generators from reset. Note that it is necessary to release the internal frame sync generators from reset, even if an external frame sync is being used, because the frame sync error detection logic is built into the frame sync generator.
  - (a) Take the respective frame sync generator(s) out of reset by setting the RFRST bit for the receiver, and/or the XFRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
11. Upon the first frame sync signal, McASP transfers begin. The McASP synchronizes to an edge on the frame sync pin, not the level on the frame sync pin. This makes it easy to release the state machine and frame sync generators from reset.
  - (a) For example, if you configure the McASP for a rising edge transmit frame sync, then you do not need to wait for a low level on the frame sync pin before releasing the McASP transmitter state machine and frame sync generators from reset.

### 20.0.27.1.3 *Separate Transmit and Receive Initialization*

In many cases, it is desirable to separately initialize the McASP transmitter and receiver. For example, you may delay the initialization of the transmitter until the type of data coming in on the receiver is recognized. Or a change in the incoming data stream on the receiver may necessitate a reinitialization of the transmitter.

In this case, you may still follow the sequence outlined in [Section 20.0.27.1.2](#), but use it for each section (transmit, receive) individually. The GBLCTL register is aliased to RGBLCTL and XGBLCTL to facilitate separate initialization of transmit and receive sections.

Also, make sure that the initialization or reinitialization sequence follows the guidelines in [Bits With Restrictions on When They May be Changed](#).

### 20.0.27.1.4 *Importance of Reading Back GBLCTL*

In [Section 20.0.27.1.2](#), steps 4b, 5b, 7c, 9b, and 10b state that GBLCTL should be read back until the bits that were written are successfully latched. This is important, because the transmitter and receiver state machines run off of the respective bit clocks, which are typically about tens to hundreds of times slower than the DSP's internal bus clock. Therefore, it takes many cycles between when the DSP writes to GBLCTL (or RGBLCTL and XGBLCTL), and when the McASP actually recognizes the write operation. If you skip this step, then the McASP may never see the reset bits in the global control registers get asserted and deasserted; resulting in an uninitialized McASP.

Therefore, the logic in McASP has been implemented such that once the DSP writes GBLCTL, RGBLCTL, or XGBLCTL, the resulting write is not visible by reading back GBLCTL until the McASP has recognized the change. This typically requires two bit clocks plus two DSP bus clocks to occur.

Also, if the bit clocks can be completely stopped, any software that polls GBLCTL should be implemented with a time-out. If GBLCTL does not have a time-out, and the bit clock stops, the changes written to GBLCTL will not be reflected until the bit clock restarts.

Finally, please note that while RGBLCTL and XGBLCTL allow separate changing of the receive and transmit halves of GBLCTL, they also immediately reflect the updated value (useful for debug purposes). Only GBLCTL can be used for the read back step.

### 20.0.27.1.5 *Synchronous Transmit and Receive Operation (ASYNC = 0)*

When ASYNC = 0 in ACLKXCTL, the transmit and receive sections operate synchronously from the transmit section clock and transmit frame sync signals ([Figure 20-15](#)). The receive section may have a different (but compatible in terms of slot size) data format. Note that when ASYNC = 0, XCLK is automatically inverted to produce RCLK (note the inversion on the ASYNC multiplexer as shown in [Figure 20-16](#)).

When  $ASYNC = 0$ , the transmit and receive sections must share some common settings, since they both use the same clock and frame sync signals:

- $DITEN = 0$  in  $DITCTL$  (TDM mode is enabled)
- The total number of bits per frame must be the same (that is,  $RSSZ \times RMOD$  must equal  $XSSZ \times XMOD$ )
- Both transmit and receive should either be specified as burst or TDM mode, but not mixed
- The settings in  $ACLKCTL$  are irrelevant
- $RCLK$  is an inverted version of  $XCLK$  (note the inversion on the multiplexer labeled “ASYNC” shown in [Figure 20-16](#))
- $FSXM$  must match  $FSRM$
- $FXWID$  must match  $FRWID$

For all other settings, the transmit and receive sections may be programmed independently.

### 20.0.27.1.6 Asynchronous Transmit and Receive Operation ( $ASYNC = 1$ )

When  $ASYNC = 1$  in  $ACLKXCTL$ , the transmit and receive sections operate completely independently and have separate clock and frame sync signals ([Figure 20-15](#), [Figure 20-16](#), and [Figure 20-17](#)). The events generated by each section come asynchronously.

## 20.0.27.2 Transfer Modes

### 20.0.27.2.1 Burst Transfer Mode

The McASP supports a burst transfer mode, which is useful for nonaudio data such as passing control information between two DSPs. Burst transfer mode uses a synchronous serial format similar to the TDM mode. The frame sync generation is not periodic or time-driven as in TDM mode, but data driven, and the frame sync is generated for each data word transferred.

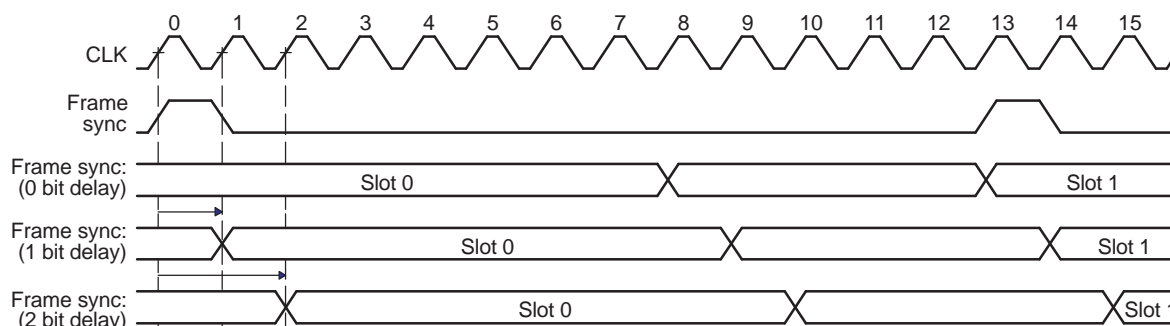
When operating in burst frame sync mode ([Figure 20-23](#)), as specified for transmit ( $XMOD = 0$  in  $AFSXCTL$ ) and receive ( $RMOD = 0$  in  $AFSRCTL$ ), one slot is shifted for each active edge of the frame sync signal that is recognized. Additional clocks after the slot and before the next frame sync edge are ignored.

In burst frame sync mode, the frame sync delay may be specified as 0, 1, or 2 serial clock cycles. This is the delay between the frame sync active edge and the start of the slot. The frame sync signal lasts for a single bit clock duration ( $FRWID = 0$  in  $AFSRCTL$ ,  $FXWID = 0$  in  $AFSXCTL$ ).

For transmit, when generating the transmit frame sync internally, the frame sync begins when the previous transmission has completed and when all the  $XBUF_n$  (for every serializer set to operate as a transmitter) has been updated with new data.

For receive, when generating the receive frame sync internally, frame sync begins when the previous transmission has completed and when all the  $RBUF_n$  (for every serializer set to operate as a receiver) has been read.

**Figure 20-23. Burst Frame Sync Mode**





The control registers must be configured as follows for the burst transfer mode. The burst mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in [Section 20.0.27.1.2](#) to configure this register.
- AMUTE: Not applicable. Leave at default.
- DLBCTL: If loopback mode is desired, configure this register according to [Section 20.0.27.7](#), otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select non-DIT mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to [Section 20.0.27.2](#) and [Section 20.0.27.4](#).
- RFMT/XFMT: Program all fields according to data format desired. See [Section 20.0.27.4](#).
- AFSRCTL/AFSXCTL: Clear **RMOD/XMOD** bits to 0 to indicate burst mode. Clear **FRWID/FXWID** bits to 0 for single bit frame sync duration. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See [Section 20.0.27](#).
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See [Section 20.0.27](#).
- RTDM/XTDM: Program RTDMS0/XTDMS0 to 1 to indicate one active slot only. Leave other fields at default.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Not applicable. Leave at default.
- SRCTLn: Program SRMOD to inactive/transmitter/receiver as desired. DISMOD is not applicable and should be left at default.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.



### 20.0.27.2.2 Time-Division Multiplexed (TDM) Transfer Mode

The McASP time-division multiplexed (TDM) transfer mode supports the TDM format discussed in [Section 20.0.24.1](#).

Transmitting data in the TDM transfer mode requires a minimum set of pins:

- ACLKX - transmit bit clock
- AFSX - transmit frame sync (or commonly called left/right clock)
- One or more serial data pins, AXR[n], whose serializers have been configured to transmit

The transmitter has the option to receive the ACLKX bit clock as an input, or to generate the ACLKX bit clock by dividing down the AHCLKX high-frequency master clock. The transmitter can either generate AHCLKX internally or receive AHCLKX as an input. See [Section 20.0.27.1](#).

Similarly, to receive data in the TDM transfer mode requires a minimum set of pins:

- ACLKR - receive bit clock
- AFSR - receive frame sync (or commonly called left/right clock)
- One or more serial data pins, AXR[n], whose serializers have been configured to receive

The receiver has the option to receive the ACLKR bit clock as an input or to generate the ACLKR bit clock by dividing down the AHCLKR high-frequency master clock. The receiver can either generate AHCLKR internally or receive AHCLKR as an input. See [Section 20.0.27.2](#) and [Section 20.0.27.3](#).

The control registers must be configured as follows for the TDM mode. The TDM mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in [Section 20.0.27.1.2](#) to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: If loopback mode is desired, configure this register according to [Section 20.0.27.7](#), otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select TDM mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to [Section 20.0.27.2](#) and [Section 20.0.27.4](#).
- RFMT/XFMT: Program all fields according to data format desired. See [Section 20.0.27.4](#).
- AFSRCTL/AFSXCTL: Set **RMOD/XMOD** bits to 2-32 for TDM mode. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See [Section 20.0.27](#).
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See [Section 20.0.27](#).
- RTDM/XTDM: Program all fields according to the time slot characteristics desired.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Program all fields according to serializer operation desired.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

#### 20.0.27.2.2.1 TDM Time Slots

TDM mode on the McASP can extend to support multiprocessor applications, with up to 32 time slots per frame. For each of the time slots, the McASP may be configured to participate or to be inactive by configuring XTDM and/or RTDM (this allows multiple DSPs to communicate on the same TDM serial bus).

The TDM sequencer (separate ones for transmit and receive) functions in this mode. The TDM sequencer counts the slots beginning with the frame sync. For each slot, the TDM sequencer checks the respective bit in either XTDM or RTDM to determine if the McASP should transmit/receive in that time slot.

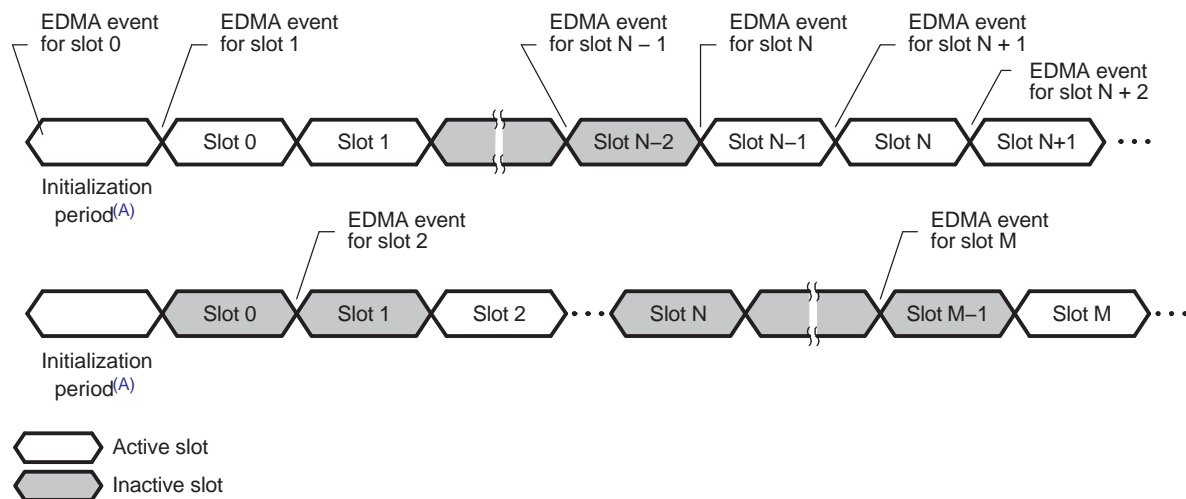
If the transmit/receive bit is active, the McASP functions normally during that time slot; otherwise, the McASP is inactive during that time slot; no update to the buffer occurs, and no event is generated. Transmit pins are automatically set to a high-impedance state, 0, or 1 during that slot, as determined by bit DISMOD in SRCTL[n].

Figure 20-24 shows when the transmit DMA event AXEVT is generated. See Section 20.0.27.3.1 for details on data ready and the initialization period indication. The transmit DMA event for an active time slot (slot N) is generated during the previous time slot (slot N - 1), regardless if the previous time slot (slot N - 1) is active or inactive.

During an active transmit time slot (slot N), if the next time slot (slot N + 1) is configured to be active, the copy from XRBUF[n] to XRSR[n] generates the DMA event for time slot N + 1. If the next time slot (slot N + 1) is configured to be inactive, then the DMA event will be delayed to time slot M - 1. In this case, slot M is the next active time slot. The DMA event for time slot M is generated during the first bit time of slot M - 1.

The receive DMA request generation does not need this capability, since the receive DMA event is generated after data is received in the buffer (looks back in time). If a time slot is disabled, then no data is copied to the buffer for that time slot and no DMA event is generated.

**Figure 20-24. Transmit DMA Event (AXEVT) Generation in TDM Time Slots**



A See Section 20.0.27.1.2, step 7a.

### 20.0.27.2.2.2 Special 384 Slot TDM Mode for Connection to External DIR

The McASP receiver also supports a 384 time slot TDM mode (DIR mode), to support S/PDIF, AES-3, IEC-60958 receiver ICs whose natural block (block corresponds to McASP frame) size is 384 samples. The advantage to using the 384 time slot TDM mode is that interrupts may be generated synchronous to the S/PDIF, AES-3, IEC-60958, such as the last slot interrupt.

The receive TDM time slot register (RTDM) should be programmed to all 1s during reception of a DIR block. Other TDM functionalities (for example, inactive slots) are not supported (only the slot counter counts the 384 subframes in a block).

To receive data in the DIR mode, the following pins are typically needed:

- ACLKR - receive bit clock.
- AFSR - receive frame sync (or commonly called left/right clock). In this mode, AFSR should be connected to a DIR which outputs a start of block signal, instead of LRCLK.
- One or more serial data pins, AXR[n], whose serializers have been configured to receive.

For this special DIR mode, the control registers can be configured just as for TDM mode, except set RMOD in AFSRCTL to 384 to receive 384 time slots.

### 20.0.27.2.3 Digital Audio Interface Transmit (DIT) Transfer Mode

In addition to the TDM and burst transfer modes, which are suitable for transmitting audio data between ICs inside the same system, the digital audio interface transmit (DIT) transfer mode of the McASP also supports transmission of audio data in the S/PDIF, AES-3, or IEC-60958 format. These formats are designed to carry audio data between different systems through an optical or coaxial cable. The DIT mode only applies to serializers configured as transmitters, not receivers. Refer to [Section 20.0.24.2](#) for a description of the S/PDIF format.

#### 20.0.27.2.3.1 Transmit DIT Encoding

The McASP operation in DIT mode is basically identical to the 2 time slot TDM mode, but the data transmitted is output as a biphase mark encoded bit stream, with preamble, channel status, user data, validity, and parity automatically stuffed into the bit stream by the McASP. The McASP includes separate validity bits for even/odd subframes and two 384-bit RAM modules to hold channel status and user data bits.

The transmit TDM time slot register (XTDM) should be programmed to all 1s during DIT mode. TDM functionality is not supported in DIT mode, except that the TDM slot counter counts the DIT subframes.

To transmit data in the DIT mode, the following pins are typically needed:

- AHCLKX - transmit high-frequency master clock
- One or more serial data pins, AXR[n], whose serializers have been configured to transmit

AHCLKX is optional (the internal clock source may be used instead), but if used as a reference, the DSP provides a clock check circuit that continually monitors the AHCLKX input for stability.

If the McASP is configured to transmit in the DIT mode on more than one serial data pin, the bit streams on all pins will be synchronized. In addition, although they will carry unique audio data, they will carry the same channel status, user data, and validity information.

The actual 24-bit audio data must always be in bit positions 23-0 after passing through the first three stages of the transmit format unit.

For left-aligned Q31 data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 010 (rotate right by 8 bits)
- XRVRS = 0 (no bit reversal, LSB first)
- XMASK = FFFF FF00h-FFFF 0000h (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present)
- XPAD = 00 (pad extra bits with 0)

For right-aligned data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 000 (rotate right by 0 bits)
- XRVRS = 0 (no bit reversal, LSB first)
- XMASK = 00FF FFFFh to 0000 FFFFh (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present)
- XPAD = 00 (pad extra bits with 0)

### 20.0.27.2.3.2 Transmit DIT Clock and Frame Sync Generation

The DIT transmitter only works in the following configuration:

- In transmit frame control register (AFSXCTL):
  - Internally-generated transmit frame sync, FSXM = 1
  - Rising-edge frame sync, FSXP = 0
  - Bit-width frame sync, FXWID = 0
  - 384-slot TDM, XMOD = 1 1000 0000b
- In transmit clock control register (ACLKXCTL), ASYNC = 1
- In transmit bitstream format register (XFMT), XSSZ = 1111 (32-bit slot size)

All combinations of AHCLKX and ACLKX are supported.

This is a summary of the register configurations required for DIT mode. The DIT mode specific bit fields are in bold face:

- PFUNC: The data pins must be configured for McASP function. If AHCLKX is used, it must also be configured for McASP function. Other pins can be configured to function as GPIO if desired.
- PDIR: The data pins must be configured as outputs. If AHCLKX is used as an input reference, it should be configured as input. If internal clock source AUXCLK is used as the reference clock, it may be output on the AHCLKX pin by configuring AHCLKX as an output.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable for DIT operation. Leave at default.
- GBLCTL: Follow the initialization sequence in [Section 20.0.27.1.2](#) to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: Not applicable. Loopback is not supported for DIT mode. Leave at default.
- DITCTL: **DITEN** bit must be set to 1 to enable DIT mode. Configure other bits as desired.
- RMASK: Not applicable. Leave at default.
- RFMT: Not applicable. Leave at default.
- AFSRCTL: Not applicable. Leave at default.
- ACLKRCTL: Not applicable. Leave at default.
- AHCLKRCTL: Not applicable. Leave at default.
- RTDM: Not applicable. Leave at default.
- RINTCTL: Not applicable. Leave at default.
- RCLKCHK: Not applicable. Leave at default.
- **XMASK**: Mask desired bits according to the discussion in this section, depending upon left-aligned or right-aligned internal data.
- **XFMT**: **XDATDLY** = 0. **XRVRS** = 0. **XPAD** = 0. **XPBIT** = default (not applicable). **XSSZ** = Fh (32-bit slot). **XBUSEL** = configured as desired. **XROT** bit is configured according to the discussion in this section, either 0 or 8-bit rotate.
- **AFSXCTL**: Configure the bits according to the discussion in this section.
- **ACLKXCTL**: **ASYNC** = 1. Program CLKXDIV bits to obtain the bit clock rate desired. Configure CLKXP and CLKXM bits as desired, because CLKX is not actually used in the DIT protocol.
- **AHCLKXCTL**: Program all fields according to high-frequency clock desired.
- **XTDM**: Set to FFFF FFFFh for all active slots for DIT transfers.
- XINTCTL: Program all fields according to interrupts desired.
- XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Set **SRMOD** = 1 (transmitter) for the DIT pins. DISMOD field is don't care for DIT mode.
- **DITCSRA[n]**, **DITCSRB[n]**: Program the channel status bits as desired.
- **DITUDRA[n]**, **DITUDRB[n]**: Program the user data bits as desired.

### 20.0.27.2.3.3 DIT Channel Status and User Data Register Files

The channel status registers (DITCSRAn and DITCSRbn) and user data registers (DITUDRA<sub>n</sub> and DITUDRB<sub>n</sub>) are not double buffered. Typically the programmer uses one of the synchronizing interrupts, such as last slot, to create an event at a safe time so the register may be updated. In addition, the CPU reads the transmit TDM slot counter to determine which word of the register is being used.

It is a requirement that the software avoid writing to the word of user data and channel status that are being used to encode the current time slot; otherwise, it will be indeterminate whether the old or new data is used to encode the bitstream.

The DIT subframe format is defined in [Section 20.0.24.2.2](#). The channel status information (C) and user data (U) are defined in these DIT control registers:

- DITCSRA0 to DITCSRA5: The 192 bits in these six registers contain the channel status information for the LEFT channel within each frame.
- DITCSRB0 to DITCSRB5: The 192 bits in these six registers contain the channel status information for the RIGHT channel within each frame.
- DITUDRA0 to DITUDRA5: The 192 bits in these six registers contain the user data information for the LEFT channel within each frame.
- DITUDRB0 to DITUDRB5: The 192 bits in these six registers contain the user data information for the RIGHT channel within each frame.

The S/PDIF block format is shown in [Figure 20-11](#). There are 192 frames within a block (frame 0 to frame 191). Within each frame there are two subframes (subframe 1 and 2 for left and right channels, respectively). The channel status and user data information sent on each subframe is summarized in [Table 20-3](#).

### 20.0.27.3 Data Transmission and Reception

The DSP services the McASP by writing data to the XBUF register(s) for transmit operations, and by reading data from the RBUF register(s) for receive operations. The McASP sets status flag and notifies the DSP whenever data is ready to be serviced. [Section 20.0.27.3.1](#) discusses data ready status in detail.

The XBUF and RBUF registers can be accessed through one of the two peripheral ports of the device:

- The DMA port: This port is dedicated for data transfers on the device.
- The peripheral configuration port: This port is used for both data transfers and peripheral configuration control on the device.

[Section 20.0.27.3.2](#) and [Section 20.0.27.3.3](#) discuss how to perform transfers through the DMA bus and the peripheral configuration bus.

Either the CPU or the DMA can be used to service the McASP through any of these two peripheral ports. The CPU and DMA usages are discussed in [Section 20.0.27.3.4](#) and [Section 20.0.27.3.5](#).

**Table 20-3. Channel Status and User Data for Each DIT Block**

Frame	Subframe	Preamble	Channel Status defined in:	User Data defined in:
<b>Defined by DITCSRA0, DITCSRB0, DITUDRA0, DITUDRB0</b>				
0	1 (L)	B	DITCSRA0[0]	DITUDRA0[0]
0	2 (R)	W	DITCSRB0[0]	DITUDRB0[0]
1	1 (L)	M	DITCSRA0[1]	DITUDRA0[1]
1	2 (R)	W	DITCSRB0[1]	DITUDRB0[1]
2	1 (L)	M	DITCSRA0[2]	DITUDRA0[2]
2	2 (R)	W	DITCSRB0[2]	DITUDRB0[2]
...	...	...	...	...
31	1 (L)	M	DITCSRA0[31]	DITUDRA0[31]
31	2 (R)	W	DITCSRB0[31]	DITUDRB0[31]
<b>Defined by DITCSRA1, DITCSRB1, DITUDRA1, DITUDRB1</b>				
32	1 (L)	M	DITCSRA1[0]	DITUDRA1[0]
32	2 (R)	W	DITCSRB1[0]	DITUDRB1[0]
...	...	...	...	...
63	1 (L)	M	DITCSRA1[31]	DITUDRA1[31]
63	2 (R)	W	DITCSRB1[31]	DITUDRB1[31]
<b>Defined by DITCSRA2, DITCSRB2, DITUDRA2, DITUDRB2</b>				
64	1 (L)	M	DITCSRA2[0]	DITUDRA2[0]
64	2 (R)	W	DITCSRB2[0]	DITUDRB2[0]
...	...	...	...	...
95	1 (L)	M	DITCSRA2[31]	DITUDRA2[31]
95	2 (R)	W	DITCSRB2[31]	DITUDRB2[31]
<b>Defined by DITCSRA3, DITCSRB3, DITUDRA3, DITUDRB3</b>				
96	1 (L)	M	DITCSRA3[0]	DITUDRA3[0]
96	2 (R)	W	DITCSRB3[0]	DITUDRB3[0]
...	...	...	...	...
127	1 (L)	M	DITCSRA3[31]	DITUDRA3[31]
127	2 (R)	W	DITCSRB3[31]	DITUDRB3[31]
<b>Defined by DITCSRA4, DITCSRB4, DITUDRA4, DITUDRB4</b>				
128	1 (L)	M	DITCSRA4[0]	DITUDRA4[0]
128	2 (R)	W	DITCSRB4[0]	DITUDRB4[0]
...	...	...	...	...
159	1 (L)	M	DITCSRA4[31]	DITUDRA4[31]
159	2 (R)	W	DITCSRB4[31]	DITUDRB4[31]
<b>Defined by DITCSRA5, DITCSRB5, DITUDRA5, DITUDRB5</b>				
160	1 (L)	M	DITCSRA5[0]	DITUDRA5[0]
160	2 (R)	W	DITCSRB5[0]	DITUDRB5[0]
...	...	...	...	...
191	1 (L)	M	DITCSRA5[31]	DITUDRA5[31]
191	2 (R)	W	DITCSRB5[31]	DITUDRB5[31]

### 20.0.27.3.1 Data Ready Status and Event/Interrupt Generation

#### 20.0.27.3.1.1 Transmit Data Ready

The transmit data ready flag XDATA bit in the XSTAT register reflects the status of the XBUF register. The XDATA flag is set when data is transferred from the XRBUF[n] buffers to the XRSR[n] shift registers, indicating that the XBUF is empty and ready to accept new data from the DSP. This flag is cleared when the XDATA bit is written with a 1, or when all the serializers configured as transmitters are written by the DSP.

Whenever XDATA is set, an DMA event AXEVT is automatically generated to notify the DMA of the XBUF empty status. An interrupt AXINT is also generated if XDATA interrupt is enabled in the XINTCTL register (See Section 20.0.27.5.1 for details).

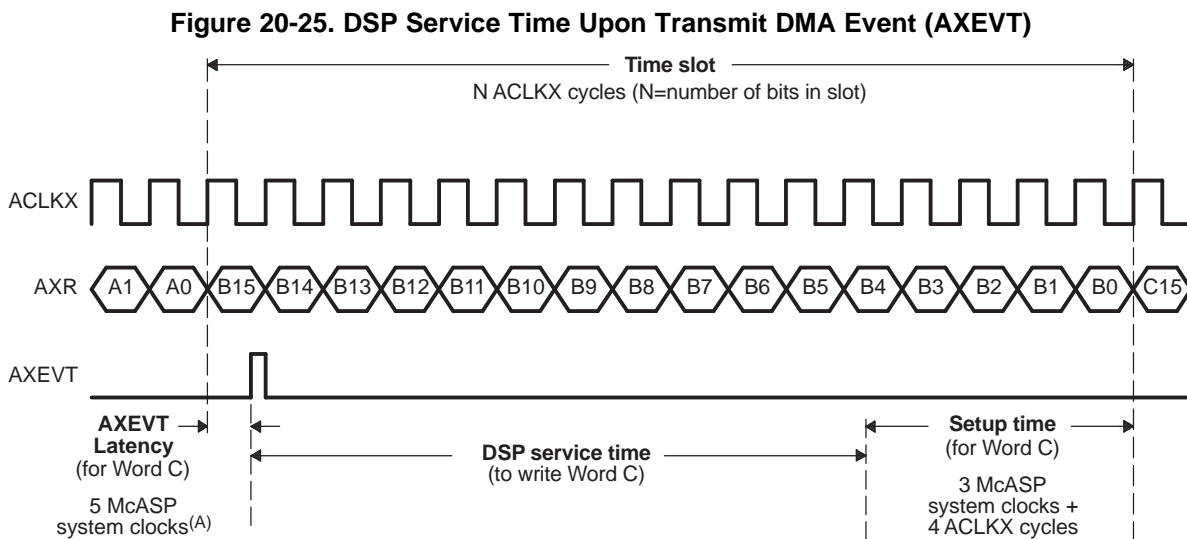
For DMA requests, the McASP does not require XSTAT to be read between DMA events. This means that even if XSTAT already has the XDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Since all serializers act in lockstep, only one DMA event is generated to indicate that all active transmit serializers are ready to be written to with new data.

Figure 20-25 shows the timing details of when AXEVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is transmitted, the McASP sets the XDATA flag and generates an AXEVT event. However, it takes up to 5 McASP system clocks (AXEVT Latency) before AXEVT is active at the McASP boundary. Upon AXEVT, the DSP can begin servicing the McASP by writing Word C into the XBUF (DSP Service Time). The DSP must write Word C into the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum DSP Service Time (Figure 20-25) can be calculated as:

$$\text{DSP Service Time} = \text{Time Slot} - \text{AXEVT Latency} - \text{Setup Time}$$



A This is not the same as AUXCLK. The DSP uses SYSCLK2 as the McASP system clock source.



**Example 20-5. DSP Service Time Calculation for Transmit DMA Event (AXEVT)**

The following is an example to show how to calculate DSP Service Time. Assume the following setup:

- Device: DSP at 300 MHz
- McASP transmits in I2S format at 192 kHz frame rate. Assume slot size is 32 bit

With the above setup, we obtain the following parameters corresponding to [Figure 20-25](#):

- Calculation of McASP system clock cycle:
  - DSP uses SYSCLK2 as the McASP system clock. It runs at 150 MHz (half of device frequency)
  - Therefore, McASP system clock cycle =  $1/150 \text{ MHz} = 6.7 \text{ ns}$
- Calculation of ACLKX clock cycle:
  - This example has two 32-bit slots per frame, for a total of 64 bits per frame
  - ACLKX clock cycle is  $(1/192 \text{ kHz})/64 = 81.4 \text{ ns}$
- Time Slot between AXEVT events:
  - For I2S format, McASP generates two AXEVT events per 192 kHz frame
  - Therefore, Time Slot between AXEVT events is  $(1/192 \text{ kHz})/2 = 2604 \text{ ns}$
- AXEVT Latency
  - = 5 McASP system clocks
  - =  $6.7 \text{ ns} \times 5 = 33.5 \text{ ns}$
- Setup Time
  - = 3 McASP system clocks + 4 ACLKX cycles
  - =  $(6.7 \text{ ns} \times 3) + (81.4 \text{ ns} \times 4)$
  - = 345.7 ns
- DSP Service Time
  - = Time Slot - AXEVT Latency - Setup Time
  - =  $2604 \text{ ns} - 33.5 \text{ ns} - 345.7 \text{ ns}$
  - = 2225 ns



### 20.0.27.3.1.2 Receive Data Ready

Similarly, the receive data ready flag RDATA bit in the RSTAT reflects the status of the RBUF register. The RDATA flag is set when data is transferred from the XRSR[n] shift registers to the XRBUF[n] buffers, indicating that the RBUF contains received data and is ready to have the DSP read the data. This flag is cleared when the RDATA bit is written with a 1, or when all the serializers configured as receivers are read.

Whenever RDATA is set, an DMA event AREVT is automatically generated to notify the DMA of the RBUF ready status. An interrupt ARINT is also generated if RDATA interrupt is enabled in the RINTCTL register (See Section 20.0.27.5.2 for details).

For DMA requests, the McASP does not require RSTAT to be read between DMA events. This means that even if RSTAT already has the RDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

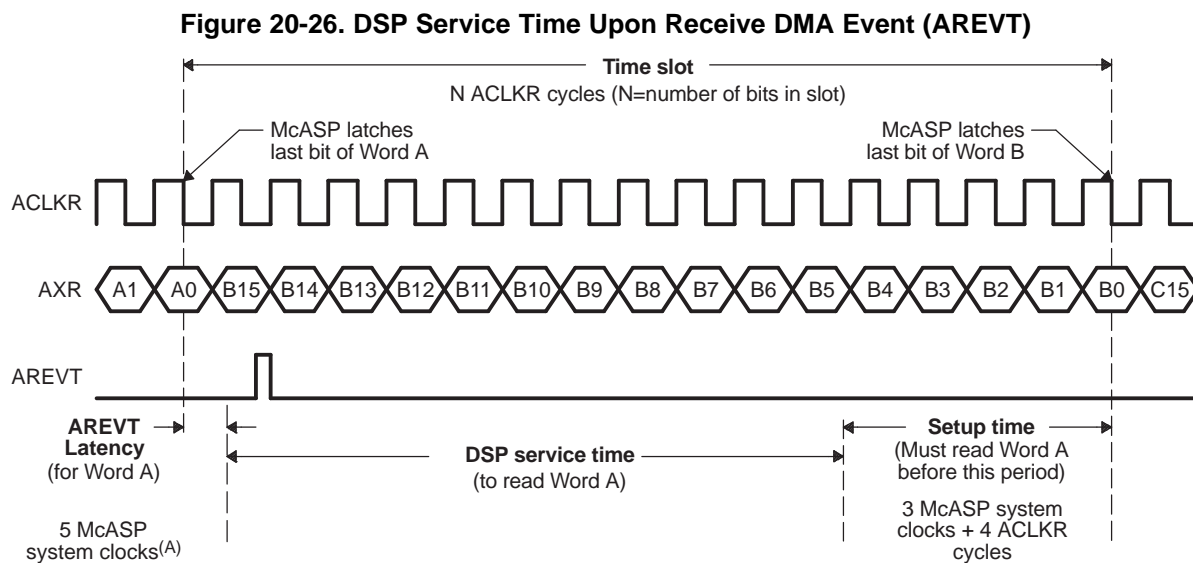
Since all serializers act in lockstep, only one DMA event is generated to indicate that all active receive serializers are ready to receive new data.

Figure 20-26 shows the timing details of when AREVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is received, the McASP sets the RDATA flag and generates an AREVT event. However, it takes up to 5 McASP system clocks (AREVT Latency) before AREVT is active at the McASP boundary. Upon AREVT, the DSP can begin servicing the McASP by reading Word A from the RBUF (DSP Service Time). The DSP must read Word A from the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum DSP Service Time (Figure 20-26) can be calculated as:

$$\text{DSP Service Time} = \text{Time Slot} - \text{AREVT Latency} - \text{Setup Time}$$

The DSP Service Time calculation for receive is similar to the calculation for transmit. See Example 20-5 for DSP Service Time calculation using transmit as an example.



A This is not the same as AUXCLK. The DSP uses SYSCLK2 as the McASP system clock source.

### 20.0.27.3.2 Transfers through the DMA Port

#### CAUTION

To perform internal transfers through the DMA port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

Typically, you will access the McASP XRBUF registers through the DMA port. To access through the DMA port, simply have the CPU or DMA access the XRBUF through its DMA port location. See your device-specific data manual for the exact memory address. Through the DMA port, the DMA/CPU can service all the serializers through a single address. The McASP automatically cycles through the appropriate serializers.

For transmit operations through the DMA port, the DMA/CPU should write to the same XBUF DMA port address to service all of the active transmit serializers. In addition, the DMA/CPU should write to the XBUF for all active transmit serializers in incremental (although not necessarily consecutive) order. For example, if serializers 0, 4, 5, and 7 are set up as active transmitters, the DMA/CPU should write to the XBUF DMA port address four times with data for serializers 0, 4, 5, and 7 upon each transmit data ready event. This exact servicing order must be followed so that data appears in the appropriate serializers.

Similarly, for receive operations through the DMA port, the DMA/CPU should read from the same RBUF DMA port address to service all of the active receive serializers. In addition, reads from the active receive serializers through the DMA port return data in incremental (although not necessarily consecutive) order. For example, if serializers 1, 2, 3, and 6 are set up as active receivers, the DMA/CPU should read from the RBUF DMA port address four times to obtain data for serializers 1, 2, 3, and 6 in this exact order, upon each receive data ready event.

When transmitting, the DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition ([Section 20.0.27.6.2](#)). Similarly, when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do so results in a buffer overrun condition ([Section 20.0.27.6.3](#)).

To perform internal transfers through the DMA port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers.

### 20.0.27.3.3 Transfers Through the Peripheral Configuration Bus

#### CAUTION

The DSP does not support the emulation suspend signal. Therefore, if a data window is open in the Code Composer Studio™ integrated development environment to observe the XRBUF locations, the emulation read from the XRBUF locations causes an undesirable side effect of clearing the RDATA bit in RSTAT. Furthermore, if you write to the XRBUF through the Code Composer Studio™ integrated development environment, the emulation write to the XRBUF locations causes the XDATA bit in XSTAT to be cleared.

To perform internal transfers through the peripheral configuration bus, set XBUSEL/RBUSEL bit to 1 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

In this method, the DMA/CPU accesses the XRBUF through the peripheral configuration bus address. The exact XRBUF address for any particular serializer is determined by adding the offset for that particular serializer to the base address for the particular McASP (found in the device-specific data manual). XRBUF for the serializers configured as transmitters is given the name XBUF $n$ . For example, the XRBUF associated with transmit serializer 2 is named XBUF2. Similarly, XRBUF for the serializers configured as receivers is given the name RBUF $n$ .

Accessing the XRBUFF through the DMA port is different because the CPU/DMA only needs to access one single address. When accessing through the peripheral configuration bus, the CPU/DMA must provide the exact XBUF $n$  or RBUF $n$  address for each access.

When transmitting, DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition (Section 20.0.27.6.2). Similarly when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do so results in a buffer overrun condition (Section 20.0.27.6.3).

#### 20.0.27.3.4 Using the CPU for McASP Servicing

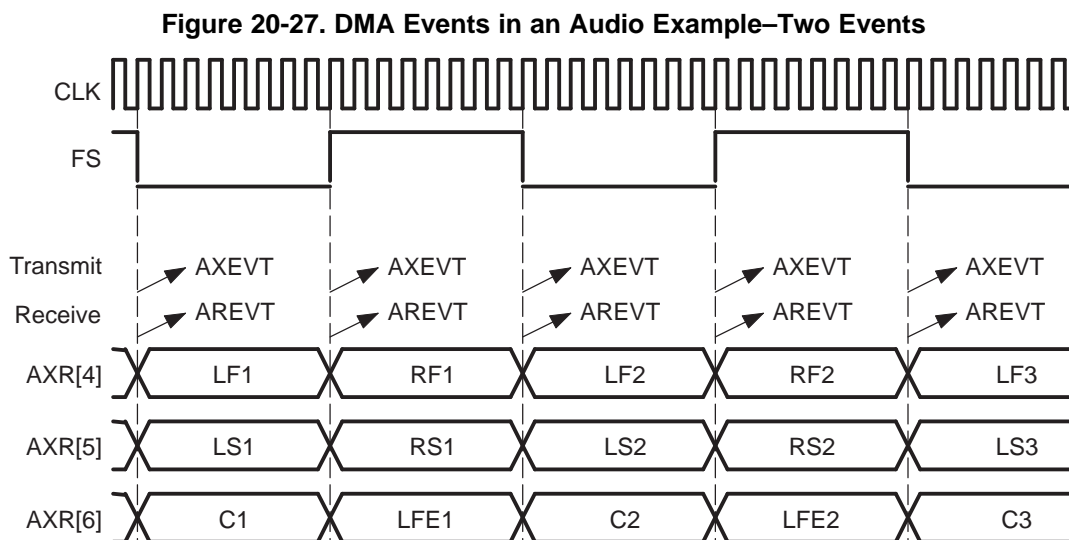
The CPU can be used to service the McASP through interrupt (upon AXINT/ARINT interrupts) or through polling the XDATA bit in the XSTAT register. As discussed in Section 20.0.27.3.2 and Section 20.0.27.3.3, the CPU can access either through the DMA port or through the peripheral configuration port.

To use the CPU to service the McASP through interrupts, the XSTAT/RSTAT bit must be enabled in the respective XINTCTL/RINTCTL registers, to generate interrupts AXINT/ARINT to the CPU upon data ready.

#### 20.0.27.3.5 Using the DMA for McASP Servicing

The most typical scenario is to use the DMA to service the McASP through the DMA port, although the DMA can also service the McASP through the peripheral configuration port. Use AXEVT/AREVT that is triggered upon each XDATA/RDATA transition from 0 to 1.

Figure 20-27 shows an example audio system with six audio channels (LF, RF, LS, RS, C, and LFE) transmitted from three AXR[n] pins on the McASP and shows when events AXEVT and AREVT are triggered.



In Figure 20-27, a DMA event AXEVT/AREVT is triggered on each time slot. In the example, AXEVT is triggered for each of the transmit audio channel time slot (time slot for channels LF, LS, and C; and time slot for channels RF, RS, LFE). Similarly, AREVT is triggered for each of the receive audio channel time slot. This allows for the use of a single DMA to transmit all audio channels, and a single DMA to receive all audio channels.

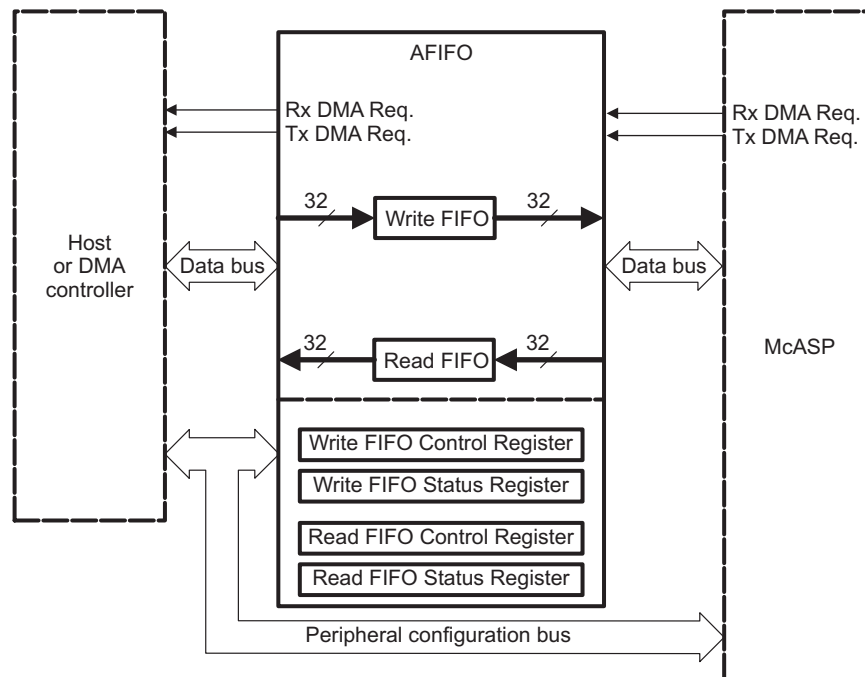
Note the difference between DMA event generation and the CPU interrupt generation. DMA events are generated automatically upon data ready; whereas CPU interrupt generation needs to be enabled in the XINTCTL/RINTCTL register.

## McASP Audio FIFO (AFIFO)

The AFIFO contains two FIFOs: one Read FIFO (RFIFO), and one Write FIFO (WFIFO). To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See [Figure 20-28](#) for a high-level block diagram of the AFIFO.

The AFIFO may be enabled/disabled and configured via the WFIFOCTL and RFIFOCTL registers. Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the McASP (see [Section 20.0.27.1.2](#) for details).

**Figure 20-28. McASP Audio FIFO (AFIFO) Block Diagram**



### AFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the McASP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the McASP generates transmit DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Write FIFO is enabled, transmit DMA requests from the McASP are sent to the AFIFO, which in turn generates transmit DMA requests to the host/DMA controller.

If the Write FIFO is enabled, upon a transmit DMA request from the McASP, the WFIFO writes *WNUMDMA* 32-bit words to the McASP if and when there are at least *WNUMDMA* words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied. At that point, it writes *WNUMDMA* words to the McASP. (See description for WFIFOCTL.WNUMDMA in [Section 20.1.45](#).)

If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded.

Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

## Transmit DMA Event Pacer

The AFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the host or DMA controller is reduced.

If the Write FIFO has space to accept *WNUMEVT* 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once *WNUMEVT* words have been written to the FIFO, it checks again to see if there is space for *WNUMEVT* 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled.

Note that if transmit DMA event pacing is desired, *WFIFOCTL.WNUMEVT* should be set to a non-zero integer multiple of the value in *WFIFOCTL.WNUMDMA*. If transmit DMA event pacing is not desired, then the value in *WFIFOCTL.WNUMEVT* should be set equal to the value in *WFIFOCTL.WNUMDMA*.

## AFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from McASP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the McASP generates receive DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Read FIFO is enabled, receive DMA requests from the McASP are sent to the AFIFO, which in turn generates receive DMA requests to the host/DMA controller.

If the Read FIFO is enabled and the McASP makes a receive DMA request, the RFIFO reads *RNUMDMA* 32-bit words from the McASP, if and when the RFIFO has space for *RNUMDMA* words. If it does not, the RFIFO waits until this condition has been satisfied; at that point, it reads *RNUMDMA* words from the McASP. (See description for *RFIFOCTL.RNUMDMA* in [Section 20.1.47](#).)

If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than *RNUMEVT* words, those words will be read correctly, emptying the FIFO.

## Receive DMA Event Pacer

The AFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced.

If the Read FIFO contains at least *RNUMEVT* 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once *RNUMEVT* 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another *RNUMEVT* words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty.

Note that if receive DMA event pacing is desired, *RFIFOCTL.RNUMEVT* should be set to a non-zero integer multiple of the value in *RFIFOCTL.RNUMDMA*. If receive DMA event pacing is not desired, then the value in *RFIFOCTL.RNUMEVT* should be set equal to the value in *RFIFOCTL.RNUMDMA*.

## Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

## 20.0.27.4 Formatter

### 20.0.27.4.1 Transmit Bit Stream Data Alignment

The McASP transmitter supports serial formats of:

- Slot (or Time slot) size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size  $\leq$  Slot size
- Alignment: when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order: order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the transmit bitstream format register (XFMT):

- XRVRS: bit reverse (1) or no bit reverse (0)
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- XSSZ: transmit slot size of 8, 12, 16, 20, 24, 28, or 32 bits

XSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the XROT field.

[Table 20-4](#) and [Figure 20-29](#) show the XRVRS and XROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size (SLOT in [Table 20-4](#)) and word size (WORD in [Table 20-4](#)) options are multiples of 4, since the transmit rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The transmit bit mask/pad unit operates on data as an initial step of the transmit format unit (see [Figure 20-20](#)), and the data is aligned in the same representation as it is written to the transmitter by the DSP (typically Q31 or integer).

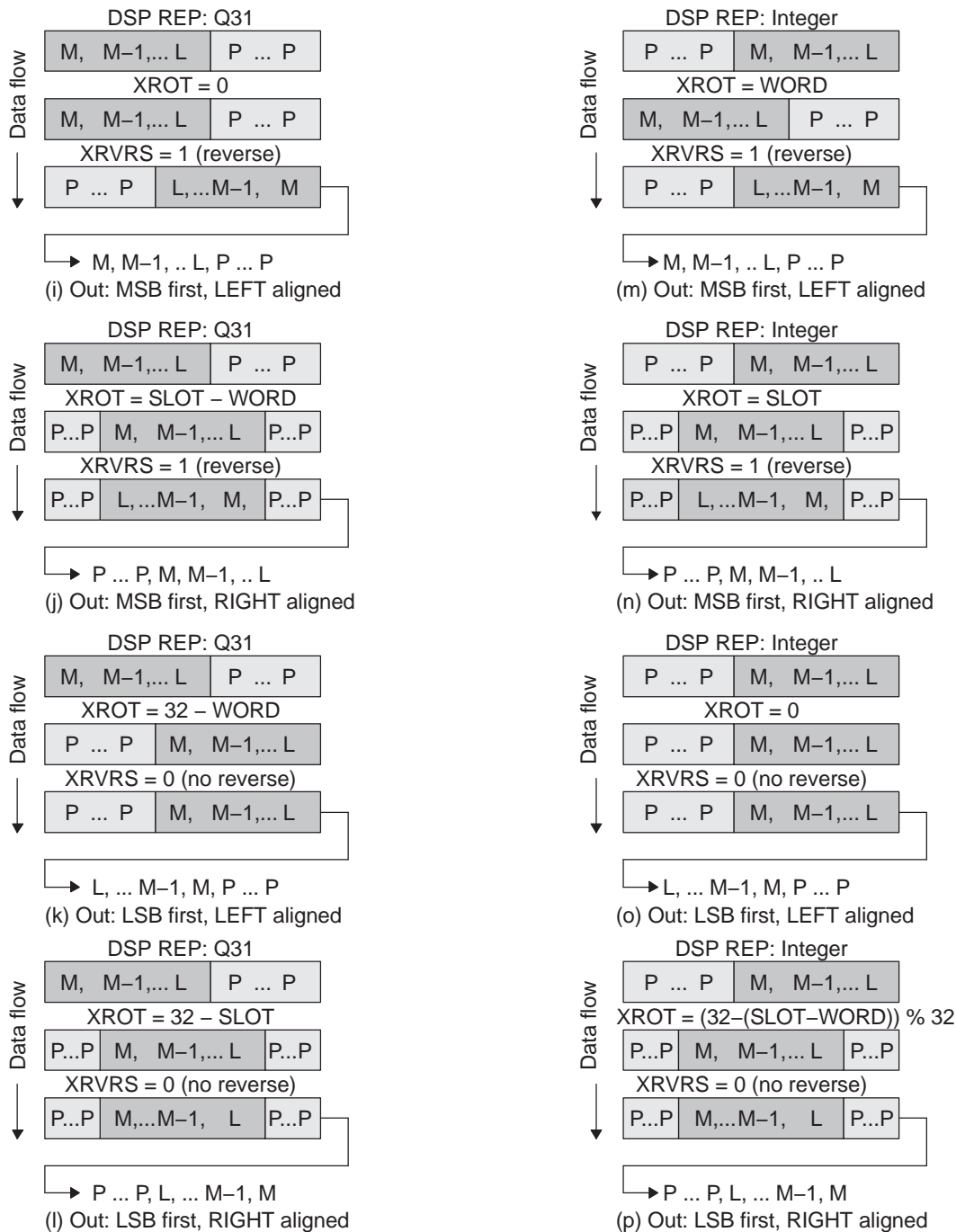
**Table 20-4. Transmit Bitstream Data Alignment**

Figure 20-29	Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	XFMT Bit	
				XROT <sup>(1)</sup>	XRVRS
(a) <sup>(2)</sup>	MSB first	Left aligned	Q31 fraction	0	1
(b)	MSB first	Right aligned	Q31 fraction	SLOT - WORD	1
(c)	LSB first	Left aligned	Q31 fraction	32 - WORD	0
(d)	LSB first	Right aligned	Q31 fraction	32 - SLOT	0
(e) <sup>(2)</sup>	MSB first	Left aligned	Integer	WORD	1
(f)	MSB first	Right aligned	Integer	SLOT	1
(g)	LSB first	Left aligned	Integer	0	0
(h)	LSB first	Right aligned	Integer	(32 - (SLOT - WORD)) % 32	0

<sup>(1)</sup> WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

<sup>(2)</sup> To transmit in I2S format, use MSB first, left aligned, and also select XDATDLY = 01 (1 bit delay)

Figure 20-29. Data Flow Through Transmit Format Unit





### 20.0.27.4.2 Receive Bit Stream Data Alignment

The McASP receiver supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the receive bitstream format register (RFMT):

- RRVRS: bit reverse (1) or no bit reverse (0)
- RROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- RSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits

RSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the RROT field.

Table 20-5 and Figure 20-30 show the RRVRS and RROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size and word size options are multiples of 4; since the receive rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The receive bit mask/pad unit operates on data as the final step of the receive format unit (see Figure 20-19), and the data is aligned in the same representation as it is read from the receiver by the DSP (typically Q31 or integer).

**Table 20-5. Receive Bitstream Data Alignment**

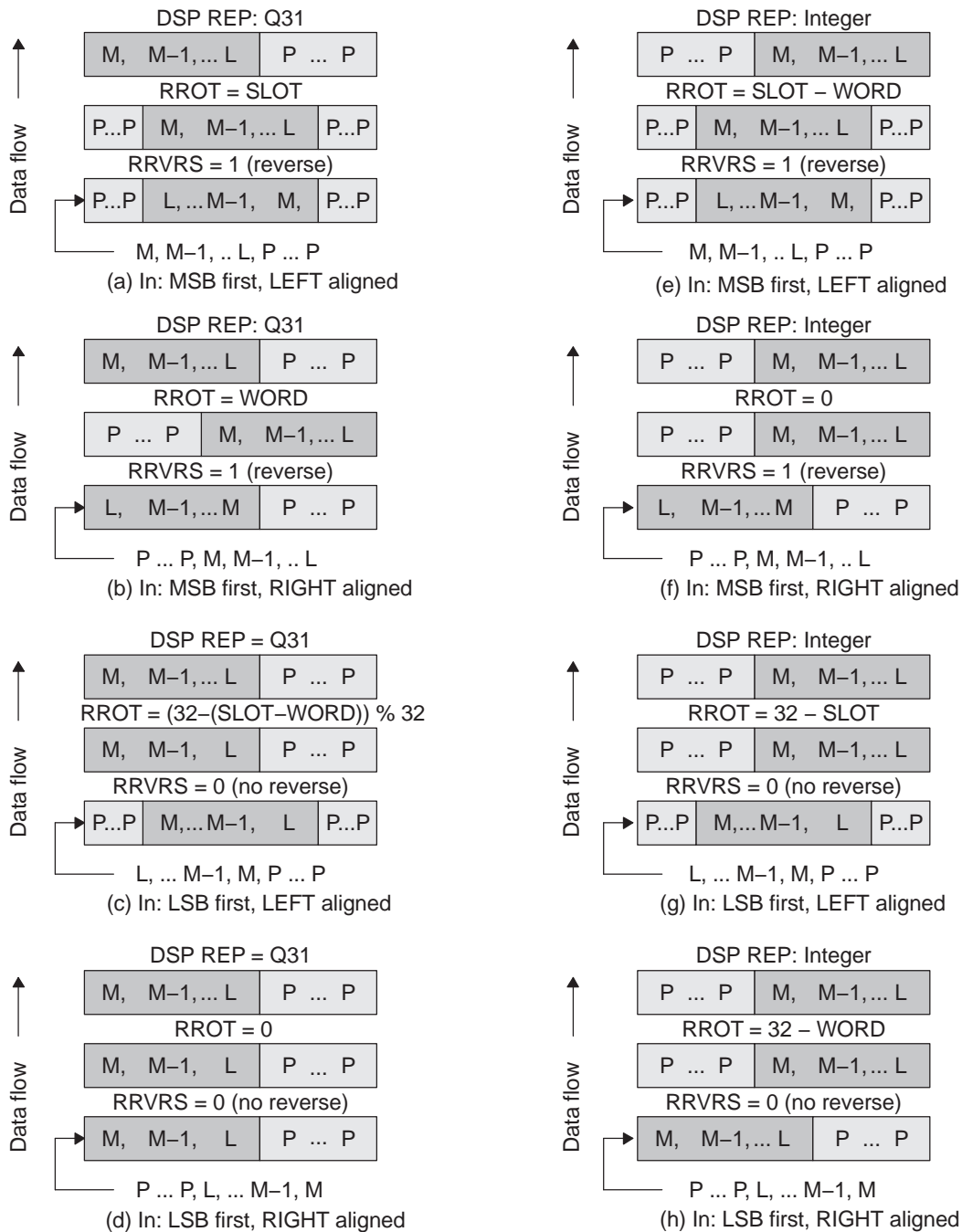
Figure 20-30	Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	RFMT Bit	
				RROT <sup>(1)</sup>	RRVRS
(a) <sup>(2)</sup>	MSB first	Left aligned	Q31 fraction	SLOT	1
(b)	MSB first	Right aligned	Q31 fraction	WORD	1
(c)	LSB first	Left aligned	Q31 fraction	$(32 - (\text{SLOT} - \text{WORD})) \% 32$	0
(d)	LSB first	Right aligned	Q31 fraction	0	0
(e) <sup>(2)</sup>	MSB first	Left aligned	Integer	SLOT - WORD	1
(f)	MSB first	Right aligned	Integer	0	1
(g)	LSB first	Left aligned	Integer	32 - SLOT	0
(h)	LSB first	Right aligned	Integer	32 - WORD	0

<sup>(1)</sup> WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

<sup>(2)</sup> To transmit in I2S format, select MSB first, left aligned, and also select RDATDLY = 01 (1 bit delay)



**Figure 20-30. Data Flow Through Receive Format Unit**



## 20.0.27.5 Interrupts

### 20.0.27.5.1 Transmit Data Ready Interrupt

The transmit data ready interrupt (XDATA) is generated if XDATA is 1 in the XSTAT register and XDATA is also enabled in XINTCTL. [Section 20.0.27.3.1](#) provides details on when XDATA is set in the XSTAT register.

A transmit start of frame interrupt (XSTAFRM) is triggered by the recognition of transmit frame sync. A transmit last slot interrupt (XLAST) is a qualified version of the data ready interrupt (XDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data requested belonging to the last slot (the slot that just ended was next-to-last TDM slot, current slot is last slot).

### 20.0.27.5.2 Receive Data Ready Interrupt

The receive data ready interrupt (RDATA) is generated if RDATA is 1 in the RSTAT register and RDATA is also enabled in RINTCTL. [Section 20.0.27.3.2](#) provides details on when RDATA is set in the RSTAT register.

A receiver start of frame interrupt (RSTAFRM) is triggered by the recognition of a receiver frame sync. A receiver last slot interrupt (RLAST) is a qualified version of the data ready interrupt (RDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data in the buffer come from the last TDM time slot (the slot that just ended was last TDM slot).

### 20.0.27.5.3 Error Interrupts

Upon detection, the following error conditions generate interrupt flags:

- In the receive status register (RSTAT):
  - Receiver overrun (ROVRN)
  - Unexpected receive frame sync (RSYNCERR)
  - Receive clock failure (RCKFAIL)
  - Receive DMA error (RDMAERR)
- In the transmit status register (XSTAT):
  - Transmit underrun (XUNDRN)
  - Unexpected transmit frame sync (XSYNCERR)
  - Transmit clock failure (XCKFAIL)
  - Transmit DMA error (XDMAERR)

Each interrupt source also has a corresponding enable bit in the receive interrupt control register (RINTCTL) and transmit interrupt control register (XINTCTL). If the enable bit is set in RINTCTL or XINTCTL, an interrupt is requested when the interrupt flag is set in RSTAT or XSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

### 20.0.27.5.4 Audio Mute (AMUTE) Function

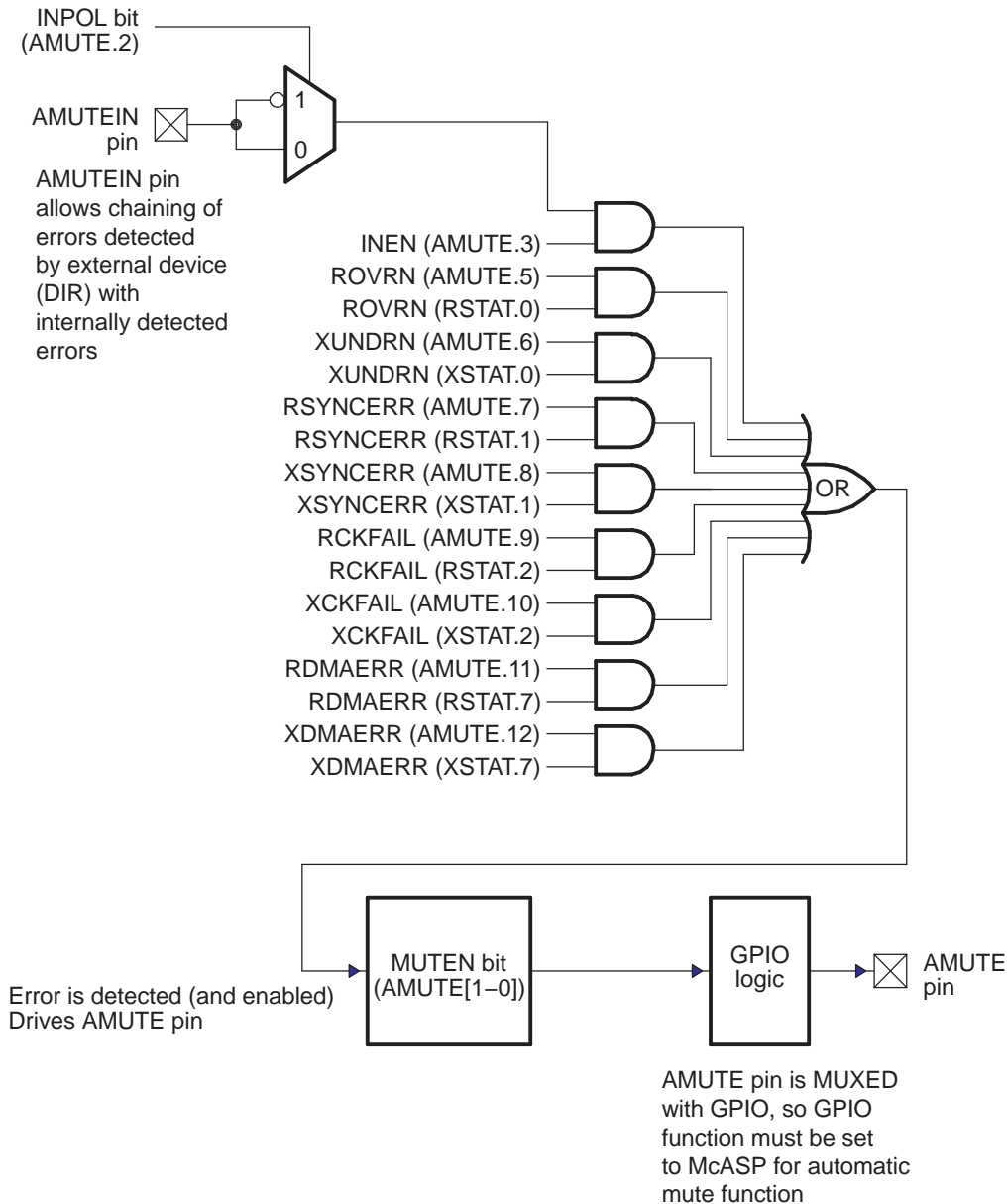
The McASP includes an automatic audio mute function ([Figure 20-31](#)) that asserts in hardware the AMUTE device pin to a preprogrammed output state, as selected by the MUTEN bit in the audio mute control register (AMUTE). The AMUTE device pin is asserted when one of the interrupt flags is set or an external device issues an error signal on the AMUTEIN input. Typically, the AMUTEIN input is shared with a device pin.

The AMUTEIN input allows the on-chip logic to consider a mute input from other devices in the system, so that all errors may be considered. The AMUTEIN input has a programmable polarity to allow it to adapt to different devices, as selected by the INPOL bit in AMUTE, and it must be enabled explicitly.

In addition to the external AMUTEIN input, the AMUTE device pin output may be asserted when one of the error interrupt flags is set and its mute function is enabled in AMUTE.

When one or more of the errors is detected and enabled, the AMUTE device pin is driven to an active state that is selected by MUTEN in AMUTE. The active polarity of the AMUTE device pin is programmable by MUTEN (and the inactive polarity is the opposite of the active polarity). The AMUTE device pin remains driven active until software clears all the error interrupt flags that are enabled to mute, and until the AMUTEIN is inactive.

**Figure 20-31. Audio Mute (AMUTE) Block Diagram**



### 20.0.27.5.5 Multiple Interrupts

This only applies to interrupts and not to DMA requests. The following terms are defined:

- **Active Interrupt Request:** a flag in RSTAT or XSTAT is set and the interrupt is enabled in RINTCTL or XINTCTL.
- **Outstanding Interrupt Request:** An interrupt request has been issued on one of the McASP transmit/receive interrupt ports, but that request has not yet been serviced.
- **Serviced:** The CPU writes to RSTAT or XSTAT to clear one or more of the active interrupt request flags.

The first interrupt request to become active for the transmitter with the interrupt flag set in XSTAT and the interrupt enabled in XINTCTL generates a request on the McASP transmit interrupt port AXINT.

If more than one interrupt request becomes active in the same cycle, a single interrupt request is generated on the McASP transmit interrupt port. Subsequent interrupt requests that become active while the first interrupt request is outstanding do not immediately generate a new request pulse on the McASP transmit interrupt port.

The transmit interrupt is serviced with the CPU writing to XSTAT. If any interrupt requests are active after the write, a new request is generated on the McASP transmit interrupt port.

The receiver operates in a similar way, but using RSTAT, RINTCTL, and the McASP receive interrupt port ARINT.

One outstanding interrupt request is allowed on each port, so a transmit and a receive interrupt request may both be outstanding at the same time.

### 20.0.27.6 Error Handling and Management

To support the design of a robust audio system, the McASP includes error-checking capability for the serial protocol, data underrun, and data overrun. In addition, the McASP includes a timer that continually measures the high-frequency master clock every 32 AHCLKX/AHCLKR clock cycles. The timer value can be read to get a measurement of the clock frequency and has a minimum and maximum range setting that can set an error flag if the master clock goes out of a specified range.

Upon the detection of any one or more errors (software selectable), or the assertion of the AMUTEIN input pin, the AMUTE output pin may be asserted to a high or low level to immediately mute the audio output. In addition, an interrupt may be generated if desired, based on any one or more of the error sources.

#### 20.0.27.6.1 Unexpected Frame Sync Error

An unexpected frame sync occurs when:

- In burst mode, when the next active edge of the frame sync occurs early such that the current slot will not be completed by the time the next slot is scheduled to begin.
- In TDM mode, a further constraint is that the frame sync must occur exactly during the correct bit clock (not a cycle earlier or later) and only before slot 0. An unexpected frame sync occurs if this condition is not met.

When an unexpected frame sync occurs, there are two possible actions depending upon when the unexpected frame sync occurs:

1. Early: An early unexpected frame sync occurs when the McASP is in the process of completing the current frame and a new frame sync is detected (not including overlap that occurs due to a 1 or 2 bit frame sync delay). When an early unexpected frame sync occurs:
  - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
  - Current frame is not resynchronized. The number of bits in the current frame is completed. The next frame sync, which occurs after the current frame is completed, will be resynchronized.

2. Late: A late unexpected frame sync occurs when there is a gap or delay between the last bit of the previous frame and the first bit of the next frame. When a late unexpected frame sync occurs (as soon as the gap is detected):
  - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
  - Resynchronization occurs upon the arrival of the next frame sync.

Late frame sync is detected the same way in both burst mode and TDM mode; however, in burst mode, late frame sync is not meaningful and its interrupt enable should not be set.

#### **20.0.27.6.2 Buffer Underrun Error - Transmitter**

A buffer underrun can only occur for serializers programmed to be transmitters. A buffer underrun occurs when the serializer is instructed by the transmit state machine to transfer data from XRBUF[n] to XRSR[n], but XRBUF[n] has not yet been written with new data since the last time the transfer occurred. When this occurs, the transmit state machine sets the XUNDRN flag.

An underrun is checked only once per time slot. The XUNDRN flag is set when an underrun condition occurs. Once set, the XUNDRN flag remains set until the DSP explicitly writes a 1 to the XUNDRN bit to clear the XUNDRN bit.

In DIT mode, a pair of BMC zeros is shifted out when an underrun occurs (four bit times at  $128 \times f_s$ ). By shifting out a pair of zeros, a clock may be recovered on the receiver. To recover, reset the McASP and start again with the proper initialization.

In TDM mode, during an underrun case, a long stream of zeros are shifted out causing the DACs to mute. To recover, reset the McASP and start again with the proper initialization.

#### **20.0.27.6.3 Buffer Overrun Error - Receiver**

A buffer overrun can only occur for serializers programmed to be receivers. A buffer overrun occurs when the serializer is instructed to transfer data from XRSR[n] to XRBUF[n], but XRBUF[n] has not yet been read by either the DMA or the DSP. When this occurs, the receiver state machine sets the ROVRN flag. However, the individual serializer writes over the data in the XRBUF[n] register (destroying the previous sample) and continues shifting.

An overrun is checked only once per time slot. The ROVRN flag is set when an overrun condition occurs. It is possible that an overrun occurs on one time slot but then the DSP catches up and does not cause an overrun on the following time slots. However, once the ROVRN flag is set, it remains set until the DSP explicitly writes a 1 to the ROVRN bit to clear the ROVRN bit.

#### **20.0.27.6.4 DMA Error - Transmitter**

A transmit DMA error, as indicated by the XDMAERR flag in the XSTAT register, occurs when the DMA (or CPU) writes more words to the DMA port of the McASP than it should. For each DMA event, the DMA should write exactly as many words as there are serializers enabled as transmitters.

XDMAERR indicates that the DMA (or CPU) wrote too many words to the McASP for a given transmit DMA event. Writing too few words results in a transmit underrun error setting XUNDRN in XSTAT.

While XDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP transmitter and the DMA to resynchronize them.

### 20.0.27.6.5 DMA Error - Receiver

A receive DMA error, as indicated by the RDMAERR flag in the RSTAT register, occurs when the DMA (or CPU) reads more words from the DMA port of the McASP than it should. For each DMA event, the DMA should read exactly as many words as there are serializers enabled as receivers.

RDMAERR indicates that the DMA (or CPU) read too many words from the McASP for a given receive DMA event. Reading too few words results in a receiver overrun error setting ROVRN in RSTAT.

While RDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP receiver and the DMA to resynchronize them.

### 20.0.27.6.6 Clock Failure Detection

#### 20.0.27.6.6.1 Clock-Failure Check Startup

It is expected, initially, that the clock-failure circuits will generate an error until at least one measurement has been taken. Therefore, the clock failure interrupts, clock switch, and mute functions should not immediately be enabled, but be enabled only after a specific startup procedure. The startup procedure is:

1. For the transmit clock failure check:
  - (a) Configure transmit clock failure detect logic (XMIN, XMAX, XPS) in the transmit clock check control register (XCLKCHK).
  - (b) Clear transmit clock failure flag (XCKFAIL) in the transmit status register (XSTAT).
  - (c) Wait until first measurement is taken (> 32 AHCLKX clock periods).
  - (d) Verify no clock failure is detected.
  - (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
  - (f) After the transmit clock is measured and falls within the acceptable range, the following may be enabled:
    - (i) transmit clock failure interrupt enable bit (XCKFAIL) in the transmitter interrupt control register (XINTCTL)
    - (ii) transmit clock failure detect autoswitch enable bit (XCKFAILSW) in the transmit clock check control register (XCLKCHK)
    - (iii) mute option (XCKFAIL) in the mute control register (AMUTE)
2. For the receive clock failure check:
  - (a) Configure receive clock failure detect logic (RMIN, RMAX, RPS) in the receive clock check control register (RCLKCHK).
  - (b) Clear receive clock failure flag (RCKFAIL) in the receive status register (RSTAT).
  - (c) Wait until first measurement is taken (> 32 AHCLKR clock periods).
  - (d) Verify no clock failure is detected.
  - (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
  - (f) After the receive clock is measured and falls within the acceptable range, the following may be enabled:
    - (i) receive clock failure interrupt enable bit (RCKFAIL) in the receiver interrupt control register (RINTCTL)
    - (ii) mute option (RCKFAIL) in the mute control register (AMUTE)

### 20.0.27.6.6.2 Transmit Clock Failure Check and Recovery

The transmit clock failure check circuit (Figure 20-32) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKX). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKX) periods, and stores the count in XCNT of the transmit clock check control register (XCLKCHK) every 32 high rate serial clock cycles.

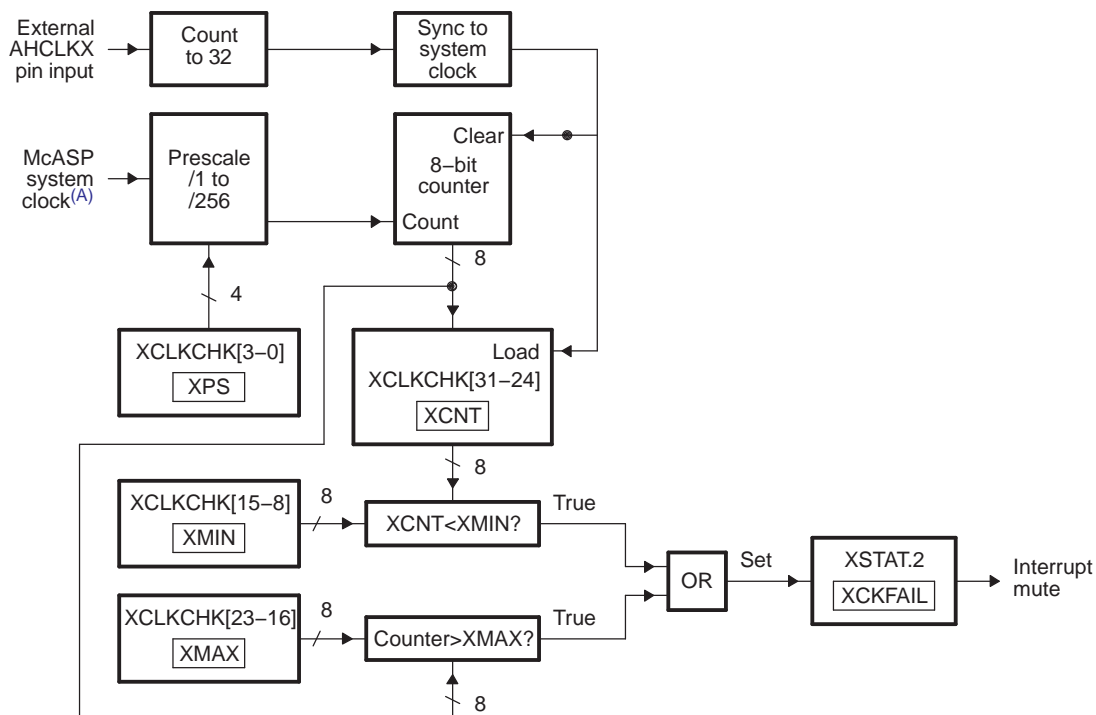
The logic compares the count against a user-defined minimum allowable boundary (XMIN), and automatically flags an interrupt (XCKFAIL in XSTAT) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than XMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (XMAX). This is in case the external clock completely stops, so that the counter value is not copied to XCNT. An out-of-range maximum condition occurs when the count is greater than XMAX. Note that the XMIN and XMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected, or that the audio source has changed and a new sample rate is being used.

In order for the transmit clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKX is internally generated or externally sourced.

If a clock failure is detected, the transmit clock failure flag (XCKFAIL) in XSTAT is set. This causes an interrupt, if the transmit clock failure interrupt enable bit (XCKFAIL) in XINTCTL is set.

**Figure 20-32. Transmit Clock Failure Detection Circuit Block Diagram**



A This is not the same as AUXCLK. The DSP uses SYSCLK2 as the McASP system clock.

### 20.0.27.6.6.3 Receive Clock Failure Check and Recovery

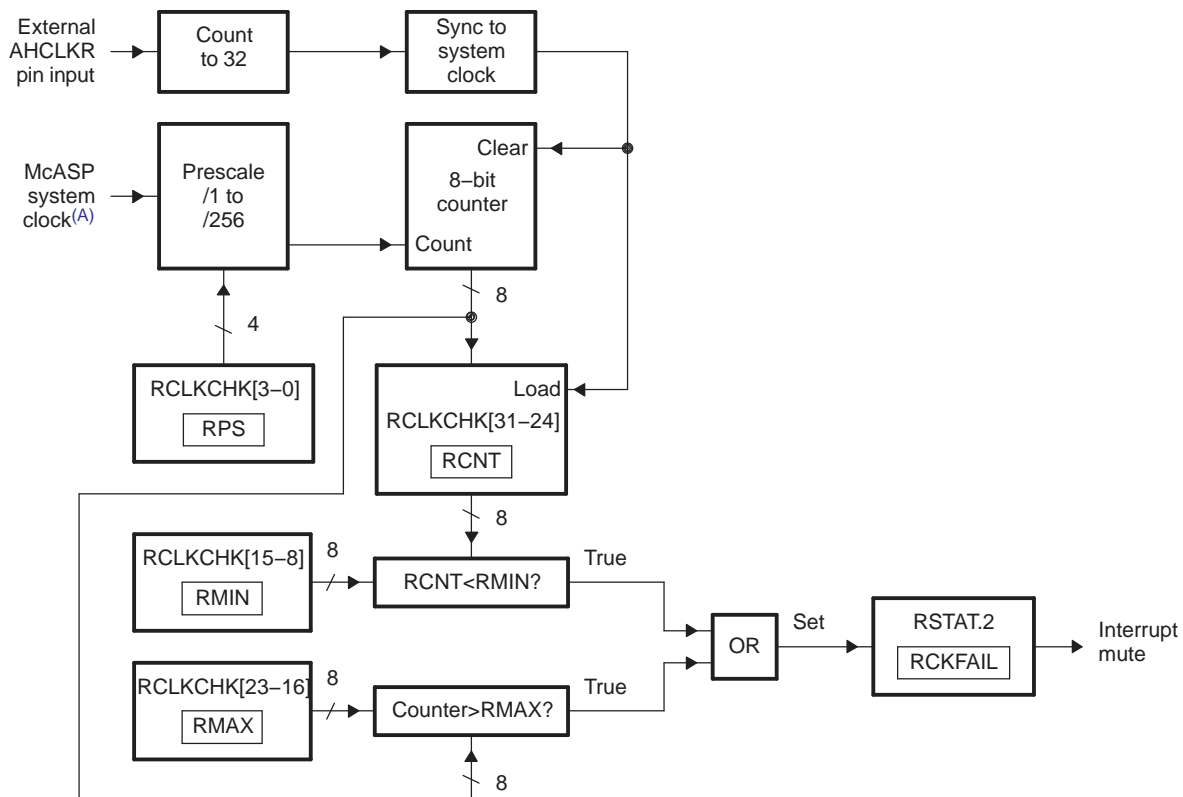
The receive clock failure check circuit (Figure 20-33) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKR). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKR) periods, and stores the count in RCNT of the receive clock check control register (RCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (RMIN) and automatically flags an interrupt (RCKFAIL in RSTAT) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than RMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (RMAX). This is in case the external clock completely stops, so that the counter value is not copied to RCNT. An out-of-range maximum condition occurs when the count is greater than RMAX. Note that the RMIN and RMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

In order for the receive clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKR is internally generated or externally sourced.

Figure 20-33. Receive Clock Failure Detection Circuit Block Diagram



A This is not the same as AUXCLK. The DSP uses SYSCLK2 as the McASP system clock source.



### 20.0.27.7 Loopback Modes

The McASP features a digital loopback mode (DLB) that allows testing of the McASP code in TDM mode with a single DSP device. In loopback mode, output of the transmit serializers is connected internally to the input of the receive serializers. Therefore, you can check the receive data against the transmit data to ensure that the McASP settings are correct. Digital loopback mode applies to TDM mode only (2 to 32 slots in a frame). It does not apply to DIT mode (XMOD = 180h) or burst mode (XMOD = 0).

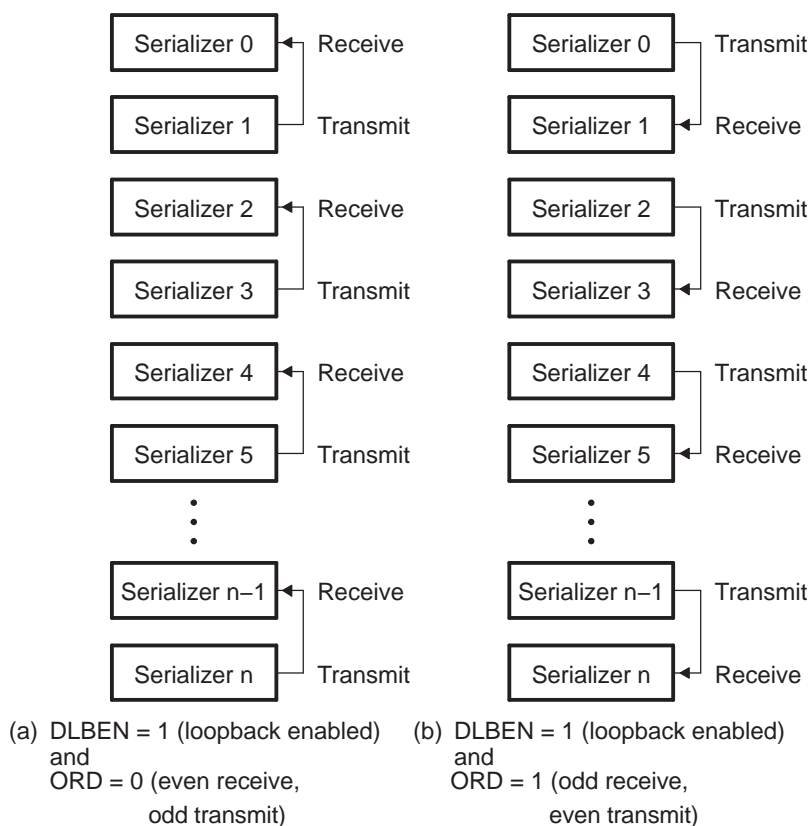
Figure 20-34 shows the basic logical connection of the serializers in loopback mode. Two types of loopback connections are possible, selected by the ORD bit in the digital loopback control register (DLBCTL) as follows:

- ORD = 0: Outputs of odd serializers are connected to inputs of even serializers. If this mode is selected, you should configure odd serializers to be transmitters and even serializers to be receivers.
- ORD = 1: Outputs of even serializers are connected to inputs of odd serializers. If this mode is selected, you should configure even serializers to be transmitters and odd serializers to be receivers.

Data can be externally visible at the I/O pin of the transmit serializer if the pin is configured as a McASP output pin by setting the corresponding PFUNC bit to 0 and PDIR bit to 1.

In loopback mode, the transmit clock and frame sync are used by both the transmit and receive sections of the McASP. The transmit and receive sections operate synchronously. This is achieved by setting the MODE bit of the DLBCTL register to 01b and the ASYNC bit of the ACLKXCTL register to 0.

**Figure 20-34. Serializers in Loopback Mode**



### 20.0.27.7.1 Loopback Mode Configurations

This is a summary of the settings required for digital loopback mode for TDM format:

- The DLBEN bit in DLBCTL must be set to 1 to enable loopback mode.
- The MODE bits in DLBCTL must be set to 01b for both the transmit and receive sections to use the transmit clock and frame sync generator.
- The ORD bit in DLBCTL must be programmed appropriately to select odd or even serializers to be transmitters or receivers. The corresponding serializers must be configured accordingly.
- The ASYNC bit in ACLKXCTL must be cleared to 0 to ensure synchronous transmit and receive operations.
- RMOD field in AFSRCTL and XMOD field in AFSXCTL must be set to 2h to 20h to indicate TDM mode. Loopback mode does not apply to DIT or burst mode.

### 20.0.28 Reset Considerations

The McASP has two reset sources: software reset and hardware reset.

#### 20.0.28.1 Software Reset Considerations

The transmitter and receiver portions of the McASP may be put in reset through the global control register (GBLCTL). Note that a valid serial clock must be supplied to the desired portion of the McASP (transmit and/or receive) in order to assert the software reset bits in GBLCTL. See [Section 20.0.27.1.2](#) for details on how to ensure reset has occurred.

The entire McASP module may also be reset through the Power and Sleep Controller (PSC). Note that from the McASP perspective, this reset appears as a hardware reset to the entire module.

#### 20.0.28.2 Hardware Reset Considerations

When the McASP is reset due to device reset, the entire serial port (including the transmitter and receiver state machines, and other registers) is reset.

### 20.0.29 EDMA Event Support

The McASP-related EDMA events are shown in [Table 20-6](#).

**Table 20-6. EDMA Events - McASP**

Channel	Event Name	Event Description
0	AREVT0	McASP0 Receive Event
1	AXEVT0	McASP0 Transmit Event

### 20.0.30 Power Management

The McASP can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

## 20.1 Registers

Control registers for the McASP are summarized in [Table 20-7](#). The control registers are accessed through the peripheral configuration port. The receive buffer registers (RBUF) and transmit buffer registers (XBUF) can also be accessed through the DMA port, as listed in [Table 20-8](#). See your device-specific data manual for the memory address of these registers.

Control registers for the McASP Audio FIFO (AFIFO) are summarized in [Table 20-9](#). Note that the AFIFO Write FIFO (WFIFO) and Read FIFO (RFIFO) have independent control and status registers. The AFIFO control registers are accessed through the peripheral configuration port. See your device-specific data manual for the memory address of these registers.

**Table 20-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port**

Offset	Acronym	Register Description	Section
0h	REV	Revision identification register	<a href="#">Section 20.1.2</a>
10h	PFUNC	Pin function register	<a href="#">Section 20.1.3</a>
14h	PDIR	Pin direction register	<a href="#">Section 20.1.4</a>
18h	PDOUT	Pin data output register	<a href="#">Section 20.1.5</a>
1Ch	PDIN	Read returns: Pin data input register	<a href="#">Section 20.1.6</a>
1Ch	PDSET	Writes affect: Pin data set register (alternate write address: PDOUT)	<a href="#">Section 20.1.7</a>
20h	PDCLR	Pin data clear register (alternate write address: PDOUT)	<a href="#">Section 20.1.8</a>
44h	GBLCTL	Global control register	<a href="#">Section 20.1.9</a>
48h	AMUTE	Audio mute control register	<a href="#">Section 20.1.10</a>
4Ch	DLBCTL	Digital loopback control register	<a href="#">Section 20.1.11</a>
50h	DITCTL	DIT mode control register	<a href="#">Section 20.1.12</a>
60h	RGBLCTL	Receiver global control register: Alias of GBLCTL, only receive bits are affected - allows receiver to be reset independently from transmitter	<a href="#">Section 20.1.13</a>
64h	RMASK	Receive format unit bit mask register	<a href="#">Section 20.1.14</a>
68h	RFMT	Receive bit stream format register	<a href="#">Section 20.1.15</a>
6Ch	AFSRCTL	Receive frame sync control register	<a href="#">Section 20.1.16</a>
70h	ACLKRCTL	Receive clock control register	<a href="#">Section 20.1.17</a>
74h	AHCLKRCTL	Receive high-frequency clock control register	<a href="#">Section 20.1.18</a>
78h	RTDM	Receive TDM time slot 0-31 register	<a href="#">Section 20.1.19</a>
7Ch	RINTCTL	Receiver interrupt control register	<a href="#">Section 20.1.20</a>
80h	RSTAT	Receiver status register	<a href="#">Section 20.1.21</a>
84h	RSLOT	Current receive TDM time slot register	<a href="#">Section 20.1.22</a>
88h	RCLKCHK	Receive clock check control register	<a href="#">Section 20.1.23</a>
8Ch	REVTCTL	Receiver DMA event control register	<a href="#">Section 20.1.24</a>
A0h	XGBLCTL	Transmitter global control register. Alias of GBLCTL, only transmit bits are affected - allows transmitter to be reset independently from receiver	<a href="#">Section 20.1.25</a>
A4h	XMASK	Transmit format unit bit mask register	<a href="#">Section 20.1.26</a>
A8h	XFMT	Transmit bit stream format register	<a href="#">Section 20.1.27</a>
ACh	AFSXCTL	Transmit frame sync control register	<a href="#">Section 20.1.28</a>
B0h	ACLKXCTL	Transmit clock control register	<a href="#">Section 20.1.29</a>
B4h	AHCLKXCTL	Transmit high-frequency clock control register	<a href="#">Section 20.1.30</a>
B8h	XTDM	Transmit TDM time slot 0-31 register	<a href="#">Section 20.1.31</a>
BCh	XINTCTL	Transmitter interrupt control register	<a href="#">Section 20.1.32</a>
C0h	XSTAT	Transmitter status register	<a href="#">Section 20.1.33</a>
C4h	XSLOT	Current transmit TDM time slot register	<a href="#">Section 20.1.34</a>
C8h	XCLKCHK	Transmit clock check control register	<a href="#">Section 20.1.35</a>
CCh	XEVTCTL	Transmitter DMA event control register	<a href="#">Section 20.1.36</a>

**Table 20-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Description</b>	<b>Section</b>
100h	DITCSRA0	Left (even TDM time slot) channel status register (DIT mode) 0	<a href="#">Section 20.1.38</a>
104h	DITCSRA1	Left (even TDM time slot) channel status register (DIT mode) 1	<a href="#">Section 20.1.38</a>
108h	DITCSRA2	Left (even TDM time slot) channel status register (DIT mode) 2	<a href="#">Section 20.1.38</a>
10Ch	DITCSRA3	Left (even TDM time slot) channel status register (DIT mode) 3	<a href="#">Section 20.1.38</a>
110h	DITCSRA4	Left (even TDM time slot) channel status register (DIT mode) 4	<a href="#">Section 20.1.38</a>
114h	DITCSRA5	Left (even TDM time slot) channel status register (DIT mode) 5	<a href="#">Section 20.1.38</a>
118h	DITCSRB0	Right (odd TDM time slot) channel status register (DIT mode) 0	<a href="#">Section 20.1.39</a>
11Ch	DITCSRB1	Right (odd TDM time slot) channel status register (DIT mode) 1	<a href="#">Section 20.1.39</a>
120h	DITCSRB2	Right (odd TDM time slot) channel status register (DIT mode) 2	<a href="#">Section 20.1.39</a>
124h	DITCSRB3	Right (odd TDM time slot) channel status register (DIT mode) 3	<a href="#">Section 20.1.39</a>
128h	DITCSRB4	Right (odd TDM time slot) channel status register (DIT mode) 4	<a href="#">Section 20.1.39</a>
12Ch	DITCSRB5	Right (odd TDM time slot) channel status register (DIT mode) 5	<a href="#">Section 20.1.39</a>
130h	DITUDRA0	Left (even TDM time slot) channel user data register (DIT mode) 0	<a href="#">Section 20.1.40</a>
134h	DITUDRA1	Left (even TDM time slot) channel user data register (DIT mode) 1	<a href="#">Section 20.1.40</a>
138h	DITUDRA2	Left (even TDM time slot) channel user data register (DIT mode) 2	<a href="#">Section 20.1.40</a>
13Ch	DITUDRA3	Left (even TDM time slot) channel user data register (DIT mode) 3	<a href="#">Section 20.1.40</a>
140h	DITUDRA4	Left (even TDM time slot) channel user data register (DIT mode) 4	<a href="#">Section 20.1.40</a>
144h	DITUDRA5	Left (even TDM time slot) channel user data register (DIT mode) 5	<a href="#">Section 20.1.40</a>
148h	DITUDRB0	Right (odd TDM time slot) channel user data register (DIT mode) 0	<a href="#">Section 20.1.41</a>
14Ch	DITUDRB1	Right (odd TDM time slot) channel user data register (DIT mode) 1	<a href="#">Section 20.1.41</a>
150h	DITUDRB2	Right (odd TDM time slot) channel user data register (DIT mode) 2	<a href="#">Section 20.1.41</a>
154h	DITUDRB3	Right (odd TDM time slot) channel user data register (DIT mode) 3	<a href="#">Section 20.1.41</a>
158h	DITUDRB4	Right (odd TDM time slot) channel user data register (DIT mode) 4	<a href="#">Section 20.1.41</a>
15Ch	DITUDRB5	Right (odd TDM time slot) channel user data register (DIT mode) 5	<a href="#">Section 20.1.41</a>
180h	SRCTL0	Serializer control register 0	<a href="#">Section 20.1.37</a>
184h	SRCTL1	Serializer control register 1	<a href="#">Section 20.1.37</a>
188h	SRCTL2	Serializer control register 2	<a href="#">Section 20.1.37</a>
18Ch	SRCTL3	Serializer control register 3	<a href="#">Section 20.1.37</a>
190h	SRCTL4	Serializer control register 4	<a href="#">Section 20.1.37</a>
194h	SRCTL5	Serializer control register 5	<a href="#">Section 20.1.37</a>
198h	SRCTL6	Serializer control register 6	<a href="#">Section 20.1.37</a>
19Ch	SRCTL7	Serializer control register 7	<a href="#">Section 20.1.37</a>
1A0h	SRCTL8	Serializer control register 8	<a href="#">Section 20.1.37</a>
1A4h	SRCTL9	Serializer control register 9	<a href="#">Section 20.1.37</a>
1A8h	SRCTL10	Serializer control register 10	<a href="#">Section 20.1.37</a>
1ACh	SRCTL11	Serializer control register 11	<a href="#">Section 20.1.37</a>
1B0h	SRCTL12	Serializer control register 12	<a href="#">Section 20.1.37</a>
1B4h	SRCTL13	Serializer control register 13	<a href="#">Section 20.1.37</a>
1B8h	SRCTL14	Serializer control register 14	<a href="#">Section 20.1.37</a>
1BCh	SRCTL15	Serializer control register 15	<a href="#">Section 20.1.37</a>

**Table 20-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port (continued)**

Offset	Acronym	Register Description	Section
200h	XBUF0 <sup>(1)</sup>	Transmit buffer register for serializer 0	<a href="#">Section 20.1.42</a>
204h	XBUF1 <sup>(1)</sup>	Transmit buffer register for serializer 1	<a href="#">Section 20.1.42</a>
208h	XBUF2 <sup>(1)</sup>	Transmit buffer register for serializer 2	<a href="#">Section 20.1.42</a>
20Ch	XBUF3 <sup>(1)</sup>	Transmit buffer register for serializer 3	<a href="#">Section 20.1.42</a>
210h	XBUF4 <sup>(1)</sup>	Transmit buffer register for serializer 4	<a href="#">Section 20.1.42</a>
214h	XBUF5 <sup>(1)</sup>	Transmit buffer register for serializer 5	<a href="#">Section 20.1.42</a>
218h	XBUF6 <sup>(1)</sup>	Transmit buffer register for serializer 6	<a href="#">Section 20.1.42</a>
21Ch	XBUF7 <sup>(1)</sup>	Transmit buffer register for serializer 7	<a href="#">Section 20.1.42</a>
220h	XBUF8 <sup>(1)</sup>	Transmit buffer register for serializer 8	<a href="#">Section 20.1.42</a>
224h	XBUF9 <sup>(1)</sup>	Transmit buffer register for serializer 9	<a href="#">Section 20.1.42</a>
228h	XBUF10 <sup>(1)</sup>	Transmit buffer register for serializer 10	<a href="#">Section 20.1.42</a>
22Ch	XBUF11 <sup>(1)</sup>	Transmit buffer register for serializer 11	<a href="#">Section 20.1.42</a>
230h	XBUF12 <sup>(1)</sup>	Transmit buffer register for serializer 12	<a href="#">Section 20.1.42</a>
234h	XBUF13 <sup>(1)</sup>	Transmit buffer register for serializer 13	<a href="#">Section 20.1.42</a>
238h	XBUF14 <sup>(1)</sup>	Transmit buffer register for serializer 14	<a href="#">Section 20.1.42</a>
23Ch	XBUF15 <sup>(1)</sup>	Transmit buffer register for serializer 15	<a href="#">Section 20.1.42</a>
280h	RBUF0 <sup>(2)</sup>	Receive buffer register for serializer 0	<a href="#">Section 20.1.43</a>
284h	RBUF1 <sup>(2)</sup>	Receive buffer register for serializer 1	<a href="#">Section 20.1.43</a>
288h	RBUF2 <sup>(2)</sup>	Receive buffer register for serializer 2	<a href="#">Section 20.1.43</a>
28Ch	RBUF3 <sup>(2)</sup>	Receive buffer register for serializer 3	<a href="#">Section 20.1.43</a>
290h	RBUF4 <sup>(2)</sup>	Receive buffer register for serializer 4	<a href="#">Section 20.1.43</a>
294h	RBUF5 <sup>(2)</sup>	Receive buffer register for serializer 5	<a href="#">Section 20.1.43</a>
298h	RBUF6 <sup>(2)</sup>	Receive buffer register for serializer 6	<a href="#">Section 20.1.43</a>
29Ch	RBUF7 <sup>(2)</sup>	Receive buffer register for serializer 7	<a href="#">Section 20.1.43</a>
2A0h	RBUF8 <sup>(2)</sup>	Receive buffer register for serializer 8	<a href="#">Section 20.1.43</a>
2A4h	RBUF9 <sup>(2)</sup>	Receive buffer register for serializer 9	<a href="#">Section 20.1.43</a>
2A8h	RBUF10 <sup>(2)</sup>	Receive buffer register for serializer 10	<a href="#">Section 20.1.43</a>
2ACh	RBUF11 <sup>(2)</sup>	Receive buffer register for serializer 11	<a href="#">Section 20.1.43</a>
2B0h	RBUF12 <sup>(2)</sup>	Receive buffer register for serializer 12	<a href="#">Section 20.1.43</a>
2B4h	RBUF13 <sup>(2)</sup>	Receive buffer register for serializer 13	<a href="#">Section 20.1.43</a>
2B8h	RBUF14 <sup>(2)</sup>	Receive buffer register for serializer 14	<a href="#">Section 20.1.43</a>
2BCh	RBUF15 <sup>(2)</sup>	Receive buffer register for serializer 15	<a href="#">Section 20.1.43</a>

<sup>(1)</sup> Writes to XRBUF[n] by way of XBUF<sub>n</sub> by the CPU/EDMA can only occur through the peripheral configuration port when XBUSEL = 1 in XFMT.

<sup>(2)</sup> Reads from XRBUF[n] by way of RBUF<sub>n</sub> by the CPU/EDMA can only occur through the peripheral configuration port when RBUSEL = 1 in RFMT.

**Table 20-8. McASP Registers Accessed by CPU/EDMA Through DMA Port**

Offset <sup>(1)</sup>	Access	Acronym	Register Description
2000h	Read Accesses	RBUF	Receive buffer DMA port address. Cycles through receive serializers, skipping over transmit serializers and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. Reads from XRBUF[n] by way of RBUF by the CPU/EDMA can only occur through the DMA port when RBUSEL = 0 in RFMT.
2000h	Write Accesses	XBUF	Transmit buffer DMA port address. Cycles through transmit serializers, skipping over receive and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. Writes to XRBUF[n] by way of XBUF by the CPU/EDMA can only occur through the DMA port when XBUSEL = 0 in XFMT.

<sup>(1)</sup> RBUF and XBUF are at the same address location. Reads access RBUF and writes access XBUF.

**Table 20-9. McASP AFIFO Registers Accessed Through Peripheral Configuration Port<sup>(1)</sup>**

Offset	Acronym	Register Description	Section
1000h	AFIFOREV	AFIFO revision identification register	<a href="#">Section 20.1.44</a>
1010h	WFIFOCTL	Write FIFO control register	<a href="#">Section 20.1.45</a>
1014h	WFIFOSTS	Write FIFO status register	<a href="#">Section 20.1.46</a>
1018h	RFIFOCTL	Read FIFO control register	<a href="#">Section 20.1.47</a>
101Ch	RFIFOSTS	Read FIFO status register	<a href="#">Section 20.1.48</a>

<sup>(1)</sup> The AFIFO cannot be used with the peripheral configuration port. Only the DMA port has access to the AFIFO.

### 20.1.1 Register Bit Restrictions

Some bit fields (see [Bits With Restrictions on When They May be Changed](#)) have restrictions on when they may be changed. These restrictions take the form of certain registers that must be asserted in GBLCTL. Once these registers have been asserted, the user may then, and only then, change the desired bit field.

#### Bits With Restrictions on When They May be Changed

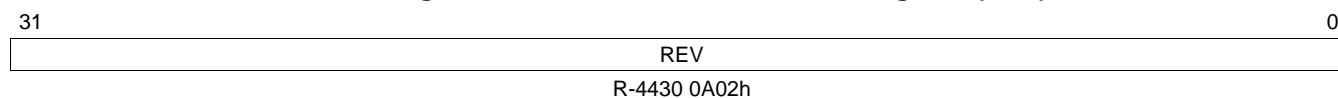
To Change Register:	To Change Bit Field:	... these registers must be asserted in GBLCTL									
		HCLKRRST	RGRST	RSRCLR	RSMRST	RFRST	HCLKXRST	XGRST	XSRCLR	XSMRST	XFRST
DITCTL	DITEN									x	x
XFMT	XSSZ									x	
XFMT	XDATDLY				x					x	
RFMT	RSSZ				x						
RFMT	RDATDLY				x						
AFSXCTL	FSXP									x	x
AFSXCTL	FSXM									x	x
AFSXCTL	FXWID									x	x
AFSXCTL	XMOD									x	x
AFSRCTL	FSRP				x	x					
AFSRCTL	FSRM				x	x					
AFSRCTL	FRWID				x	x					
AFSRCTL	RMOD				x	x					
ACLKXCTL	CLKXDIV							x	x	x	x
ACLKXCTL	CLKXM								x	x	x
ACLKXCTL	ASYNC				x	x					
ACLKXCTL	CLKXP								x	x	x
ACLKRCTL	CLKRDIV		x	x	x	x					
ACLKRCTL	CLKRM			x	x	x					
ACLKRCTL	CLKRP			x	x	x					

**Bits With Restrictions on When They May be Changed (continued)**

To Change Register:	To Change Bit Field:	... these registers must be asserted in GBLCTL									
		HCLKRRST	RGRST	RSRCLR	RSMRST	FRST	HCLKXRST	XGRST	XSRCLR	XSMRST	XFRST
AHCLKXCTL	HCLKXDIV						x	x	x	x	x
AHCLKXCTL	HCLKXP						x	x	x	x	x
AHCLKXCTL	HCLKXM						x	x	x	x	x
AHCLKRCTL	HCLKRDIV	x	x	x	x	x					
AHCLKRCTL	HCLKRP	x	x	x	x	x					
AHCLKRCTL	HCLKRM	x	x	x	x	x					
DLBCTL	DLBEN			x	x	x			x	x	x
DLBCTL	ORD			x	x	x			x	x	x
DLBCTL	MODE			x	x	x			x	x	x

**20.1.2 Revision Identification Register (REV)**

The revision identification register (REV) contains revision data for the peripheral. The REV is shown in [Figure 20-35](#) and described in [Table 20-10](#).

**Figure 20-35. Revision Identification Register (REV)**


LEGEND: R = Read only; -n = value after reset

**Table 20-10. Revision Identification Register (REV) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4430 0A02h	Identifies revision of peripheral.

### 20.1.3 Pin Function Register (PFUNC)

The pin function register (PFUNC) specifies the function of AXR[n], ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either a McASP pin or a general-purpose input/output (GPIO) pin. The PFUNC is shown in [Figure 20-36](#) and described in [Table 20-11](#).

**CAUTION**

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation. This includes bits that are not implemented on a particular DSP.

**Figure 20-36. Pin Function Register (PFUNC)**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved <sup>(A)</sup>
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23	Reserved <sup>(A)</sup>						16
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.



**Table 20-11. Pin Function Register (PFUNC) Field Descriptions**

Bit	Field	Value	Description
31	AFSR	0	Determines if AFSR pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
30	AHCLKR	0	Determines if AHCLKR pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
29	ACLKR	0	Determines if ACLKR pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
28	AFSX	0	Determines if AFSX pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
27	AHCLKX	0	Determines if AHCLKX pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
26	ACLKX	0	Determines if ACLKX pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
25	AMUTE	0	Determines if AMUTE pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]	0	Determines if AXR[n] pin functions as McASP or GPIO. Pin functions as McASP pin.
		1	Pin functions as GPIO pin.

### 20.1.4 Pin Direction Register (PDIR)

The pin direction register (PDIR) specifies the direction of AXR[n], ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either an input or an output pin. The PDIR is shown in [Figure 20-37](#) and described in [Table 20-12](#).

Regardless of the pin function register (PFUNC) setting, each PDIR bit must be set to 1 for the specified pin to be enabled as an output and each PDIR bit must be cleared to 0 for the specified pin to be an input.

For example, if the McASP is configured to use an internally-generated bit clock and the clock is to be driven out to the system, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output).

When AXR[n] is configured to transmit, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output). Similarly, when AXR[n] is configured to receive, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be cleared to 0 (an input).

#### CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation. This includes bits that are not implemented on a particular DSP.

**Figure 20-37. Pin Direction Register (PDIR)**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved <sup>(A)</sup>
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23							16
Reserved <sup>(A)</sup>							
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-12. Pin Direction Register (PDIR) Field Descriptions**

Bit	Field	Value	Description
31	AFSR	0	Determines if AFSR pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
30	AHCLKR	0	Determines if AHCLKR pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
29	ACLKR	0	Determines if ACLKR pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
28	AFSX	0	Determines if AFSX pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
27	AHCLKX	0	Determines if AHCLKX pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
26	ACLKX	0	Determines if ACLKX pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
25	AMUTE	0	Determines if AMUTE pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]	0	Determines if AXR[n] pin functions as an input or output. Pin functions as input.
		1	Pin functions as output.

### 20.1.5 Pin Data Output Register (PDOUT)

The pin data output register (PDOUT) holds a value for data out at all times, and may be read back at all times. The value held by PDOUT is not affected by writing to PDIR and PFUNC. However, the data value in PDOUT is driven out onto the McASP pin only if the corresponding bit in PFUNC is set to 1 (GPIO function) and the corresponding bit in PDIR is set to 1 (output). When reading data, returns the corresponding bit value in PDOUT[n], does not return input from I/O pin; when writing data, writes to the corresponding PDOUT[n] bit. The PDOUT is shown in [Figure 20-38](#) and described in [Table 20-13](#).

PDOUT has these aliases or alternate addresses:

- PDSET - when written to at this address, writing a 1 to a bit in PDSET sets the corresponding bit in PDOUT to 1; writing a 0 has no effect and keeps the bits in PDOUT unchanged.
- PDCLR - when written to at this address, writing a 1 to a bit in PDCLR clears the corresponding bit in PDOUT to 0; writing a 0 has no effect and keeps the bits in PDOUT unchanged.

There is only one set of data out bits, PDOUT[31-0]. The other registers, PDSET and PDCLR, are just different addresses for the same control bits, with different behaviors during writes.

**CAUTION**

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation. This includes bits that are not implemented on a particular DSP.

**Figure 20-38. Pin Data Output Register (PDOUT)**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved <sup>(A)</sup>
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23	Reserved <sup>(A)</sup>						16
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-13. Pin Data Output Register (PDOUT) Field Descriptions**

Bit	Field	Value	Description
31	AFSR	0 1	Determines drive on AFSR output pin when the corresponding PFUNC[31] and PDIR[31] bits are set to 1. Pin drives low. Pin drives high.
30	AHCLKR	0 1	Determines drive on AHCLKR output pin when the corresponding PFUNC[30] and PDIR[30] bits are set to 1. Pin drives low. Pin drives high.
29	ACLKR	0 1	Determines drive on ACLKR output pin when the corresponding PFUNC[29] and PDIR[29] bits are set to 1. Pin drives low. Pin drives high.
28	AFSX	0 1	Determines drive on AFSX output pin when the corresponding PFUNC[28] and PDIR[28] bits are set to 1. Pin drives low. Pin drives high.
27	AHCLKX	0 1	Determines drive on AHCLKX output pin when the corresponding PFUNC[27] and PDIR[27] bits are set to 1. Pin drives low. Pin drives high.
26	ACLKX	0 1	Determines drive on ACLKX output pin when the corresponding PFUNC[26] and PDIR[26] bits are set to 1. Pin drives low. Pin drives high.
25	AMUTE	0 1	Determines drive on AMUTE output pin when the corresponding PFUNC[25] and PDIR[25] bits are set to 1. Pin drives low. Pin drives high.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]	0 1	Determines drive on AXR[n] output pin when the corresponding PFUNC[n] and PDIR[n] bits are set to 1. Pin drives low. Pin drives high.

### 20.1.6 Pin Data Input Register (PDIN)

The pin data input register (PDIN) holds the I/O pin state of each of the McASP pins. PDIN allows the actual value of the pin to be read, regardless of the state of PFUNC and PDIR. The value after reset for registers 1 through 15 and 24 through 31 depends on how the pins are being driven. The PDIN is shown in [Figure 20-39](#) and described in [Table 20-14](#).

**CAUTION**

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation. This includes bits that are not implemented on a particular DSP.

**Figure 20-39. Pin Data Input Register (PDIN)**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved <sup>(A)</sup>
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23							16
Reserved <sup>(A)</sup>							
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-14. Pin Data Input Register (PDIN) Field Descriptions**

Bit	Field	Value	Description
31	AFSR	0	Logic level on AFSR pin. Pin is logic low.
		1	Pin is logic high.
30	AHCLKR	0	Logic level on AHCLKR pin. Pin is logic low.
		1	Pin is logic high.
29	ACLKR	0	Logic level on ACLKR pin. Pin is logic low.
		1	Pin is logic high.
28	AFSX	0	Logic level on AFSX pin. Pin is logic low.
		1	Pin is logic high.
27	AHCLKX	0	Logic level on AHCLKX pin. Pin is logic low.
		1	Pin is logic high.
26	ACLKX	0	Logic level on ACLKX pin. Pin is logic low.
		1	Pin is logic high.
25	AMUTE	0	Logic level on AMUTE pin. Pin is logic low.
		1	Pin is logic high.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]	0	Logic level on AXR[n] pin. Pin is logic low.
		1	Pin is logic high.

### 20.1.7 Pin Data Set Register (PDSET)

The pin data set register (PDSET) is an alias of the pin data output register (PDOOUT) for writes only. Writing a 1 to the PDSET bit sets the corresponding bit in PDOOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic high on the pin. PDSET is useful for a multitasking system because it allows you to set to a logic high only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDSET is shown in [Figure 20-40](#) and described in [Table 20-15](#).

#### CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation. This includes bits that are not implemented on a particular DSP.

**Figure 20-40. Pin Data Set Register (PDSET)**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved <sup>(A)</sup>
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23	Reserved <sup>(A)</sup>						16
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.



**Table 20-15. Pin Data Set Register (PDSET) Field Descriptions**

Bit	Field	Value	Description
31	AFSR	0	Allows the corresponding AFSR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[31] bit is set to 1.
30	AHCLKR	0	Allows the corresponding AHCLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[30] bit is set to 1.
29	ACLKR	0	Allows the corresponding ACLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[29] bit is set to 1.
28	AFSX	0	Allows the corresponding AFSX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[28] bit is set to 1.
27	AHCLKX	0	Allows the corresponding AHCLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[27] bit is set to 1.
26	ACLKX	0	Allows the corresponding ACLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[26] bit is set to 1.
25	AMUTE	0	Allows the corresponding AMUTE bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[25] bit is set to 1.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]	0	Allows the corresponding AXR[n] bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[n] bit is set to 1.

### 20.1.8 Pin Data Clear Register (PDCLR)

The pin data clear register (PDCLR) is an alias of the pin data output register (PDOOUT) for writes only. Writing a 1 to the PDCLR bit clears the corresponding bit in PDOOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic low on the pin. PDCLR is useful for a multitasking system because it allows you to clear to a logic low only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDCLR is shown in [Figure 20-41](#) and described in [Table 20-16](#).

#### CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation. This includes bits that are not implemented on a particular DSP.

**Figure 20-41. Pin Data Clear Register (PDCLR)**

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved <sup>(A)</sup>
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
23							16
Reserved <sup>(A)</sup>							
R-0							
15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-16. Pin Data Clear Register (PDCLR) Field Descriptions**

Bit	Field	Value	Description
31	AFSR	0	Allows the corresponding AFSR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[31] bit is cleared to 0.
30	AHCLKR	0	Allows the corresponding AHCLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[30] bit is cleared to 0.
29	ACLKR	0	Allows the corresponding ACLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[29] bit is cleared to 0.
28	AFSX	0	Allows the corresponding AFSX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[28] bit is cleared to 0.
27	AHCLKX	0	Allows the corresponding AHCLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[27] bit is cleared to 0.
26	ACLKX	0	Allows the corresponding ACLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[26] bit is cleared to 0.
25	AMUTE	0	Allows the corresponding AMUTE bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[25] bit is cleared to 0.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]	0	Allows the corresponding AXR[n] bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port. No effect.
		1	PDOUT[n] bit is cleared to 0.

### 20.1.9 Global Control Register (GBLCTL)

The global control register (GBLCTL) provides initialization of the transmit and receive sections. The GBLCTL is shown in [Figure 20-42](#) and described in [Table 20-17](#).

The bit fields in GBLCTL are synchronized and latched by the corresponding clocks (ACLKX for bits 12-8 and ACLKR for bits 4-0). Before GBLCTL is programmed, you must ensure that serial clocks are running. If the corresponding external serial clocks, ACLKX and ACLKR, are not yet running, you should select the internal serial clock source in AHCLKXCTL, AHCLKRCTL, ACLKXCTL, and ACLKRCTL before GBLCTL is programmed. Also, after programming any bits in GBLCTL you should not proceed until you have read back from GBLCTL and verified that the bits are latched in GBLCTL.

**Figure 20-42. Global Control Register (GBLCTL)**

31	Reserved <sup>(A)</sup>						16
R-0							
15	13	12	11	10	9	8	
Reserved <sup>(A)</sup>		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	5	4	3	2	1	0	
Reserved <sup>(A)</sup>		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-17. Global Control Register (GBLCTL) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST	0 1	Transmit frame sync generator reset enable bit. 0 Transmit frame sync generator is reset. 1 Transmit frame sync generator is active. When released from reset, the transmit frame sync generator begins counting serial clocks and generating frame sync as programmed.
11	XSMRST	0 1	Transmit state machine reset enable bit. 0 Transmit state machine is held in reset. AXR[n] pin state: If PFUNC[n] = 0 and PDIR[n] = 1; then the serializer drives the AXR[n] pin to the state specified for inactive time slot (as determined by DISMOD bits in SRCTL). 1 Transmit state machine is released from reset. When released from reset, the transmit state machine immediately transfers data from XRBUF[n] to XRSR[n]. The transmit state machine sets the underrun flag (XUNDRN) in XSTAT, if XRBUF[n] have not been preloaded with data before reset is released. The transmit state machine also immediately begins detecting frame sync and is ready to transmit. Transmit TDM time slot begins at slot 0 after reset is released.
10	XSRCLR	0 1	Transmit serializer clear enable bit. By clearing then setting this bit, the transmit buffer is flushed to an empty state (XDATA = 1). If XSMRST = 1, XSRCLR = 1, XDATA = 1, and XBUF is not loaded with new data before the start of the next active time slot, an underrun will occur. 0 Transmit serializers are cleared. 1 Transmit serializers are active. When the transmit serializers are first taken out of reset (XSRCLR changes from 0 to 1), the transmit data ready bit (XDATA) in XSTAT is set to indicate XBUF is ready to be written.
9	XHCLKRST	0 1	Transmit high-frequency clock divider reset enable bit. 0 Transmit high-frequency clock divider is held in reset. 1 Transmit high-frequency clock divider is running.

**Table 20-17. Global Control Register (GBLCTL) Field Descriptions (continued)**

Bit	Field	Value	Description
8	XCLKRST	0	Transmit clock divider reset enable bit. Transmit clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Transmit clock divider is running.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST	0	Receive frame sync generator reset enable bit. Receive frame sync generator is reset.
		1	Receive frame sync generator is active. When released from reset, the receive frame sync generator begins counting serial clocks and generating frame sync as programmed.
3	RSMRST	0	Receive state machine reset enable bit. Receive state machine is held in reset.
		1	Receive state machine is released from reset. When released from reset, the receive state machine immediately begins detecting frame sync and is ready to receive. Receive TDM time slot begins at slot 0 after reset is released.
2	RSRCLR	0	Receive serializer clear enable bit. By clearing then setting this bit, the receive buffer is flushed. Receive serializers are cleared.
		1	Receive serializers are active.
1	RHCLKRST	0	Receive high-frequency clock divider reset enable bit. Receive high-frequency clock divider is held in reset.
		1	Receive high-frequency clock divider is running.
0	RCLKRST	0	Receive clock divider reset enable bit. Receive clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Receive clock divider is running.

### 20.1.10 Audio Mute Control Register (AMUTE)

The audio mute control register (AMUTE) controls the McASP audio mute (AMUTE) output pin. The value after reset for register 4 depends on how the pins are being driven. The AMUTE is shown in [Figure 20-43](#) and described in [Table 20-18](#).

**Figure 20-43. Audio Mute Control Register (AMUTE)**

31	Reserved <sup>(A)</sup>						16
R-0							
15	13	12	11	10	9	8	
Reserved <sup>(A)</sup>		XDMAERR	RDMAERR	XCKFAIL	RCKFAIL	XSYNCERR	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	
RSYNCERR	XUNDRN	ROVRN	INSTAT	INEN	INPOL	MUTEN	
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-18. Audio Mute Control Register (AMUTE) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XDMAERR	0 1	If transmit DMA error (XDMAERR), drive AMUTE active enable bit. 0 Drive is disabled. Detection of transmit DMA error is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of transmit DMA error, AMUTE is active and is driven according to MUTEN bit.
11	RDMAERR	0 1	If receive DMA error (RDMAERR), drive AMUTE active enable bit. 0 Drive is disabled. Detection of receive DMA error is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of receive DMA error, AMUTE is active and is driven according to MUTEN bit.
10	XCKFAIL	0 1	If transmit clock failure (XCKFAIL), drive AMUTE active enable bit. 0 Drive is disabled. Detection of transmit clock failure is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of transmit clock failure, AMUTE is active and is driven according to MUTEN bit.
9	RCKFAIL	0 1	If receive clock failure (RCKFAIL), drive AMUTE active enable bit. 0 Drive is disabled. Detection of receive clock failure is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of receive clock failure, AMUTE is active and is driven according to MUTEN bit.
8	XSYNCERR	0 1	If unexpected transmit frame sync error (XSYNCERR), drive AMUTE active enable bit. 0 Drive is disabled. Detection of unexpected transmit frame sync error is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of unexpected transmit frame sync error, AMUTE is active and is driven according to MUTEN bit.
7	RSYNCERR	0 1	If unexpected receive frame sync error (RSYNCERR), drive AMUTE active enable bit. 0 Drive is disabled. Detection of unexpected receive frame sync error is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of unexpected receive frame sync error, AMUTE is active and is driven according to MUTEN bit.
6	XUNDRN	0 1	If transmit underrun error (XUNDRN), drive AMUTE active enable bit. 0 Drive is disabled. Detection of transmit underrun error is ignored by AMUTE. 1 Drive is enabled (active). Upon detection of transmit underrun error, AMUTE is active and is driven according to MUTEN bit.

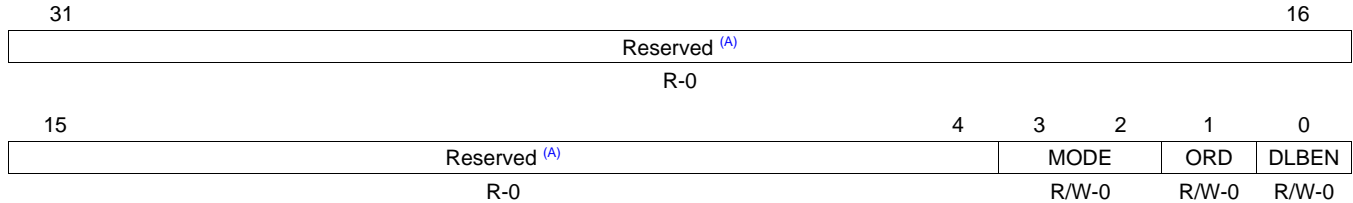
**Table 20-18. Audio Mute Control Register (AMUTE) Field Descriptions (continued)**

Bit	Field	Value	Description
5	ROVRN		If receiver overrun error (ROVRN), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receiver overrun error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receiver overrun error, AMUTE is active and is driven according to MUTEN bit.
4	INSTAT		Determines drive on AXR[n] pin when PFUNC[n] and PDIR[n] bits are set to 1.
		0	AMUTEIN pin is inactive.
		1	AMUTEIN pin is active. Audio mute in error is detected.
3	INEN		Drive AMUTE active when AMUTEIN error is active (INSTAT = 1).
		0	Drive is disabled. AMUTEIN is ignored by AMUTE.
		1	Drive is enabled (active). INSTAT = 1 drives AMUTE active.
2	INPOL		Audio mute in (AMUTEIN) polarity select bit.
		0	Polarity is active high. A high on AMUTEIN sets INSTAT to 1.
		1	Polarity is active low. A low on AMUTEIN sets INSTAT to 1.
1-0	MUTEN	0-3h	AMUTE pin enable bit (unless overridden by GPIO registers).
		0	AMUTE pin is disabled, pin goes to tri-state condition.
		1h	AMUTE pin is driven high if error is detected.
		2h	AMUTE pin is driven low if error is detected.
		3h	Reserved

### 20.1.11 Digital Loopback Control Register (DLBCTL)

The digital loopback control register (DLBCTL) controls the internal loopback settings of the McASP in TDM mode. The DLBCTL is shown in [Figure 20-44](#) and described in [Table 20-19](#).

**Figure 20-44. Digital Loopback Control Register (DLBCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-19. Digital Loopback Control Register (DLBCTL) Field Descriptions**

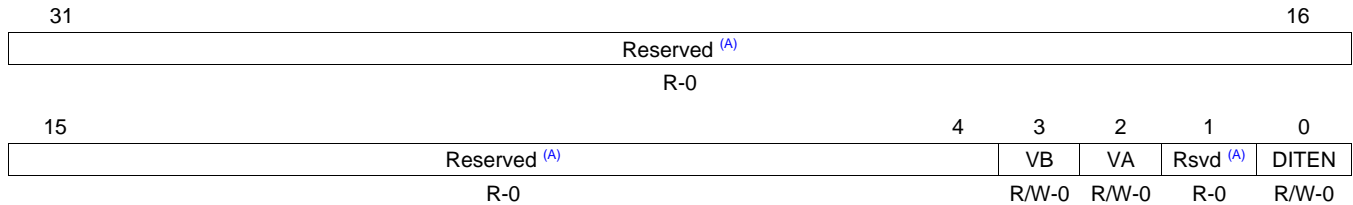
Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-2	MODE	0-3h	Loopback generator mode bits. Applies only when loopback mode is enabled (DLBEN = 1).
		0	Default and reserved on loopback mode (DLBEN = 1). When in non-loopback mode (DLBEN = 0), MODE should be left at default (00). When in loopback mode (DLBEN = 1), MODE = 00 is reserved and not applicable.
		1h	Transmit clock and frame sync generators used by both transmit and receive sections. When in loopback mode (DLBEN = 1), MODE must be 01.
		2h-3h	Reserved.
1	ORD	0	Loopback order bit when loopback mode is enabled (DLBEN = 1). Odd serializers N + 1 transmit to even serializers N that receive. The corresponding serializers must be programmed properly.
		1	Even serializers N transmit to odd serializers N+1 that receive. The corresponding serializers must be programmed properly.
0	DLBEN	0	Loopback mode enable bit. Loopback mode is disabled.
		1	Loopback mode is enabled.



### 20.1.12 Digital Mode Control Register (DITCTL)

The DIT mode control register (DITCTL) controls DIT operations of the McASP. The DITCTL is shown in Figure 20-45 and described in Table 20-20.

**Figure 20-45. Digital Mode Control Register (DITCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-20. Digital Mode Control Register (DITCTL) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3	VB	0 1	Valid bit for odd time slots (DIT right subframe). V bit is 0 during odd DIT subframes. V bit is 1 during odd DIT subframes.
2	VA	0 1	Valid bit for even time slots (DIT left subframe). V bit is 0 during even DIT subframes. V bit is 1 during even DIT subframes.
1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	DITEN	0 1	DIT mode enable bit. DITEN should only be changed while the XSMRST bit in GBLCTL is in reset (and for startup, XSRCLR also in reset). However, it is not necessary to reset the XCLKRST or XHCLKRST bits in GBLCTL to change DITEN. 0 DIT mode is disabled. Transmitter operates in TDM or burst mode. 1 DIT mode is enabled. Transmitter operates in DIT encoded mode.

### 20.1.13 Receiver Global Control Register (RGBLCTL)

Alias of the global control register (GBLCTL). Writing to the receiver global control register (RRGBLCTL) affects only the receive bits of GBLCTL (bits 4-0). Reads from RRGBLCTL return the value of GBLCTL. RRGBLCTL allows the receiver to be reset independently from the transmitter. The RRGBLCTL is shown in Figure 20-46 and described in Table 20-21. See Section 20.1.9 for a detailed description of GBLCTL.

**Figure 20-46. Receiver Global Control Register (RRGBLCTL)**

31	Reserved <sup>(A)</sup>						16
R-0							
15	13	12	11	10	9	8	
Reserved <sup>(A)</sup>		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST	
R-0		R-0	R-0	R-0	R-0	R-0	
7	5	4	3	2	1	0	
Reserved <sup>(A)</sup>		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-21. Receiver Global Control Register (RRGBLCTL) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST	x	Transmit frame sync generator reset enable bit. A read of this bit returns the XFRST bit value of GBLCTL. Writes have no effect.
11	XSMRST	x	Transmit state machine reset enable bit. A read of this bit returns the XSMRST bit value of GBLCTL. Writes have no effect.
10	XSRCLR	x	Transmit serializer clear enable bit. A read of this bit returns the XSRCLR bit value of GBLCTL. Writes have no effect.
9	XHCLKRST	x	Transmit high-frequency clock divider reset enable bit. A read of this bit returns the XHCLKRST bit value of GBLCTL. Writes have no effect.
8	XCLKRST	x	Transmit clock divider reset enable bit. a read of this bit returns the XCLKRST bit value of GBLCTL. Writes have no effect.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST	0 1	Receive frame sync generator reset enable bit. A write to this bit affects the RFRST bit of GBLCTL. 0 Receive frame sync generator is reset. 1 Receive frame sync generator is active.
3	RSMRST	0 1	Receive state machine reset enable bit. A write to this bit affects the RSMRST bit of GBLCTL. 0 Receive state machine is held in reset. 1 Receive state machine is released from reset.
2	RSRCLR	0 1	Receive serializer clear enable bit. A write to this bit affects the RSRCLR bit of GBLCTL. 0 Receive serializers are cleared. 1 Receive serializers are active.
1	RHCLKRST	0 1	Receive high-frequency clock divider reset enable bit. A write to this bit affects the RHCLKRST bit of GBLCTL. 0 Receive high-frequency clock divider is held in reset. 1 Receive high-frequency clock divider is running.
0	RCLKRST	0 1	Receive clock divider reset enable bit. A write to this bit affects the RCLKRST bit of GBLCTL. 0 Receive clock divider is held in reset. 1 Receive clock divider is running.

### 20.1.14 Receive Format Unit Bit Mask Register (RMASK)

The receive format unit bit mask register (RMASK) determines which bits of the received data are masked off and padded with a known value before being read by the CPU or DMA. The RMASK is shown in Figure 20-47 and described in Table 20-22.

**Figure 20-47. Receive Format Unit Bit Mask Register (RMASK)**

31	30	29	28	27	26	25	24
RMASK31	RMASK30	RMASK29	RMASK28	RMASK27	RMASK26	RMASK25	RMASK24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RMASK23	RMASK22	RMASK21	RMASK20	RMASK19	RMASK18	RMASK17	RMASK16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
RMASK15	RMASK14	RMASK13	RMASK12	RMASK11	RMASK10	RMASK9	RMASK8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RMASK7	RMASK6	RMASK5	RMASK4	RMASK3	RMASK2	RMASK1	RMASK0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 20-22. Receive Format Unit Bit Mask Register (RMASK) Field Descriptions**

Bit	Field	Value	Description
31-0	RMASK[31-0]	0	Receive data mask enable bit.
		1	Corresponding bit of receive data (after passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (RPAD and RPBIT bits in RFMT).
		1	Corresponding bit of receive data (after passing through reverse and rotate units) is returned to CPU or DMA.

### 20.1.15 Receive Bit Stream Format Register (RFMT)

The receive bit stream format register (RFMT) configures the receive data format. The RFMT is shown in Figure 20-48 and described in Table 20-23.

**Figure 20-48. Receive Bit Stream Format Register (RFMT)**

31											18		17	16						
Reserved <sup>(A)</sup>											RDATDLY									
R-0											R/W-0									
15			14		13		12		8		7		4		3		2		0	
RRVRS			RPAD		RPBIT		RSSZ		RBUSEL		RROT									
R/W-0			R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-23. Receive Bit Stream Format Register (RFMT) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
17-16	RDATDLY	0-3h	Receive bit delay.
		0	0-bit delay. The first receive data bit, AXR[n], occurs in same ACLKR cycle as the receive frame sync (AFSR).
		1h	1-bit delay. The first receive data bit, AXR[n], occurs one ACLKR cycle after the receive frame sync (AFSR).
		2h	2-bit delay. The first receive data bit, AXR[n], occurs two ACLKR cycles after the receive frame sync (AFSR).
		3h	Reserved.
15	RRVRS		Receive serial bitstream order.
		0	Bitstream is LSB first. No bit reversal is performed in receive format bit reverse unit.
		1	Bitstream is MSB first. Bit reversal is performed in receive format bit reverse unit.
14-13	RPAD	0-3h	Pad value for extra bits in slot not belonging to the word. This field only applies to bits when RMASK[n] = 0.
		0	Pad extra bits with 0.
		1h	Pad extra bits with 1.
		2h	Pad extra bits with one of the bits from the word as specified by RPBIT bits.
		3h	Reserved.
12-8	RPBIT	0-1Fh	RPBIT value determines which bit (as read by the CPU or DMA from RBUF[n]) is used to pad the extra bits. This field only applies when RPAD = 2h.
		0	Pad with bit 0 value.
		1h-1Fh	Pad with bit 1 to bit 31 value.

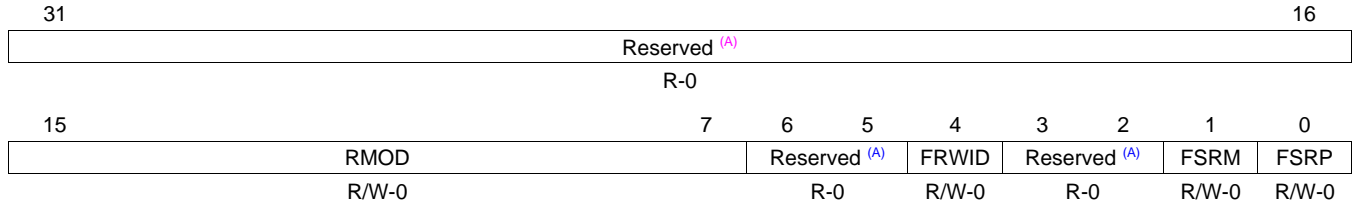
**Table 20-23. Receive Bit Stream Format Register (RFMT) Field Descriptions (continued)**

Bit	Field	Value	Description
7-4	RSSZ	0-Fh 0-2h 3h 4h 5h 6h 7h 8h 9h Ah Bh Ch Dh Eh Fh	Receive slot size. Reserved Slot size is 8 bits. Reserved Slot size is 12 bits. Reserved Slot size is 16 bits. Reserved Slot size is 20 bits. Reserved Slot size is 24 bits Reserved Slot size is 28 bits. Reserved Slot size is 32 bits.
3	RBUSEL	0 1	Selects whether reads from serializer buffer XRBUF[n] by way of RBUF $n$ by the CPU/EDMA occur through the peripheral configuration port or the DMA port. 0 Reads from XRBUF[n] originate on the DMA port. Reads from XRBUF[n] on the peripheral configuration port are ignored. 1 Reads from XRBUF[n] originate on the peripheral configuration port. Reads from XRBUF[n] on the DMA port are ignored.
2-0	RROT	0-7h 0 1h 2h 3h 4h 5h 6h 7h	Right-rotation value for receive rotate right format unit. 0 Rotate right by 0 (no rotation). 1h Rotate right by 4 bit positions. 2h Rotate right by 8 bit positions. 3h Rotate right by 12 bit positions. 4h Rotate right by 16 bit positions. 5h Rotate right by 20 bit positions. 6h Rotate right by 24 bit positions. 7h Rotate right by 28 bit positions.

### 20.1.16 Receive Frame Sync Control Register (AFSRCTL)

The receive frame sync control register (AFSRCTL) configures the receive frame sync (AFSR). The AFSRCTL is shown in [Figure 20-49](#) and described in [Table 20-24](#).

**Figure 20-49. Receive Frame Sync Control Register (AFSRCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-24. Receive Frame Sync Control Register (AFSRCTL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-7	RMOD	0-1FFh	Receive frame sync mode select bits.
		0	Burst mode
		1h	Reserved
		2h-20h	2-slot TDM (I2S mode) to 32-slot TDM
		21h-17Fh	Reserved
		180h	384-slot TDM (external DIR IC inputting 384-slot DIR frames to McASP over I2S interface)
		181h-1FFh	Reserved
6-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	FRWID		Receive frame sync width select bit indicates the width of the receive frame sync (AFSR) during its active period.
		0	Single bit
		1	Single word
3-2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	FSRM		Receive frame sync generation select bit.
		0	Externally-generated receive frame sync
		1	Internally-generated receive frame sync
0	FSRP		Receive frame sync polarity select bit.
		0	A rising edge on receive frame sync (AFSR) indicates the beginning of a frame.
		1	A falling edge on receive frame sync (AFSR) indicates the beginning of a frame.

### 20.1.17 Receive Clock Control Register (ACLKRCTL)

The receive clock control register (ACLKRCTL) configures the receive bit clock (ACLKR) and the receive clock generator. The ACLKRCTL is shown in [Figure 20-50](#) and described in [Table 20-25](#).

**Figure 20-50. Receive Clock Control Register (ACLKRCTL)**

31	Reserved <sup>(A)</sup>					16
R-0						
15	8	7	6	5	4	0
Reserved <sup>(A)</sup>		CLKRP	Rsvd <sup>(A)</sup>	CLKRM	CLKRDIV	
R-0		R/W-0	R-0	R/W-1	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

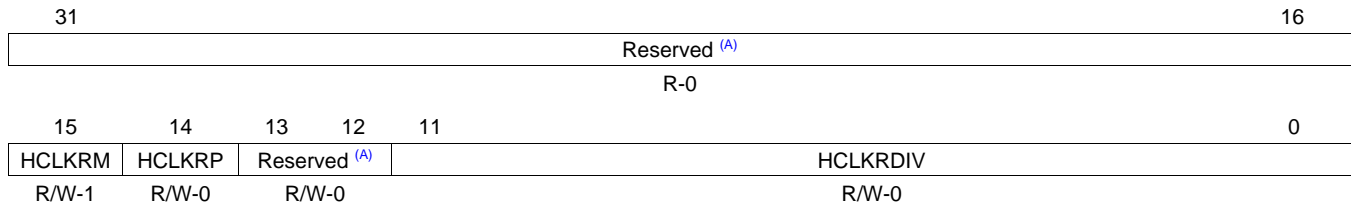
**Table 20-25. Receive Clock Control Register (ACLKRCTL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	CLKRP	0 1	Receive bitstream clock polarity select bit. Note that this bitfield does not have any effect, if ACLKXCTL.ASYNC = 0 (see <a href="#">Section 20.1.29</a> for a description for the ASYNC bit). 0 Falling edge. Receiver samples data on the falling edge of the serial clock, so the external transmitter driving this receiver must shift data out on the rising edge of the serial clock. 1 Rising edge. Receiver samples data on the rising edge of the serial clock, so the external transmitter driving this receiver must shift data out on the falling edge of the serial clock.
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	CLKRM	0 1	Receive bit clock source bit. Note that this bitfield does not have any effect, if ACLKXCTL.ASYNC = 0 (see <a href="#">Section 20.1.29</a> for a description for the ASYNC bit). 0 External receive clock source from ACLKR pin. 1 Internal receive clock source from output of programmable bit clock divider.
4-0	CLKRDIV	0-1Fh 0 1h 2h-1Fh	Receive bit clock divide ratio bits determine the divide-down ratio from AHCLKR to ACLKR. Note that this bitfield does not have any effect, if ACLKXCTL.ASYNC = 0 (see <a href="#">Section 20.1.29</a> for a description for the ASYNC bit). 0 Divide-by-1 1h Divide-by-2 2h-1Fh Divide-by-3 to divide-by-32

### 20.1.18 Receive High-Frequency Clock Control Register (AHCLKRCTL)

The receive high-frequency clock control register (AHCLKRCTL) configures the receive high-frequency master clock (AHCLKR) and the receive clock generator. The AHCLKRCTL is shown in [Figure 20-51](#) and described in [Table 20-26](#).

**Figure 20-51. Receive High-Frequency Clock Control Register (AHCLKRCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-26. Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15	HCLKRM	0	Receive high-frequency clock source bit. External receive high-frequency clock source from AHCLKR pin.
		1	Internal receive high-frequency clock source from output of programmable high clock divider.
14	HCLKRP	0	Receive bitstream high-frequency clock polarity select bit. Not inverted. AHCLKR is not inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.
		1	Inverted. AHCLKR is inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.
13-12	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
11-0	HCLKRDIV	0-FFFh	Receive high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKR.
		0	Divide-by-1
		1h	Divide-by-2
		2h-FFFh	Divide-by-3 to divide-by-4096



### 20.1.19 Receive TDM Time Slot Register (RTDM)

The receive TDM time slot register (RTDM) specifies which TDM time slot the receiver is active. The RTDM is shown in [Figure 20-52](#) and described in [Table 20-27](#).

**Figure 20-52. Receive TDM Time Slot Register (RTDM)**

31	30	29	28	27	26	25	24
RTDMS31	RTDMS30	RTDMS29	RTDMS28	RTDMS27	RTDMS26	RTDMS25	RTDMS24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
RTDMS23	RTDMS22	RTDMS21	RTDMS20	RTDMS19	RTDMS18	RTDMS17	RTDMS16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
RTDMS15	RTDMS14	RTDMS13	RTDMS12	RTDMS11	RTDMS10	RTDMS9	RTDMS8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RTDMS7	RTDMS6	RTDMS5	RTDMS4	RTDMS3	RTDMS2	RTDMS1	RTDMS0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 20-27. Receive TDM Time Slot Register (RTDM) Field Descriptions**

Bit	Field	Value	Description
31-0	RTDMS[31-0]	0	Receiver mode during TDM time slot <i>n</i> . Receive TDM time slot <i>n</i> is inactive. The receive serializer does not shift in data during this slot.
		1	Receive TDM time slot <i>n</i> is active. The receive serializer shifts in data during this slot.

## 20.1.20 Receiver Interrupt Control Register (RINTCTL)

The receiver interrupt control register (RINTCTL) controls generation of the McASP receive interrupt (RINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates RINT. The RINTCTL is shown in Figure 20-53 and described in Table 20-28. See Section 20.1.21 for a description of the interrupt conditions.

**Figure 20-53. Receiver Interrupt Control Register (RINTCTL)**

Reserved <sup>(A)</sup>							
R-0							
7	6	5	4	3	2	1	0
RSTAFRM	Reserved <sup>(A)</sup>	RDATA	RLAST	RDMAERR	RCKFAIL	RSYNCERR	ROVRN
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-28. Receiver Interrupt Control Register (RINTCTL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	RSTAFRM	0	Receive start of frame interrupt enable bit. Interrupt is disabled. A receive start of frame interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive start of frame interrupt generates a McASP receive interrupt (RINT).
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	RDATA	0	Receive data ready interrupt enable bit. Interrupt is disabled. A receive data ready interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive data ready interrupt generates a McASP receive interrupt (RINT).
4	RLAST	0	Receive last slot interrupt enable bit. Interrupt is disabled. A receive last slot interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive last slot interrupt generates a McASP receive interrupt (RINT).
3	RDMAERR	0	Receive DMA error interrupt enable bit. Interrupt is disabled. A receive DMA error interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive DMA error interrupt generates a McASP receive interrupt (RINT).
2	RCKFAIL	0	Receive clock failure interrupt enable bit. Interrupt is disabled. A receive clock failure interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive clock failure interrupt generates a McASP receive interrupt (RINT).
1	RSYNCERR	0	Unexpected receive frame sync interrupt enable bit. Interrupt is disabled. An unexpected receive frame sync interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. An unexpected receive frame sync interrupt generates a McASP receive interrupt (RINT).
0	ROVRN	0	Receiver overrun interrupt enable bit. Interrupt is disabled. A receiver overrun interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receiver overrun interrupt generates a McASP receive interrupt (RINT).

### 20.1.21 Receiver Status Register (RSTAT)

The receiver status register (RSTAT) provides the receiver status and receive TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The RSTAT is shown in [Figure 20-54](#) and described in [Table 20-29](#).

**Figure 20-54. Receiver Status Register (RSTAT)**

31							9	8
Reserved <sup>(A)</sup>							RERR	
R-0							R/W-0	
7	6	5	4	3	2	1	0	
RDMAERR	RSTAFRM	RDATA	RLAST	RTDMSLOT	RCKFAIL	RSYNCERR	ROVRN	
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-29. Receiver Status Register (RSTAT) Field Descriptions**

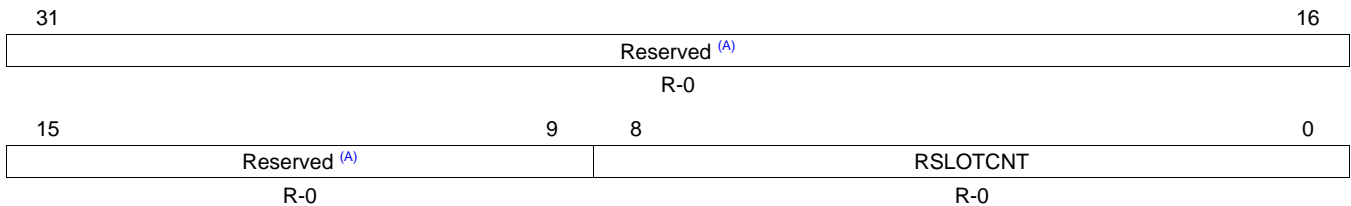
Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	RERR	0 1	RERR bit always returns a logic-OR of: ROVRN   RSYNCERR   RCKFAIL   RDMAERR Allows a single bit to be checked to determine if a receiver error interrupt has occurred. 0 No errors have occurred. 1 An error has occurred.
7	RDMAERR	0 1	Receive DMA error flag. RDMAERR is set when the CPU or DMA reads more serializers through the DMA port in a given time slot than were programmed as receivers. Causes a receive interrupt (RINT), if this bit is set and RDMAERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. 0 Receive DMA error did not occur. 1 Receive DMA error did occur.
6	RSTAFRM	0 1	Receive start of frame flag. Causes a receive interrupt (RINT), if this bit is set and RSTAFRM in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. 0 No new receive frame sync (AFSR) is detected. 1 A new receive frame sync (AFSR) is detected.
5	RDATA	0 1	Receive data ready flag. Causes a receive interrupt (RINT), if this bit is set and RDATA in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. 0 No new data in RBUF. 1 Data is transferred from XRSR to RBUF and ready to be serviced by the CPU or DMA. When RDATA is set, it always causes a DMA event (AREVT).
4	RLAST	0 1	Receive last slot flag. RLAST is set along with RDATA, if the current slot is the last slot in a frame. Causes a receive interrupt (RINT), if this bit is set and RLAST in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. 0 Current slot is not the last slot in a frame. 1 Current slot is the last slot in a frame. RDATA is also set.
3	RTDMSLOT	0 1	Returns the LSB of RSLLOT. Allows a single read of RSTAT to determine whether the current TDM time slot is even or odd. 0 Current TDM time slot is odd. 1 Current TDM time slot is even.
2	RCKFAIL	0 1	Receive clock failure flag. RCKFAIL is set when the receive clock failure detection circuit reports an error (see <a href="#">Section 20.0.27.6.6</a> ). Causes a receive interrupt (RINT), if this bit is set and RCKFAIL in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. 0 Receive clock failure did not occur. 1 Receive clock failure did occur.

**Table 20-29. Receiver Status Register (RSTAT) Field Descriptions (continued)**

Bit	Field	Value	Description
1	RSYNCERR	0 1	Unexpected receive frame sync flag. RSYNCERR is set when a new receive frame sync (AFSR) occurs before it is expected. Causes a receive interrupt (RINT), if this bit is set and RSYNCERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. Unexpected receive frame sync did not occur. Unexpected receive frame sync did occur.
0	ROVRN	0 1	Receiver overrun flag. ROVRN is set when the receive serializer is instructed to transfer data from XRSR to RBUF, but the former data in RBUF has not yet been read by the CPU or DMA. Causes a receive interrupt (RINT), if this bit is set and ROVRN in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. Receiver overrun did not occur. Receiver overrun did occur.

### 20.1.22 Current Receive TDM Time Slot Registers (RSLOT)

The current receive TDM time slot register (RSLOT) indicates the current time slot for the receive data frame. The RSLOT is shown in [Figure 20-55](#) and described in [Table 20-30](#).

**Figure 20-55. Current Receive TDM Time Slot Registers (RSLOT)**

LEGEND: R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-30. Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8-0	RSLOTCNT	0-17Fh	Current receive time slot count. Legal values: 0 to 383 (17Fh). TDM function is not supported for > 32 time slots. However, TDM time slot counter may count to 383 when used to receive a DIR block (transferred over TDM format).

### 20.1.23 Receive Clock Check Control Register (RCLKCHK)

The receive clock check control register (RCLKCHK) configures the receive clock failure detection circuit. The RCLKCHK is shown in [Figure 20-56](#) and described in [Table 20-31](#).

**Figure 20-56. Receive Clock Check Control Register (RCLKCHK)**

31	24	23	16
RCNT		RMAX	
R-0		R/W-0	
15	8	7	0
RMIN		Reserved <sup>(A)</sup>	RPS
R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-31. Receive Clock Check Control Register (RCLKCHK) Field Descriptions**

Bit	Field	Value	Description
31-24	RCNT	0-FFh	Receive clock count value (from previous measurement). The clock circuit continually counts the number of DSP system clocks for every 32 receive high-frequency master clock (AHCLKR) signals, and stores the count in RCNT until the next measurement is taken.
23-16	RMAX	0-FFh	Receive clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If the current counter value is greater than RMAX after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.
15-8	RMIN	0-FFh	Receive clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If RCNT is less than RMIN after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.
7-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-0	RPS	0-Fh	Receive clock check prescaler value.
		0	McASP system clock divided by 1
		1h	McASP system clock divided by 2
		2h	McASP system clock divided by 4
		3h	McASP system clock divided by 8
		4h	McASP system clock divided by 16
		5h	McASP system clock divided by 32
		6h	McASP system clock divided by 64
		7h	McASP system clock divided by 128
		8h	McASP system clock divided by 256
		9h-Fh	Reserved

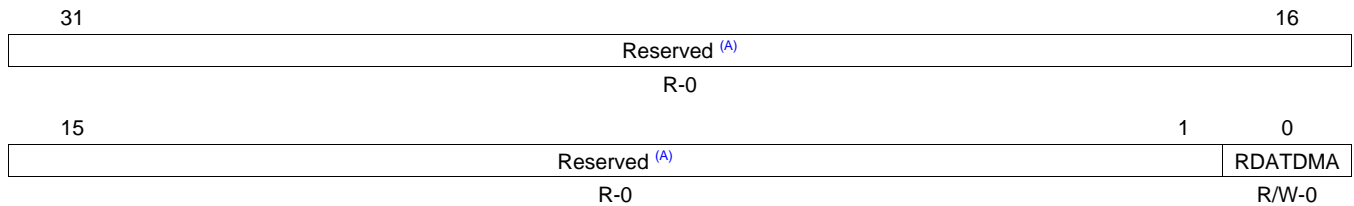
**20.1.24 Receiver DMA Event Control Register (REVTCTL)**

The receiver DMA event control register (REVTCTL) is shown in [Figure 20-57](#) and described in [Table 20-32](#).

**CAUTION**

DSP specific registers  
 Accessing REVTCTL not implemented on a specific DSP may cause improper device operation.

**Figure 20-57. Receiver DMA Event Control Register (REVTCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-32. Receiver DMA Event Control Register (REVTCTL) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	RDATDMA	0	Receive data DMA request enable bit. If writing to this field, always write the default value of 0. Receive data DMA request is enabled.
		1	Reserved.

### 20.1.25 Transmitter Global Control Register (XGBLCTL)

Alias of the global control register (GBLCTL). Writing to the transmitter global control register (XGBLCTL) affects only the transmit bits of GBLCTL (bits 12-8). Reads from XGBLCTL return the value of GBLCTL. XGBLCTL allows the transmitter to be reset independently from the receiver. The XGBLCTL is shown in Figure 20-58 and described in Table 20-33. See Section 20.1.9 for a detailed description of GBLCTL.

**Figure 20-58. Transmitter Global Control Register (XGBLCTL)**

31	Reserved <sup>(A)</sup>						16
R-0							
15	13	12	11	10	9	8	
Reserved <sup>(A)</sup>		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	5	4	3	2	1	0	
Reserved <sup>(A)</sup>		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST	
R-0		R-0	R-0	R-0	R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-33. Transmitter Global Control Register (XGBLCTL) Field Descriptions**

Bit	Field	Value	Description
31-13	Reserved	0-FFh	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST	0 1	Transmit frame sync generator reset enable bit. A write to this bit affects the XFRST bit of GBLCTL. 0 Transmit frame sync generator is reset. 1 Transmit frame sync generator is active.
11	XSMRST	0 1	Transmit state machine reset enable bit. A write to this bit affects the XSMRST bit of GBLCTL. 0 Transmit state machine is held in reset. 1 Transmit state machine is released from reset.
10	XSRCLR	0 1	Transmit serializer clear enable bit. A write to this bit affects the XSRCLR bit of GBLCTL. 0 Transmit serializers are cleared. 1 Transmit serializers are active.
9	XHCLKRST	0 1	Transmit high-frequency clock divider reset enable bit. A write to this bit affects the XHCLKRST bit of GBLCTL. 0 Transmit high-frequency clock divider is held in reset. 1 Transmit high-frequency clock divider is running.
8	XCLKRST	0 1	Transmit clock divider reset enable bit. A write to this bit affects the XCLKRST bit of GBLCTL. 0 Transmit clock divider is held in reset. 1 Transmit clock divider is running.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST	x	Receive frame sync generator reset enable bit. A read of this bit returns the RFRST bit value of GBLCTL. Writes have no effect.
3	RSMRST	x	Receive state machine reset enable bit. A read of this bit returns the RSMRST bit value of GBLCTL. Writes have no effect.
2	RSRCLR	x	Receive serializer clear enable bit. A read of this bit returns the RSRCLR bit value of GBLCTL. Writes have no effect.
1	RHCLKRST	x	Receive high-frequency clock divider reset enable bit. A read of this bit returns the RHCLKRST bit value of GBLCTL. Writes have no effect.
0	RCLKRST	x	Receive clock divider reset enable bit. A read of this bit returns the RCLKRST bit value of GBLCTL. Writes have no effect.

### 20.1.26 Transmit Format Unit Bit Mask Register (XMASK)

The transmit format unit bit mask register (XMASK) determines which bits of the transmitted data are masked off and padded with a known value before being shifted out the McASP. The XMASK is shown in Figure 20-59 and described in Table 20-34.

**Figure 20-59. Transmit Format Unit Bit Mask Register (XMASK)**

31	30	29	28	27	26	25	24
XMASK31	XMASK30	XMASK29	XMASK28	XMASK27	XMASK26	XMASK25	XMASK24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
XMASK23	XMASK22	XMASK21	XMASK20	XMASK19	XMASK18	XMASK17	XMASK16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
XMASK15	XMASK14	XMASK13	XMASK12	XMASK11	XMASK10	XMASK9	XMASK8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XMASK7	XMASK6	XMASK5	XMASK4	XMASK3	XMASK2	XMASK1	XMASK0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 20-34. Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions**

Bit	Field	Value	Description
31-0	XMASK[31-0]	0	Transmit data mask enable bit. Corresponding bit of transmit data (before passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (XPAD and XPBIT bits in XFMT), which is transmitted out the McASP in place of the original bit.
		1	Corresponding bit of transmit data (before passing through reverse and rotate units) is transmitted out the McASP.



### 20.1.27 Transmit Bit Stream Format Register (XFMT)

The transmit bit stream format register (XFMT) configures the transmit data format. The XFMT is shown in [Figure 20-60](#) and described in [Table 20-35](#).

**Figure 20-60. Transmit Bit Stream Format Register (XFMT)**

31										18		17	16						
Reserved <sup>(A)</sup>												XDATDLY							
R-0										R/W-0									
15		14		13		12		8		7		4		3		2		0	
XRVRS		XPAD		XPBIT				XSSZ				XBUSEL		XROT					
R/W-0		R/W-0		R/W-0				R/W-0				R/W-0		R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
17-16	XDATDLY	0-3h	Transmit sync bit delay.
		0	0-bit delay. The first transmit data bit, AXR[n], occurs in same ACLKX cycle as the transmit frame sync (AFSX).
		1h	1-bit delay. The first transmit data bit, AXR[n], occurs one ACLKX cycle after the transmit frame sync (AFSX).
		2h	2-bit delay. The first transmit data bit, AXR[n], occurs two ACLKX cycles after the transmit frame sync (AFSX).
		3h	Reserved.
15	XRVRS		Transmit serial bitstream order.
		0	Bitstream is LSB first. No bit reversal is performed in transmit format bit reverse unit.
		1	Bitstream is MSB first. Bit reversal is performed in transmit format bit reverse unit.
14-13	XPAD	0-3h	Pad value for extra bits in slot not belonging to word defined by XMASK. This field only applies to bits when XMASK[n] = 0.
		0	Pad extra bits with 0.
		1h	Pad extra bits with 1.
		2h	Pad extra bits with one of the bits from the word as specified by XPBIT bits.
		3h	Reserved
12-8	XPBIT	0-1Fh	XPBIT value determines which bit (as written by the CPU or DMA to XBUF[n]) is used to pad the extra bits before shifting. This field only applies when XPAD = 2h.
		0	Pad with bit 0 value.
		1-1Fh	Pad with bit 1 to bit 31 value.

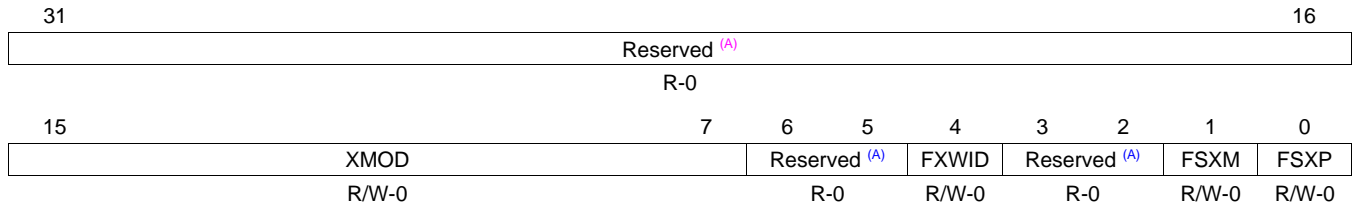
**Table 20-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions (continued)**

Bit	Field	Value	Description
7-4	XSSZ	0-Fh 0-2h 3h 4h 5h 6h 7h 8h 9h Ah Bh Ch Dh Eh Fh	Transmit slot size. Reserved Slot size is 8 bits. Reserved Slot size is 12 bits. Reserved. Slot size is 16 bits. Reserved. Slot size is 20 bits. Reserved. Slot size is 24 bits. Reserved. Slot size is 28 bits. Reserved. Slot size is 32 bits.
3	XBUSEL	0 1	Selects whether writes to serializer buffer XRBUF[n] by way of XBUF <sub>n</sub> by the CPU/EDMA occur through the peripheral configuration port or the DMA port. 0 Writes to XRBUF[n] originate from the DMA port. Writes to XRBUF[n] from the peripheral configuration port are ignored with no effect to the McASP. 1 Writes to XRBUF[n] originate from the peripheral configuration port. Writes to XRBUF[n] from the DMA port are ignored with no effect to the McASP.
2-0	XROT	0-7h 0 1h 2h 3h 4h 5h 6h 7h	Right-rotation value for transmit rotate right format unit. 0 Rotate right by 0 (no rotation). 1h Rotate right by 4 bit positions. 2h Rotate right by 8 bit positions. 3h Rotate right by 12 bit positions. 4h Rotate right by 16 bit positions. 5h Rotate right by 20 bit positions. 6h Rotate right by 24 bit positions. 7h Rotate right by 28 bit positions.

### 20.1.28 Transmit Frame Sync Control Register (AFSXCTL)

The transmit frame sync control register (AFSXCTL) configures the transmit frame sync (AFSX). The AFSXCTL is shown in [Figure 20-61](#) and described in [Table 20-36](#).

**Figure 20-61. Transmit Frame Sync Control Register (AFSXCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-36. Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-7	XMOD	0-1FFh 0 1h 2h-20h 21h-17Fh 180h 181h-1FFh	Transmit frame sync mode select bits. Burst mode Reserved 2-slot TDM (I2S mode) to 32-slot TDM Reserved 384-slot DIT mode Reserved
6-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	FXWID	0 1	Transmit frame sync width select bit indicates the width of the transmit frame sync (AFSX) during its active period. Single bit Single word
3-2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	FSXM	0 1	Transmit frame sync generation select bit. Externally-generated transmit frame sync Internally-generated transmit frame sync
0	FSXP	0 1	Transmit frame sync polarity select bit. A rising edge on transmit frame sync (AFSX) indicates the beginning of a frame. A falling edge on transmit frame sync (AFSX) indicates the beginning of a frame.

### 20.1.29 Transmit Clock Control Register (ACLKXCTL)

The transmit clock control register (ACLKXCTL) configures the transmit bit clock (ACLKX) and the transmit clock generator. The ACLKXCTL is shown in [Figure 20-62](#) and described in [Table 20-37](#).

**Figure 20-62. Transmit Clock Control Register (ACLKXCTL)**

31	Reserved <sup>(A)</sup>					16
R-0						
15	8	7	6	5	4	0
Reserved <sup>(A)</sup>			CLKXP	ASYNC	CLKXM	CLKXDIV
R-0			R/W-0	R/W-1	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

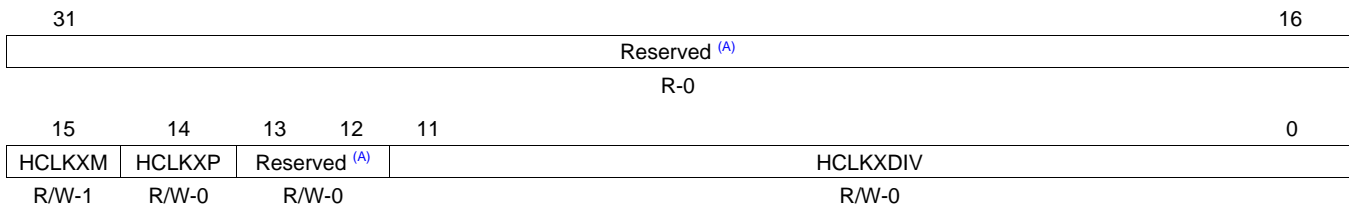
**Table 20-37. Transmit Clock Control Register (ACLKXCTL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	CLKXP	0 1	Transmit bitstream clock polarity select bit. 0 Rising edge. External receiver samples data on the falling edge of the serial clock, so the transmitter must shift data out on the rising edge of the serial clock. 1 Falling edge. External receiver samples data on the rising edge of the serial clock, so the transmitter must shift data out on the falling edge of the serial clock.
6	ASYNC	0 1	Transmit/receive operation asynchronous enable bit. 0 Synchronous. Transmit clock and frame sync provides the source for both the transmit and receive sections. Note that in this mode, the receive bit clock is an inverted version of the transmit bit clock. See <a href="#">Section 20.0.27.1.5</a> for more details. 1 Asynchronous. Separate clock and frame sync used by transmit and receive sections.
5	CLKXM	0 1	Transmit bit clock source bit. 0 External transmit clock source from ACLKX pin. 1 Internal transmit clock source from output of programmable bit clock divider.
4-0	CLKXDIV	0-1Fh 0 1h 2h-1Fh	Transmit bit clock divide ratio bits determine the divide-down ratio from AHCLKX to ACLKX. 0 Divide-by-1 1h Divide-by-2 2h-1Fh Divide-by-3 to divide-by-32

### 20.1.30 Transmit High-Frequency Clock Control Register (AHCLKXCTL)

The transmit high-frequency clock control register (AHCLKXCTL) configures the transmit high-frequency master clock (AHCLKX) and the transmit clock generator. The AHCLKXCTL is shown in [Figure 20-63](#) and described in [Table 20-38](#).

**Figure 20-63. Transmit High-Frequency Clock Control Register (AHCLKXCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-38. Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15	HCLKXM	0 1	Transmit high-frequency clock source bit. 0 External transmit high-frequency clock source from AHCLKX pin. 1 Internal transmit high-frequency clock source from output of programmable high clock divider.
14	HCLKXP	0 1	Transmit bitstream high-frequency clock polarity select bit. 0 Not inverted. AHCLKX is not inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin. 1 Inverted. AHCLKX is inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin.
13-12	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
11-0	HCLKXDIV	0-FFFh 0 1h 2h-FFFh	Transmit high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKX. 0 Divide-by-1 1h Divide-by-2 2h-FFFh Divide-by-3 to divide-by-4096

### 20.1.31 Transmit TDM Time Slot Register (XTDM)

The transmit TDM time slot register (XTDM) specifies in which TDM time slot the transmitter is active. TDM time slot counter range is extended to 384 slots (to support SPDIF blocks of 384 subframes). XTDM operates modulo 32, that is, XTDM specifies the TDM activity for time slots 0, 32, 64, 96, 128, etc. The XTDM is shown in [Figure 20-64](#) and described in [Table 20-39](#).

**Figure 20-64. Transmit TDM Time Slot Register (XTDM)**

31	30	29	28	27	26	25	24
XTDMS31	XTDMS30	XTDMS29	XTDMS28	XTDMS27	XTDMS26	XTDMS25	XTDMS24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
XTDMS23	XTDMS22	XTDMS21	XTDMS20	XTDMS19	XTDMS18	XTDMS17	XTDMS16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
XTDMS15	XTDMS14	XTDMS13	XTDMS12	XTDMS11	XTDMS10	XTDMS9	XTDMS8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
XTDMS7	XTDMS6	XTDMS5	XTDMS4	XTDMS3	XTDMS2	XTDMS1	XTDMS0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 20-39. Transmit TDM Time Slot Register (XTDM) Field Descriptions**

Bit	Field	Value	Description
31-0	XTDMS[31-0]	0	Transmitter mode during TDM time slot <i>n</i> . Transmit TDM time slot <i>n</i> is inactive. The transmit serializer does not shift out data during this slot.
		1	Transmit TDM time slot <i>n</i> is active. The transmit serializer shifts out data during this slot according to the serializer control register (SRCTL).

### 20.1.32 Transmitter Interrupt Control Register (XINTCTL)

The transmitter interrupt control register (XINTCTL) controls generation of the McASP transmit interrupt (XINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates XINT. The XINTCTL is shown in Figure 20-65 and described in Table 20-40. See Section 20.1.33 for a description of the interrupt conditions.

**Figure 20-65. Transmitter Interrupt Control Register (XINTCTL)**

31	Reserved <sup>(A)</sup>							8
R-0								
7	6	5	4	3	2	1	0	
XSTAFRM	Reserved <sup>(A)</sup>	XDATA	XLAST	XDMAERR	XCKFAIL	XSYNCERR	XUNDRN	
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-40. Transmitter Interrupt Control Register (XINTCTL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	XSTAFRM	0	Transmit start of frame interrupt enable bit. Interrupt is disabled. A transmit start of frame interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit start of frame interrupt generates a McASP transmit interrupt (XINT).
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	XDATA	0	Transmit data ready interrupt enable bit. Interrupt is disabled. A transmit data ready interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit data ready interrupt generates a McASP transmit interrupt (XINT).
4	XLAST	0	Transmit last slot interrupt enable bit. Interrupt is disabled. A transmit last slot interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit last slot interrupt generates a McASP transmit interrupt (XINT).
3	XDMAERR	0	Transmit DMA error interrupt enable bit. Interrupt is disabled. A transmit DMA error interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit DMA error interrupt generates a McASP transmit interrupt (XINT).
2	XCKFAIL	0	Transmit clock failure interrupt enable bit. Interrupt is disabled. A transmit clock failure interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit clock failure interrupt generates a McASP transmit interrupt (XINT).
1	XSYNCERR	0	Unexpected transmit frame sync interrupt enable bit. Interrupt is disabled. An unexpected transmit frame sync interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. An unexpected transmit frame sync interrupt generates a McASP transmit interrupt (XINT).
0	XUNDRN	0	Transmitter underrun interrupt enable bit. Interrupt is disabled. A transmitter underrun interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmitter underrun interrupt generates a McASP transmit interrupt (XINT).

### 20.1.33 Transmitter Status Register (XSTAT)

The transmitter status register (XSTAT) provides the transmitter status and transmit TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The XSTAT is shown in [Figure 20-66](#) and described in [Table 20-41](#).

**Figure 20-66. Transmitter Status Register (XSTAT)**

Reserved <sup>(A)</sup>							XERR	
R-0							R/W-0	
31							9	8
7	6	5	4	3	2	1	0	
XDMAERR	XSTAFRM	XDATA	XLAST	XTDMSLOT	XCKFAIL	XSYNCERR	XUNDRN	
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-41. Transmitter Status Register (XSTAT) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	XERR	0 1	XERR bit always returns a logic-OR of: XUNDRN   XSYNCERR   XCKFAIL   XDMAERR Allows a single bit to be checked to determine if a transmitter error interrupt has occurred. 0 No errors have occurred. 1 An error has occurred.
7	XDMAERR	0 1	Transmit DMA error flag. XDMAERR is set when the CPU or DMA writes more serializers through the DMA port in a given time slot than were programmed as transmitters. Causes a transmit interrupt (XINT), if this bit is set and XDMAERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 Transmit DMA error did not occur. 1 Transmit DMA error did occur.
6	XSTAFRM	0 1	Transmit start of frame flag. Causes a transmit interrupt (XINT), if this bit is set and XSTAFRM in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No new transmit frame sync (AFSX) is detected. 1 A new transmit frame sync (AFSX) is detected.
5	XDATA	0 1	Transmit data ready flag. Causes a transmit interrupt (XINT), if this bit is set and XDATA in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 XBUF is written and is full. 1 Data is copied from XBUF to XRSR. XBUF is empty and ready to be written. XDATA is also set when the transmit serializers are taken out of reset. When XDATA is set, it always causes a DMA event (AXEVT).
4	XLAST	0 1	Transmit last slot flag. XLAST is set along with XDATA, if the current slot is the last slot in a frame. Causes a transmit interrupt (XINT), if this bit is set and XLAST in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 Current slot is not the last slot in a frame. 1 Current slot is the last slot in a frame. XDATA is also set.
3	XTDMSLOT	0 1	Returns the LSB of XSLOT. Allows a single read of XSTAT to determine whether the current TDM time slot is even or odd. 0 Current TDM time slot is odd. 1 Current TDM time slot is even.
2	XCKFAIL	0 1	Transmit clock failure flag. XCKFAIL is set when the transmit clock failure detection circuit reports an error (see <a href="#">Section 20.0.27.6.6</a> ). Causes a transmit interrupt (XINT), if this bit is set and XCKFAIL in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 Transmit clock failure did not occur. 1 Transmit clock failure did occur.



**Table 20-41. Transmitter Status Register (XSTAT) Field Descriptions (continued)**

Bit	Field	Value	Description
1	XSYNCERR	0 1	Unexpected transmit frame sync flag. XSYNCERR is set when a new transmit frame sync (AFSX) occurs before it is expected. Causes a transmit interrupt (XINT), if this bit is set and XSYNCERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. Unexpected transmit frame sync did not occur. Unexpected transmit frame sync did occur.
0	XUNDRN	0 1	Transmitter underrun flag. XUNDRN is set when the transmit serializer is instructed to transfer data from XBUF to XRSR, but XBUF has not yet been serviced with new data since the last transfer. Causes a transmit interrupt (XINT), if this bit is set and XUNDRN in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. Transmitter underrun did not occur. Transmitter underrun did occur. See <a href="#">Section 20.0.27.6.2</a> for details on McASP action upon underrun conditions.

### 20.1.34 Current Transmit TDM Time Slot Register (XSLOT)

The current transmit TDM time slot register (XSLOT) indicates the current time slot for the transmit data frame. The XSLOT is shown in [Figure 20-67](#) and described in [Table 20-42](#).

**Figure 20-67. Current Transmit TDM Time Slot Register (XSLOT)**

31	Reserved <sup>(A)</sup>			16
R-0				
15	9	8	0	
Reserved <sup>(A)</sup>			XSLOT CNT	
R-0			R-17Fh	

LEGEND: R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-42. Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions**

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8-0	XSLOT CNT	0-17Fh	Current transmit time slot count. Legal values: 0 to 383 (17Fh). During reset, this counter value is 383 so the next count value, which is used to encode the first DIT group of data, will be 0 and encodes the B preamble. TDM function is not supported for > 32 time slots. However, TDM time slot counter may count to 383 when used to transmit a DIT block.

### 20.1.35 Transmit Clock Check Control Register (XCLKCHK)

The transmit clock check control register (XCLKCHK) configures the transmit clock failure detection circuit. The XCLKCHK is shown in [Figure 20-68](#) and described in [Table 20-43](#).

**Figure 20-68. Transmit Clock Check Control Register (XCLKCHK)**

31	24	23	16
XCNT			XMAX
R-0			R/W-0
15	8	7	0
XMIN	Reserved <sup>(A)</sup>		XPS
R/W-0	R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-43. Transmit Clock Check Control Register (XCLKCHK) Field Descriptions**

Bit	Field	Value	Description
31-24	XCNT	0	Transmit clock count value (from previous measurement). The clock circuit continually counts the number of DSP system clocks for every 32 transmit high-frequency master clock (AHCLKX) signals, and stores the count in XCNT until the next measurement is taken.
23-16	XMAX	0-FFh	Transmit clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If the current counter value is greater than XMAX after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.
15-8	XMIN	0-FFh	Transmit clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If XCNT is less than XMIN after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.
7-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-0	XPS	0-Fh	Transmit clock check prescaler value.
		0	McASP system clock divided by 1
		1h	McASP system clock divided by 2
		2h	McASP system clock divided by 4
		3h	McASP system clock divided by 8
		4h	McASP system clock divided by 16
		5h	McASP system clock divided by 32
		6h	McASP system clock divided by 64
		7h	McASP system clock divided by 128
		8h	McASP system clock divided by 256
		9h-Fh	Reserved

### 20.1.36 Transmitter DMA Event Control Register (XEVTCTL)

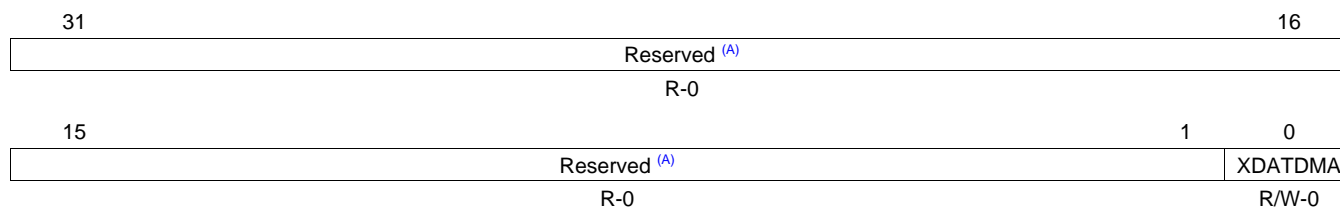
The transmitter DMA event control register (XEVTCTL) is shown in [Figure 20-69](#) and described in [Table 20-44](#).

**CAUTION**

DSP specific registers

Accessing XEVTCTL not implemented on a specific DSP may cause improper device operation.

**Figure 20-69. Transmitter DMA Event Control Register (XEVTCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

**Table 20-44. Transmitter DMA Event Control Register (XEVTCTL) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	XDATDMA	0	Transmit data DMA request is enabled.
		1	Reserved.

### 20.1.37 Serializer Control Registers (SRCTL<sub>n</sub>)

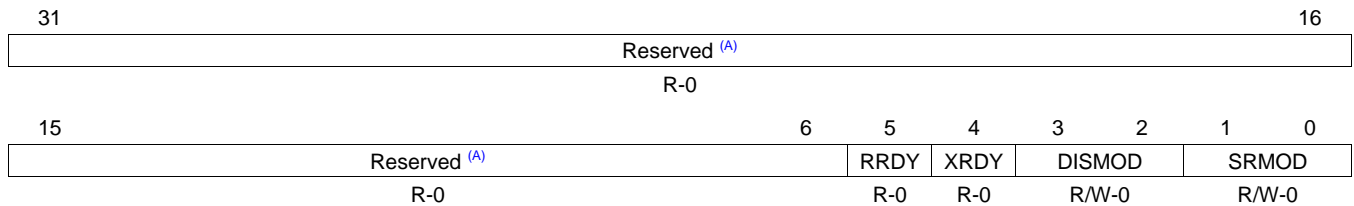
Each serializer on the McASP has a serializer control register (SRCTL). There are up to 16 serializers per McASP. The SRCTL is shown in [Figure 20-70](#) and described in [Table 20-45](#).

**CAUTION**

DSP specific registers

Accessing SRCTL<sub>n</sub> not implemented on a specific DSP may cause improper device operation.

**Figure 20-70. Serializer Control Registers (SRCTL<sub>n</sub>)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

A If writing to this field, always write the default value for future device compatibility.

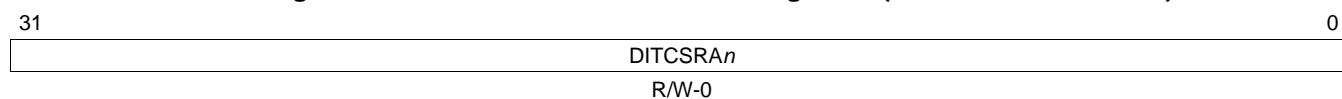
**Table 20-45. Serializer Control Registers (SRCTL<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	RRDY	0 1	Receive buffer ready bit. RRDY indicates the current receive buffer state. Always reads 0 when programmed as a transmitter or as inactive. If SRMOD bit is set to receive (2h), RRDY switches from 0 to 1 whenever data is transferred from XRSR to RBUF. 0 Receive buffer (RBUF) is empty. 1 Receive buffer (RBUF) contains data and needs to be read before the start of the next time slot or a receiver overrun occurs.
4	XRDY	0 1	Transmit buffer ready bit. XRDY indicates the current transmit buffer state. Always reads 0 when programmed as a receiver or as inactive. If SRMOD bit is set to transmit (1h), XRDY switches from 0 to 1 when XSRCLR in GBLCTL is switched from 0 to 1 to indicate an empty transmitter. XRDY remains set until XSRCLR is forced to 0, data is written to the corresponding transmit buffer, or SRMOD bit is changed to receive (2h) or inactive (0). 0 Transmit buffer (XBUF) contains data. 1 Transmit buffer (XBUF) is empty and needs to be written before the start of the next time slot or a transmit underrun occurs.
3-2	DISMOD	0-3h 0 1h 2h 3h	Serializer pin drive mode bit. Drive on pin when in inactive TDM slot of transmit mode or when serializer is inactive. This field only applies if the pin is configured as a McASP pin (PFUNC = 0). 0 Drive on pin is 3-state. 1h Reserved 2h Drive on pin is logic low. 3h Drive on pin is logic high.
1-0	SRMOD	0-3h 0 1h 2h 3h	Serializer mode bit. 0 Serializer is inactive. 1h Serializer is transmitter. 2h Serializer is receiver. 3h Reserved

### 20.1.38 DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)

The DIT left channel status registers (DITCSRA) provide the status of each left channel (even TDM time slot). Each of the six 32-bit registers (Figure 20-71) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

**Figure 20-71. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)**

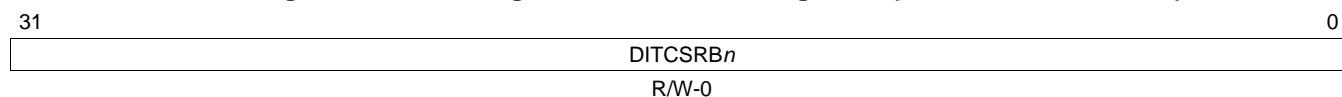


LEGEND: R/W = Read/Write; -n = value after reset

### 20.1.39 DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)

The DIT right channel status registers (DITCSRB) provide the status of each right channel (odd TDM time slot). Each of the six 32-bit registers (Figure 20-72) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

**Figure 20-72. DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)**

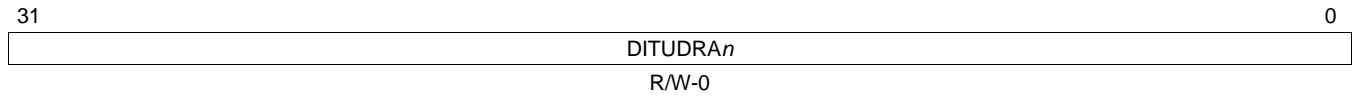


LEGEND: R/W = Read/Write; -n = value after reset

### 20.1.40 DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5)

The DIT left channel user data registers (DITUDRA) provides the user data of each left channel (even TDM time slot). Each of the six 32-bit registers (Figure 20-73) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

**Figure 20-73. DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5)**

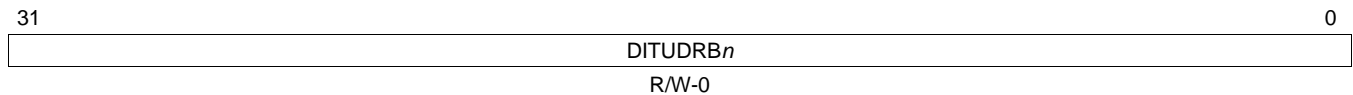


LEGEND: R/W = Read/Write; -n = value after reset

### 20.1.41 DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5)

The DIT right channel user data registers (DITUDRB) provides the user data of each right channel (odd TDM time slot). Each of the six 32-bit registers (Figure 20-74) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

**Figure 20-74. DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5)**



LEGEND: R/W = Read/Write; -n = value after reset

### 20.1.42 Transmit Buffer Registers (XBUF<sub>n</sub>)

The transmit buffers for the serializers (XBUF) hold data from the transmit format unit. For transmit operations, the XBUF (Figure 20-75) is an alias of the XRBUF in the serializer. The XBUF can be accessed through the peripheral configuration port (Table 20-7) or through the DMA port (Table 20-8).

**CAUTION**

DSP specific registers

Accessing XBUF registers not implemented on a specific DSP may cause improper device operation.

**Figure 20-75. Transmit Buffer Registers (XBUF<sub>n</sub>)**



LEGEND: R/W = Read/Write; -n = value after reset

### 20.1.43 Receive Buffer Registers (RBUF<sub>n</sub>)

The receive buffers for the serializers (RBUF) hold data from the serializer before the data goes to the receive format unit. For receive operations, the RBUF (Figure 20-76) is an alias of the XRBUF in the serializer. The RBUF can be accessed through the peripheral configuration port (Table 20-7) or through the DMA port (Table 20-8).

**CAUTION**

DSP specific registers

Accessing RBUF registers not implemented on a specific DSP may cause improper device operation.

**Figure 20-76. Receive Buffer Registers (RBUF<sub>n</sub>)**

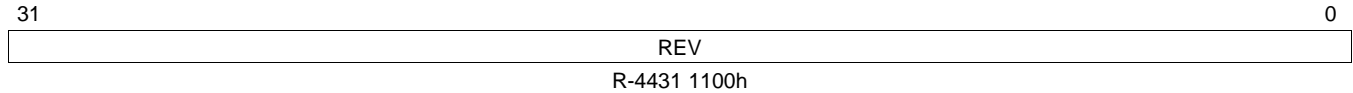


LEGEND: R/W = Read/Write; -n = value after reset

### 20.1.44 AFIFO Revision Identification Register (AFIFOREV)

The Audio FIFO (AFIFO) revision identification register (AFIFOREV) contains revision data for the Audio FIFO (AFIFO). The AFIFOREV is shown in [Figure 20-77](#) and described in [Table 20-46](#).

**Figure 20-77. AFIFO Revision Identification Register (AFIFOREV)**



LEGEND: R = Read only; -n = value after reset

**Table 20-46. AFIFO Revision Identification Register (AFIFOREV) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4431 1100h	Identifies revision of Audio FIFO.



### 20.1.45 Write FIFO Control Register (WFIFOCTL)

The Write FIFO control register (WFIFOCTL) is shown in [Figure 20-78](#) and described in [Table 20-47](#).

**NOTE:** The WNUMEVT and WNUMDMA values must be set prior to enabling the Write FIFO.  
If the Write FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

**Figure 20-78. Write FIFO Control Register (WFIFOCTL)**

31	Reserved	17	16
	R-0		WENA R/W-0
15	8	7	0
	WNUMEVT R/W-10h		WNUMDMA R/W-4h

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

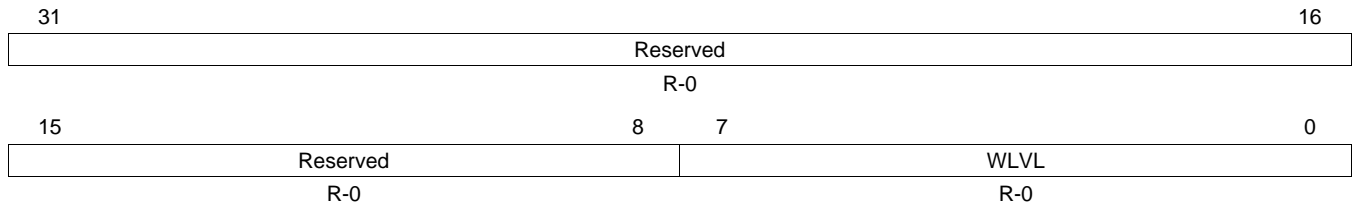
**Table 20-47. Write FIFO Control Register (WFIFOCTL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	WENA	0	Write FIFO is disabled. The WLVL field in the Write FIFO status register (WFIFOSTS) is reset to 0 and pointers are initialized, that is, the Write FIFO is "flushed."
		1	Write FIFO is enabled. If Write FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.
15-8	WNUMEVT	0-FFh	Write word count per DMA event (32-bit). When the Write FIFO has space for at least WNUMEVT words of data, then an AXEVT (transmit DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as transmitters. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-40h	3 to 64 words
		41h-FFh	Reserved
7-0	WNUMDMA	0-FFh	Write word count per transfer (32-bit words). Upon a transmit DMA event from the McASP, WNUMDMA words are transferred from the Write FIFO to the McASP. This value must equal the number of McASP serializers (not the number of channels) used as transmitters. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-10h	3-16 words
		11h-FFh	Reserved

### 20.1.46 Write FIFO Status Register (WFIFOSTS)

The Write FIFO status register (WFIFOSTS) is shown in [Figure 20-79](#) and described in [Table 20-48](#).

**Figure 20-79. Write FIFO Status Register (WFIFOSTS)**



LEGEND: R = Read only; -n = value after reset

**Table 20-48. Write FIFO Status Register (WFIFOSTS) Field Descriptions**

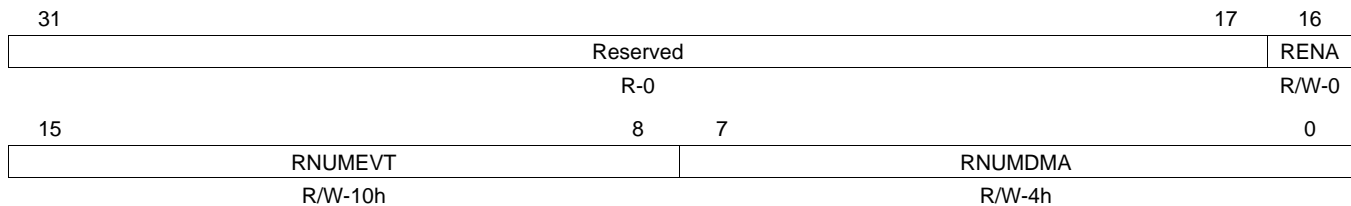
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	WLVL	0-FFh	Write level (read-only). Number of 32-bit words currently in the Write FIFO.
		0	0 words currently in Write FIFO.
		1h	1 word currently in Write FIFO.
		2h	2 words currently in Write FIFO.
		3h-40h	3 to 64 words currently in Write FIFO.
		41h-FFh	Reserved

### 20.1.47 Read FIFO Control Register (RFIFOCTL)

The Read FIFO control register (RFIFOCTL) is shown in [Figure 20-80](#) and described in [Table 20-49](#).

**NOTE:** The RNUMEVT and RNUMDMA values must be set prior to enabling the Read FIFO.  
If the Read FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

**Figure 20-80. Read FIFO Control Register (RFIFOCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

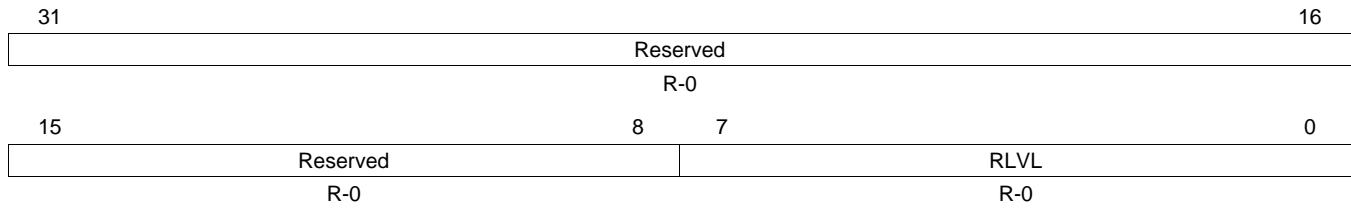
**Table 20-49. Read FIFO Control Register (RFIFOCTL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	RENA	0	Read FIFO is disabled. The RLVL bit in the Read FIFO status register (RFIFOSTS) is reset to 0 and pointers are initialized, that is, the Read FIFO is “flushed.”
		1	Read FIFO is enabled. If Read FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.
15-8	RNUMEVT	0-FFh	Read word count per DMA event (32-bit). When the Read FIFO contains at least RNUMEVT words of data, then an AREVT (receive DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as receivers. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-40h	3 to 64 words
		41h-FFh	Reserved
7-0	RNUMDMA	0-FFh	Read word count per transfer (32-bit words). Upon a receive DMA event from the McASP, the Read FIFO reads RNUMDMA words from the McASP. This value must equal the number of McASP serializers used as receivers. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1	1 word
		2	2 words
		3h-10h	3-16 words
		11h-FFh	Reserved

### 20.1.48 Read FIFO Status Register (RFIFOSTS)

The Read FIFO status register (RFIFOSTS) is shown in [Figure 20-81](#) and described in [Table 20-50](#).

**Figure 20-81. Read FIFO Status Register (RFIFOSTS)**



LEGEND: R = Read only; -n = value after reset

**Table 20-50. Read FIFO Status Register (RFIFOSTS) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RLVL	0-FFh	Read level (read-only). Number of 32-bit words currently in the Read FIFO.
		0	0 words currently in Read FIFO.
		1h	1 word currently in Read FIFO.
		2h	2 words currently in Read FIFO.
		3h-40h	3 to 64 words currently in Read FIFO.
		41h-FFh	Reserved

---

---

## ***Multichannel Buffered Serial Port (McBSP)***

---

---

This chapter describes the multichannel buffered serial port (McBSP). See your device-specific data manual to determine how many McBSPs are available on your device.

<b>Topic</b>	<b>Page</b>
<b>21.1 Introduction</b> .....	<b>891</b>
<b>21.2 Architecture</b> .....	<b>893</b>
<b>21.3 Registers</b> .....	<b>940</b>

## 21.1 Introduction

### 21.1.1 Purpose of the Peripheral

The primary use for the multichannel buffered serial port (McBSP) is for audio interface purposes. The McBSP is a specialized version of the McBSP peripheral used on other TI DSPs. The primary audio modes that are supported are the AC97 and IIS modes. In addition to the primary audio modes, the McBSP can be programmed to support other serial formats but is not intended to be used as a high-speed interface.

The McBSP consists of a data path and a control path that connect to external devices. Separate pins for transmission and reception communicate data to these external devices. The CPU communicates to the McBSP using 32-bit-wide control registers accessible via the internal peripheral bus.

### 21.1.2 Features

The McBSP provides the following functions:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected analog-to-digital (A/D) and digital-to-analog (D/A) devices
- External shift clock or an internal, programmable frequency shift clock for data transfer

In addition, the McBSP has the following capabilities:

- Direct interface to:
  - T1/E1 framers
  - MVIP switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - AC97 compliant devices (the necessary multiphase frame synchronization capability is provided)
  - IIS compliant devices
- Multichannel transmit and receive of up to 128 channels
- A wide selection of data sizes, including 8, 12, 16, 20, 24, and 32 bits
- $\mu$ -Law and A-Law companding
- 8-bit data transfers with the option of LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation
- Additional McBSP Buffer FIFO (BFIFO):
  - Provides additional data buffering
  - Provides added tolerance to variations in host/DMA controller response times
  - May be used as a DMA event pacer
  - Independent Read FIFO and Write FIFO
  - 256 bytes of RAM for each FIFO (read and write)
  - Option to bypass Write FIFO and/or Read FIFO, independently

### 21.1.3 Functional Block Diagram

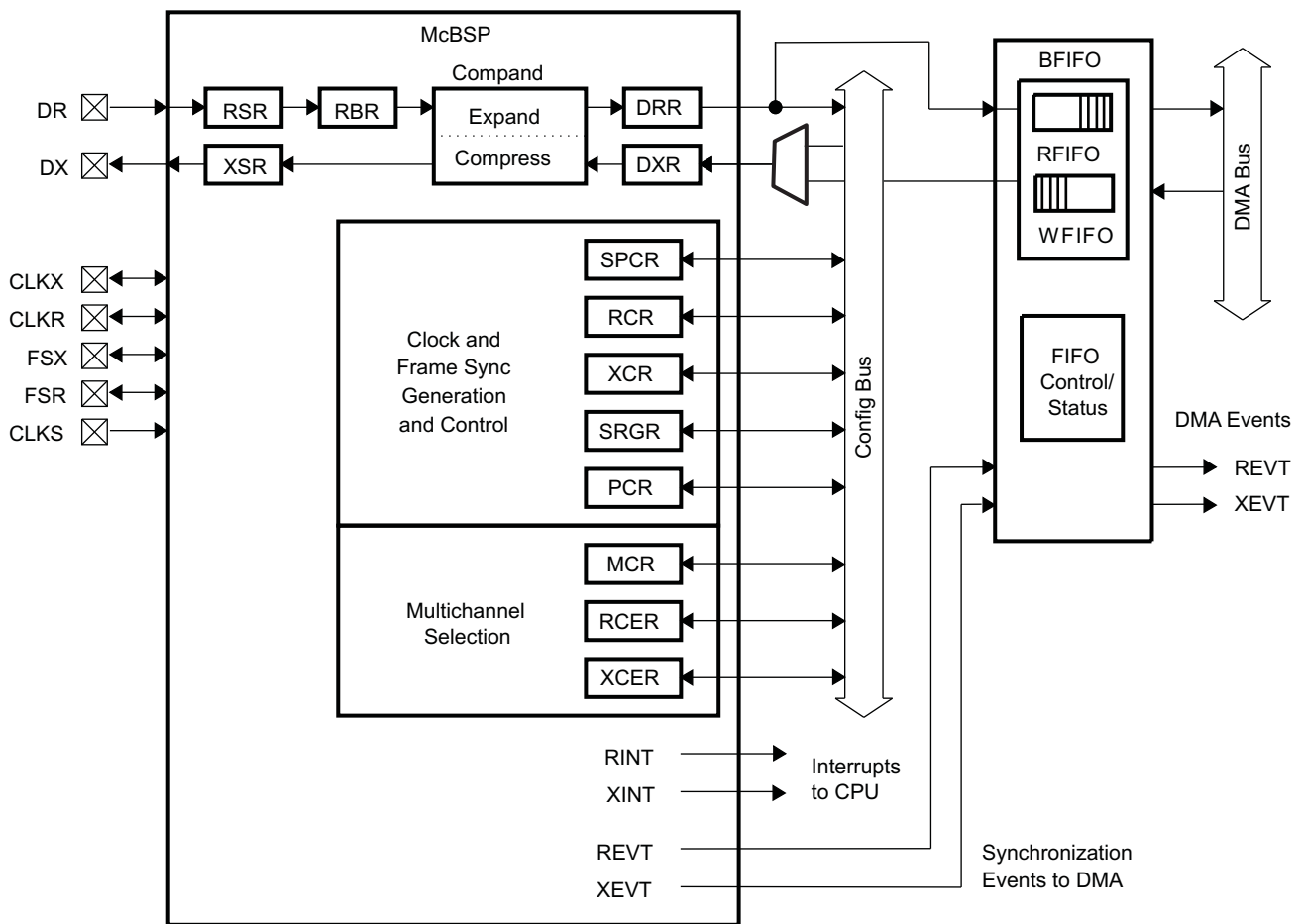
The McBSP consists of a data path and control path, as shown in [Figure 21-1](#)

### 21.1.4 Industry Standard Compliance Statement

The McBSP supports the following industry standard interfaces:

- AC97**— The AC97 standard specifies a 5-wire digital serial link between an audio codec device and its digital controller.
- IIS**— IIS is a protocol for transmitting two channels of digital audio data over a single serial connection. The IIS bus is an industry standard three-wire interface for streaming stereo audio between devices, typically between a CPU/DSP and a DAC/ADC.

**Figure 21-1. McBSP Block Diagram**



## 21.2 Architecture

This section describes the architecture of the McBSP.

### 21.2.1 Clock Control

The McBSP can use an internal or external clock source. Either clock source can be divided-down inside the McBSP to generate the actual interface bit clock frequency. For detailed timing information, see the device-specific data manual. Detailed information about how the interface clock and frame synchronization signals are generated is provided in [Section 21.2.5](#).

### 21.2.2 Signal Descriptions

The signals used on the audio interface are listed in [Table 21-1](#).

**Table 21-1. McBSP Interface Signals**

Pin	I/O/Z	Description
CLKR	I/O/Z	Receive clock - supplies or receives a reference clock for the receiver; or supplies a reference clock to the sample rate generator
CLKS	I	Supplies the input clock of the sample rate generator
CLKX	I/O/Z	Transmit clock - supplies or receives a reference clock for the transmitter; or supplies a reference clock to the sample rate generator
DR	I	Received serial data
DX	O/Z	Transmitted serial data
FSR	I/O/Z	Receive frame synchronization - control signal to synchronize the start of received data
FSX	I/O/Z	Transmit frame synchronization - control signal to synchronize the start of transmitted data

### 21.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data manual to determine how pin multiplexing affects the McBSP.

### 21.2.4 Endianness Considerations

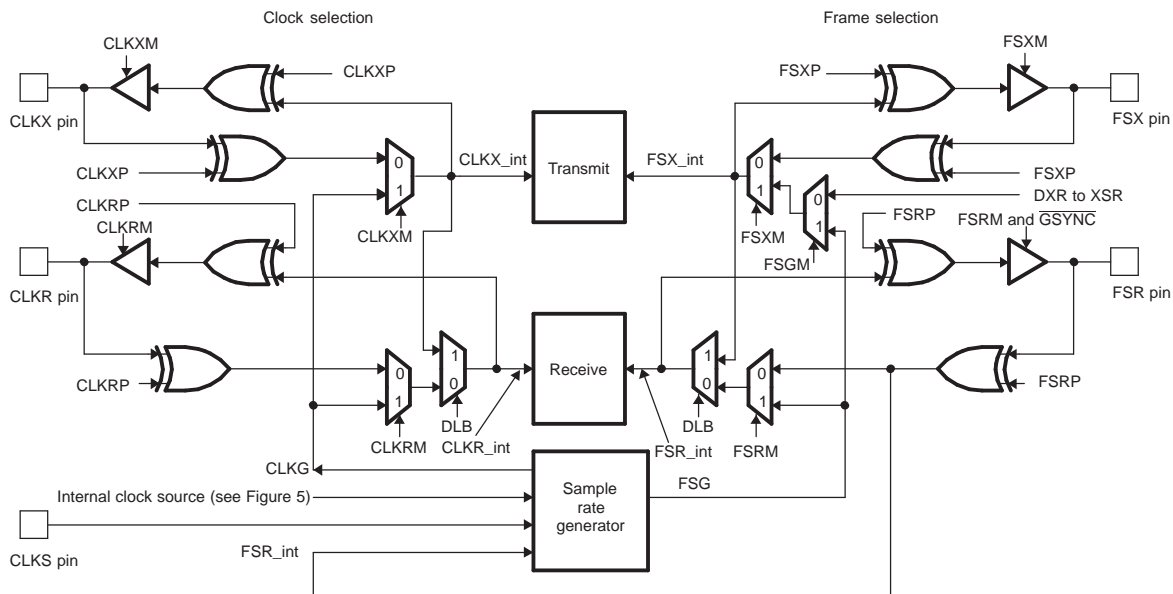
There are no endianness considerations for the McBSP.



### 21.2.5 Clock, Frames, and Data

The McBSP has several ways of selecting clocking and framing for both the receiver and transmitter. Clocking and framing can be sent to both portions by the sample rate generator. Each portion can select external clocking and/or framing independently. Figure 21-2 is a block diagram of the clock and frame selection circuitry.

Figure 21-2. Clock and Frame Generation



#### 21.2.5.1 Frame and Clock Operation

Receive and transmit frame sync pulses (FSR/X), and clocks (CLKR/X), can either be generated internally by the sample rate generator (see Section 21.2.5.2) or be driven by an external source. The source of frame sync and clock is selected by programming the mode bits, FS(R/X)M and CLK(R/X)M respectively, in the pin control register (PCR). FSR is also affected by the GSYNC bit in the sample rate generator register (SRGR), see Section 21.2.5.4.2 for details.

When FSR and FSX are inputs (FSXM = FSRM = 0), the McBSP detects them on the internal falling edge of clock, CLKR\_int and CLKX\_int, respectively (see Figure 21-2). The receive data arriving at the DR pin is also sampled on the falling edge of CLKR\_int. These internal clock signals are either derived from an external source via the CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X)\_int. Similarly, data on DX is output on the rising edge of CLKX\_int.

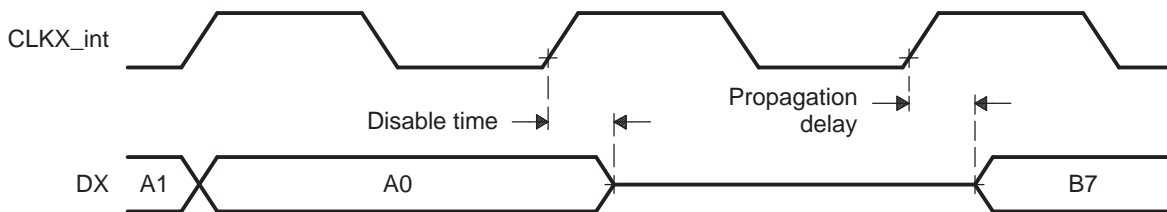
The FSRP, FSXP, CLKRP, and CLKXP bits in PCR configure the polarities of FSR, FSX, CLKR, and CLKX. All frame sync signals (FSR\_int and FSX\_int) internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to the McBSP) and FSRP = FSXP = 1, the external active (low) frame sync signals are inverted before being sent to the receiver signal (FSR\_int) and transmitter signal (FSX\_int). Similarly, if internal synchronization is selected (FSR/FSX are outputs and GSYNC = 0), the internal active (high) sync signals are inverted if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. Figure 21-2 shows this inversion using XOR gates.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of CLKX\_int (see [Figure 21-3](#)). If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge-triggered input clock on CLKX is inverted to a rising-edge-triggered clock before being sent to the transmitter. If CLKXP = 1 and internal clocking is selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge-triggered) clock, CLKX\_int, is inverted before being sent out on the CLKX pin.

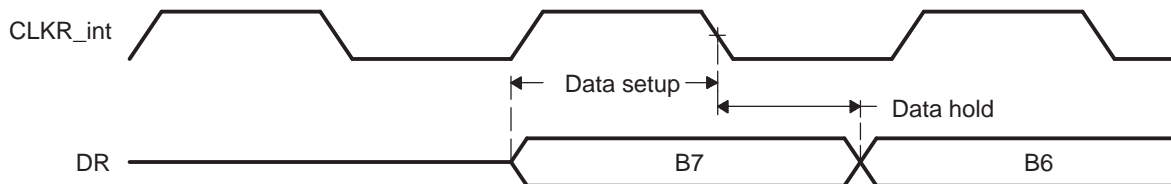
Similarly, the receiver can reliably sample data that is clocked (by the transmitter) with a rising-edge clock. The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of CLKR\_int (see [Figure 21-4](#)). Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge-triggered clock is inverted to a rising edge before being sent out on the CLKR pin.

In a system where the same clock (internal or external) is used to clock the receiver and transmitter, CLKRP = CLKXP. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold times of data around this edge. [Figure 21-4](#) shows how data clocked by an external serial device using a rising-edge clock can be sampled by the McBSP receiver with the falling edge of the same clock.

**Figure 21-3. Transmit Data Clocking**



**Figure 21-4. Receive Data Clocking**



### 21.2.5.2 Sample Rate Generator Clocking and Framing

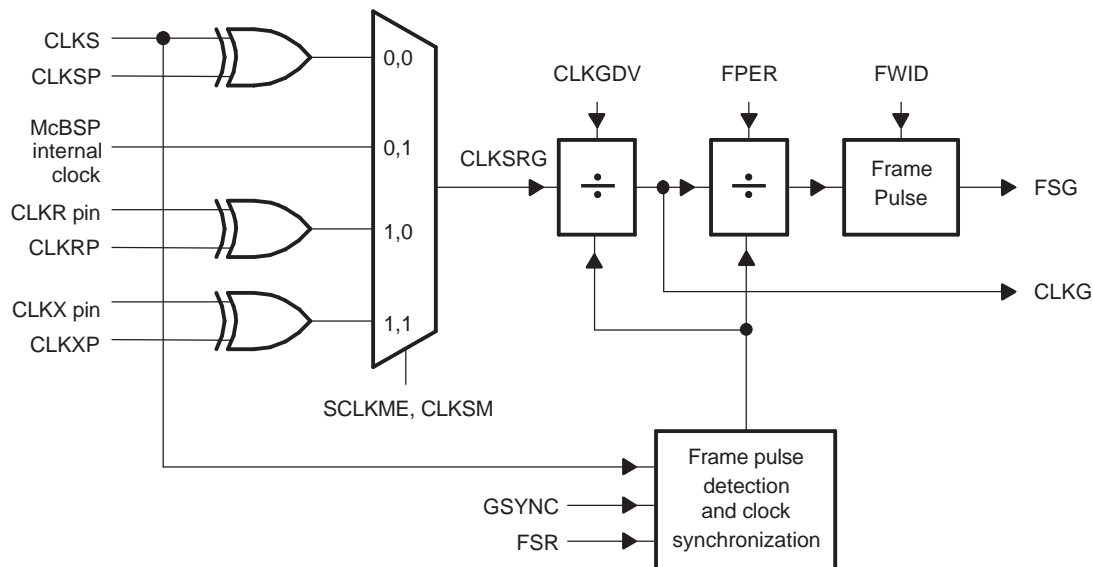
The sample rate generator is composed of a 3-stage clock divider that provides a programmable data clock (CLKG) and framing signal (FSG), as shown in Figure 21-5. CLKG and FSG are McBSP internal signals that can be programmed to drive receive and/or transmit clocking, CLK(R/X), and framing, FS(R/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source.

The sample rate generator is not used when CLKX, FSX, CLKR, and FSR are driven by an external source. Therefore, the GRST bit in the serial port control register (SPCR) does not need to be enabled (GRST = 1) for this setup. The three stages of the sample rate generator circuit compute:

- Clock divide-down (CLKGDV): The number of input clocks per data bit clock
- Frame period (FPER): The frame period in data bit clocks
- Frame width (FWID): The width of an active frame pulse in data bit clocks

In addition, a frame pulse detection and clock synchronization module allows synchronization of the clock divide-down with an incoming frame pulse. The operation of the sample rate generator during device reset is described in Section 21.2.11.

**Figure 21-5. Sample Rate Generator Block Diagram**



### 21.2.5.3 Data Clock Generation

When the receive/transmit clock mode is set to 1 (CLK(R/X)M = 1 in the pin control register (PCR)), the data clocks (CLK(R/X)) are driven by the internal sample rate generator output clock, CLKG. You can select for the receiver and transmitter from a variety of data bit clocks including:

- The input clock to the sample rate generator, which can be either the internal clock source or a dedicated external clock source via the CLKX, CLKR, or CLKS pins. Internally, the McBSP clock is selected to be either PLL0\_SYSCLK2 or PLL1\_SYSCLK2 by configuring the ASYNC3\_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module. See [Section 21.2.5.3.1](#) for details on the source of the McBSP internal clock.
- The input clock source (internal clock source or external clock CLKX/CLKR/CLKS) to the sample rate generator can be divided-down by a programmable value (CLKGDV bit in the sample rate generator register (SRGR)) to drive CLKG.

Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see [Figure 21-5](#)) generates CLKG and FSG.

#### 21.2.5.3.1 Input Clock Source Mode: CLKSM and SCLKME

The sample rate generator input clock signal can be driven from one of four sources selectable with the SCLKME bit in the pin control register (PCR) and the CLKSM bit in the sample rate generator register (SRGR), see [Table 21-2](#).

**Table 21-2. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits**

SCLKME Bit in PCR	CLKSM Bit in SRGR	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	McBSP internal input clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

#### 21.2.5.3.2 Sample Rate Generator Data Bit Clock Rate: CLKGDV

The first divider stage generates the serial data bit clock from the input clock. This divider stage uses a counter that is preloaded by the CLKGDV bit in the sample rate generator register (SRGR) and that contains the divide ratio value. The output of this stage is the data bit clock that is output on the sample rate generator output, CLKG, and that serves as the input for the second and third divider stages.

CLKG has a frequency equal to  $1/(\text{CLKGDV} + 1)$  of the sample rate generator input clock. Thus, the sample rate generator input clock frequency is divided by a value between 1 to 256. The CLKGDV value chosen must result in a clock that meets the timing requirements/limitations specified in the device-specific data manual.

When CLKGDV is an odd value or equal to 0, the CLKG duty cycle is 50%. Note that an odd CLKGDV value means an even divide down of the source clock and an even CLKGDV value means an odd divide down of the source clock. When CLKGDV is an even value (2p), the high state duration is p + 1 cycles and the low state duration is p cycles. This is illustrated in [Example 21-1](#), [Example 21-2](#), and [Example 21-3](#).

In the following examples:

$S_{IN}$  = sample generator input clock period

$f_{IN}$  = sample generator input clock frequency

$S_G$  = CLKG period

$f_G$  = CLKG frequency

The following equation is given above:  $f_G = f_{IN}/(\text{CLKGDV} + 1)$ ; therefore,  $S_G = (\text{CLKGDV} + 1) \times S_{IN}$ .

**Example 21-1. CLKGDV = 0**

$$\text{CLKGDV} = 0$$

$$S_G = (\text{CLKGDV} + 1) \times S_{\text{IN}} = (0 + 1) \times S_{\text{IN}} = S_{\text{IN}}$$

$$\text{Pulse width high} = S_{\text{IN}} \times (\text{CLKGDV} + 1)/2 = S_{\text{IN}} \times (0 + 1)/2 = 0.5 \times S_{\text{IN}}$$

$$\text{Pulse width low} = S_{\text{IN}} \times (\text{CLKGDV} + 1)/2 = S_{\text{IN}} \times (0 + 1)/2 = 0.5 \times S_{\text{IN}}$$

**Example 21-2. CLKGDV = 1**

$$\text{CLKGDV} = 1$$

$$S_G = (\text{CLKGDV} + 1) \times S_{\text{IN}} = (1 + 1) \times S_{\text{IN}} = 2 \times S_{\text{IN}}$$

$$\text{Pulse width high} = S_{\text{IN}} \times (\text{CLKGDV} + 1)/2 = S_{\text{IN}} \times (1 + 1)/2 = S_{\text{IN}}$$

$$\text{Pulse width low} = S_{\text{IN}} \times (\text{CLKGDV} + 1)/2 = S_{\text{IN}} \times (1 + 1)/2 = S_{\text{IN}}$$

**Example 21-3. CLKGDV = 2**

$$\text{CLKGDV} = 2$$

$$S_G = (\text{CLKGDV} + 1) \times S_{\text{IN}} = (2 + 1) \times S_{\text{IN}} = 3 \times S_{\text{IN}}$$

$$\text{Pulse width high} = S_{\text{IN}} \times (\text{CLKGDV}/2 + 1) = S_{\text{IN}} \times (2/2 + 1) = 2 \times S_{\text{IN}}$$

$$\text{Pulse width low} = S_{\text{IN}} \times \text{CLKGDV}/2 = S_{\text{IN}} \times 2/2 = 1 \times S_{\text{IN}}$$

**21.2.5.3.3 Bit Clock Polarity: CLKSP**

The external clock (CLKS) is selected to drive the sample rate generator clock divider by selecting CLKSM = 0 in the sample rate generator register (SRGR) and SCLKME = 0 in the pin control register (PCR). In this case, the CLKSP bit in SRGR selects the edge of CLKS on which sample rate generator data bit clock (CLKG) and frame sync signal (FSG) are generated. Since the rising edge of CLKSRG generates CLKG and FSG, the rising edge of CLKS when CLKSP = 0 or the falling edge of CLKS when CLKSP = 1 causes the transition on CLKG and FSG.

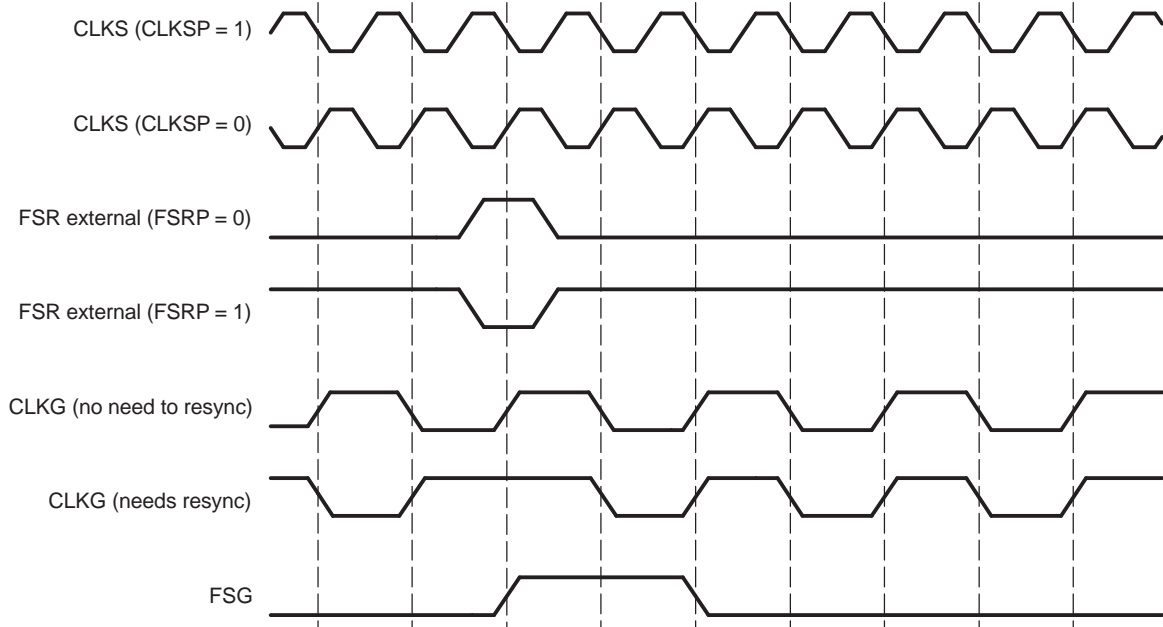
**21.2.5.3.4 Bit Clock and Frame Synchronization**

When the external clock (CLKS) is selected to drive the sample rate generator (CLKSM = 0 in SRGR and SCLKME = 0 in PCR), the GSYNC bit in SRGR can be used to configure the timing of CLKG relative to CLKS. GSYNC = 1 ensures that the McBSP and the external device to which it is communicating are dividing down the CLKS with the same phase relationship. If GSYNC = 0, this feature is disabled and CLKG runs freely and is not resynchronized. If GSYNC = 1, an inactive-to-active transition on FSR triggers a resynchronization of CLKG and the generation of FSG. CLKG always begins at a high state after synchronization. Also, FSR is always detected at the same edge of CLKS that generates CLKG, regardless of the length the FSR pulse. Although an external FSR is provided, FSG can still drive internal receive frame synchronization when GSYNC = 1. When GSYNC = 1, FPER does not matter, because the frame period is determined by the arrival of the external frame sync pulse.

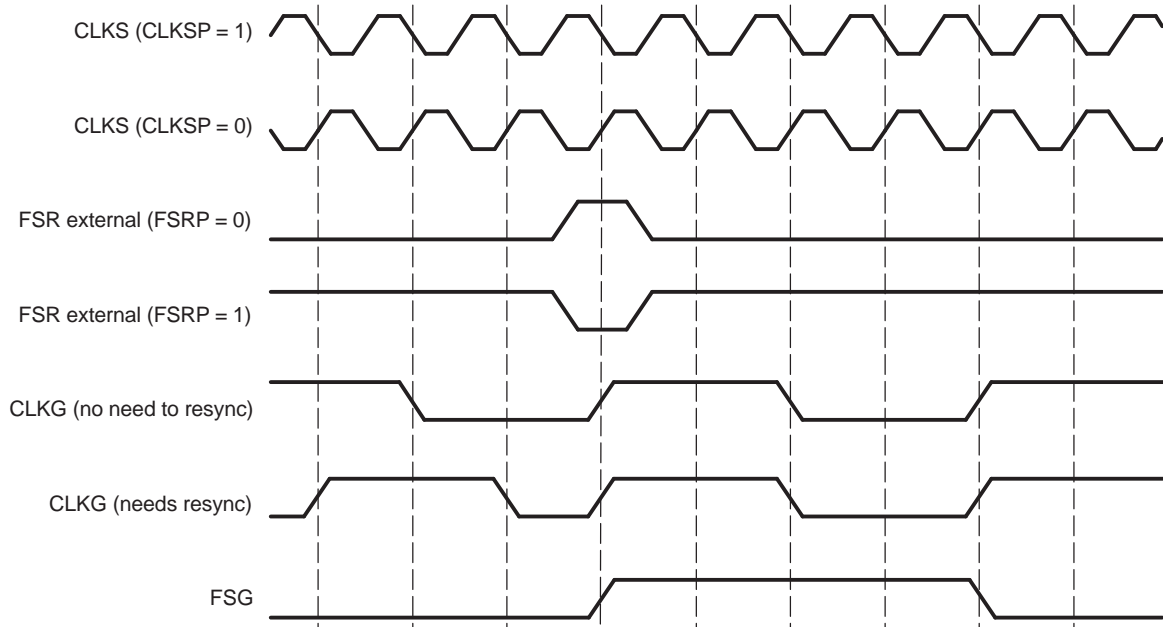
Figure 21-6 and Figure 21-7 show this operation with various polarities of CLKS and FSR. These figures assume that FWID is 0, for a FSG = 1 CLKG wide.

These figures show what happens to CLKG when it is initially in sync and GSYNC = 1, as well as when it is not in sync with the frame synchronization and GSYNC = 1.

**Figure 21-6. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1**



**Figure 21-7. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3**



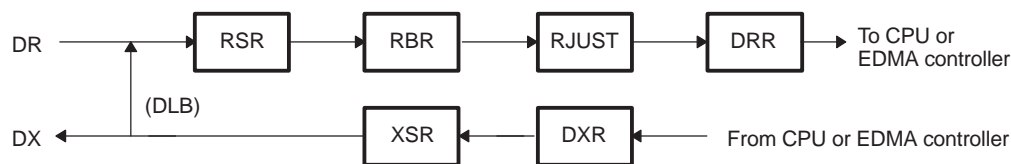
When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that the following conditions are met:

- FSX is programmed to be driven by the sample rate generator frame sync, FSG (FSGM = 1 in SRGR and FSXM = 1 in PCR).
- The sample-rate generator clock should drive the transmit and receive bit clock (CLK(R/X)M = 1 in SPCR). Therefore, the CLK(R/X) pin should not be driven by any other source.

### 21.2.5.3.5 Digital Loopback Mode: DLB

The DLB mode cannot be used when the McBSP is in clock stop mode (CLKSTP = 2h or 3h in the serial port control register (SPCR)). Setting DLB = 1 in SPCR enables digital loopback mode. In DLB mode, DR, FSR, and CLKR are internally connected through multiplexers to DX, FSX, and CLKX, respectively, as shown in Figure 21-2 and Figure 21-8. DLB mode allows testing of serial port code without using the external interface of the McBSP. CLKX and FSX must be enabled as outputs (CLKXM = FSXM = 1) in DLB mode.

**Figure 21-8. Digital Loopback Mode**



### 21.2.5.3.6 Receive Clock Selection: DLB, CLKRM

Table 21-3 shows how the digital loopback bit (DLB) in the serial port control register (SPCR) and the CLKRM bit in the pin control register (PCR) select the receiver clock. In digital loopback mode (DLB = 1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

**Table 21-3. Receive Clock Selection**

DLB Bit in SPCR	CLKRM Bit in PCR	Source of Receive Clock	CLKR Function
0	0	CLKR acts as an input driven by the external clock and inverted as determined by CLKRP before being used.	Input.
0	1	The sample rate generator clock (CLKG) drives CLKR.	Output. CLKG inverted as determined by CLKRP before being driven out on CLKR.
1	0	CLKX_int drives the receive clock CLKR_int as selected and is inverted.	High impedance.
1	1	CLKX_int drives CLKR_int as selected and is inverted.	Output. CLKR (same as CLKX) is inverted as determined by CLKRP before being driven out.

### 21.2.5.3.7 Transmit Clock Selection: CLKXM

Table 21-4 shows how the CLKXM bit in the pin control register (PCR) selects the transmit clock and whether the CLKX pin is an input or output.

**Table 21-4. Transmit Clock Selection**

CLKXM Bit in PCR	Source of Transmit Clock	CLKX Function
0	The external clock drives the CLKX input pin. CLKX is inverted as determined by CLKXP before being used.	Input.
1	The sample rate generator clock (CLKG) drives the transmit clock.	Output. CLKG is inverted as determined by CLKXP before being driven out on CLKX.

### 21.2.5.3.8 Stopping Clocks

Two methods can be used to stop serial clocks between data transfers:

- The SPI™ CLKSTP mode where clocks are stopped between single-element transfers. This mode is not supported on this device.
- The clocks are inputs to the McBSP (CLKXM or CLKRM = 0 in the pin control register (PCR)) and the McBSP operates in non-SPI mode (the clocks can be stopped between data transfers). If the external device stops the serial clock between data transfers, the McBSP interprets it as a slowed-down serial clock. Ensure that there are no glitches on the CLK(R/X) lines as the McBSP may interpret them as clock-edge transitions. Restarting the serial clock is equivalent to a normal clock transition after a slow CLK(R/X) cycle. Note that just as in normal operations, transmit under flow (XEMPTY) may occur if the DXR is not properly serviced at least three CLKX cycles before the next frame sync. Therefore, if the serial clock is stopped before DXR is properly serviced, the external device needs to restart the clock at least three CLKX cycles before the next frame sync to allow the DXR write to be properly synchronized. See Figure 21-29 for a graphical explanation on when DXR needs to be written to avoid underflow.

### 21.2.5.4 Frame Sync Generation

Data frame synchronization is independently programmable for the receiver and the transmitter for all data delay values. When the FRST bit in the serial port control register (SPCR) is set to 1 the frame generation logic is activated to generate frame sync signals, provided that FSGM = 1 in the sample rate generator register (SRGR). The frame sync programming options are:

- A frame pulse with a programmable period between sync pulses and a programmable active width specified in SRGR.
- The transmitter can trigger its own frame sync signal that is generated by a DXR-to-XSR copy. This causes a frame sync to occur on every DXR-to-XSR copy. The data delays can be programmed as required. However, maximum packet frequency cannot be achieved in this method for data delays of 1 and 2.
- Both the receiver and transmitter can independently select an external frame synchronization on the FSR and FSX pins, respectively.

#### 21.2.5.4.1 Frame Period (FPER) and Frame Width (FWID)

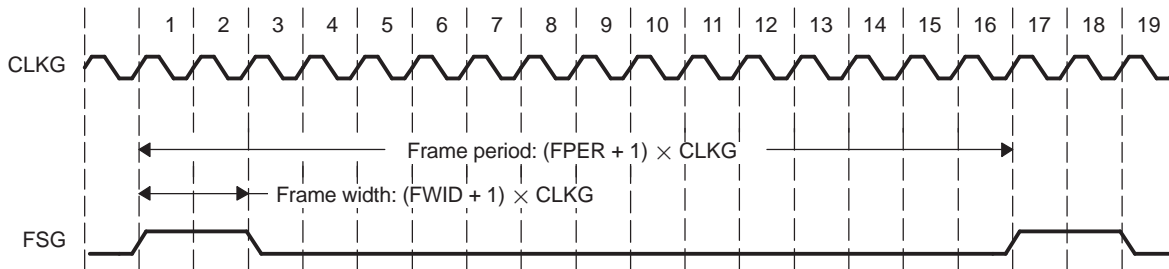
The FPER bits in the sample rate generator register (SRGR) are a 12-bit down-counter that can count down the generated data clocks from 4095 to 0. FPER controls the period of active frame sync pulses. The FWID bits in SRGR are an 8-bit down-counter. FWID controls the active width of the frame sync pulse.



When the sample rate generator comes out of reset, FSG is in an inactive (low) state. After this, when  $FRST = 1$  in SPCR and  $FSGM = 1$  in SRGR, frame sync signals are generated. The frame width value ( $FWID + 1$ ) is counted down on every CLKG cycle until it reaches 0 when FSG goes low. Thus, the value of  $FWID + 1$  determines an active frame pulse width ranging from 1 to 256 data bit clocks. At the same time, the frame period value ( $FPER + 1$ ) is also counting down, and when this value reaches 0, FSG goes high again, indicating a new frame is beginning. Thus, the value of  $FPER + 1$  determines a frame length from 1 to 4096 data bits. When  $GSYNC = 1$  in SRGR, the value of  $FPER$  does not matter. [Figure 21-9](#) shows a frame of 16 CLKG periods ( $FPER = 15$  or 0000 1111b).

It is recommended that  $FWID$  be programmed to a value less than  $(R/X)WDLEN1/2$ .

**Figure 21-9. Programmable Frame Period and Width**



**21.2.5.4.2 Receive Frame Synchronization Selection: DLB and FSRM**

[Table 21-5](#) shows how you can select various sources to provide the receive frame synchronization signal. Note that in digital loopback mode ( $DLB = 1$  in the serial port control register (SPCR)), the transmit frame sync signal is used as the receive frame sync signal and that DR is internally connected to DX.

**NOTE:** FSR\_int and FSX\_int are shown in [Figure 21-2](#).

**Table 21-5. Receive Frame Synchronization Selection**

DLB Bit in SPCR	FSRM Bit in PCR	GSYNC Bit in SRGR	Source of Receive Frame Synchronization	FSR Pin Function
0	0	X	External frame sync signal drives the FSR input pin, whose signal is then inverted as determined by FSRP before being used as FSR_int.	Input.
0	1	0	Sample rate generator frame sync signal (FSG) drives FSR_int, $FRST = 1$ .	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Sample rate generator frame sync signal (FSG) drives FSR_int, $FRST = 1$ .	Input. The external frame sync input on FSR is used to synchronize CLKG and generate FSG.
1	0	0	FSX_int drives FSR_int. FSX is selected as shown in <a href="#">Table 21-6</a> .	High impedance.
1	X	1	FSX_int drives FSR_int and is selected as shown in <a href="#">Table 21-6</a> .	Input. External FSR is not used for frame synchronization but is used to synchronize CLKG and generate FSG since $GSYNC = 1$ .
1	1	0	FSX_int drives FSR_int and is selected as shown in <a href="#">Table 21-6</a> .	Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out.

### 21.2.5.4.3 Transmit Frame Synchronization Selection: FSXM and FSGM

Table 21-6 shows how you can select the source of the transmit frame synchronization signal. The three choices are:

- External frame sync input
- The sample rate generator frame sync signal, FSG
- A signal that indicates a DXR-to-XSR copy has been made

---

**NOTE:** FSR\_int and FSX\_int are shown in [Figure 21-2](#).

---

**Table 21-6. Transmit Frame Synchronization Selection**

FSXM Bit in PCR	FSGM Bit in SRGR	Source of Transmit Frame Synchronization	FSX Pin Function
0	X	External frame sync input on the FSX pin. This is inverted by FSXP before being used as FSX_int.	Input.
1	1	Sample rate generator frame sync signal (FSG) drives FSX_int. FRST = 1.	Output. FSG is inverted by FSXP before being driven out on FSX.
1	0	A DXR-to-XSR copy activates transmit frame sync signal.	Output. 1-bit-clock-wide signal inverted as determined by FSXP before being driven out on FSX.

### 21.2.5.4.4 Frame Detection

To facilitate detection of frame synchronization, the receive and transmit CPU interrupts (RINT and XINT) can be programmed to detect frame synchronization by setting the RINTM and XINTM bits in the serial port control register (SPCR) to 10b. The associated portion (receiver/transmitter) of the McBSP must be out of reset.

## 21.2.5.5 Data and Frames

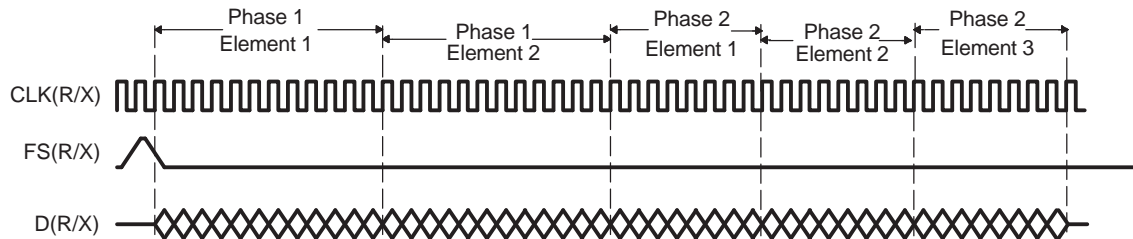
### 21.2.5.5.1 Frame Synchronization Phases

Frame synchronization indicates the beginning of a transfer on the McBSP. The data stream following frame synchronization can have up to two phases, phase 1 and phase 2. The number of phases can be selected by the phase bit, (R/X)PHASE, in RCR and XCR. The number of elements per frame and bits per element can be independently selected for each phase via (R/X)FRLN1/2 and (R/X)WDLEN1/2, respectively. [Figure 21-10](#) shows a frame in which the first phase consists of two elements of 12 bits, each followed by a second phase of three elements of 8 bits each. The entire bit stream in the frame is contiguous; no gaps exist either between elements or phases. [Table 21-7](#) shows the fields in the receive/transmit control registers (RCR/XCR) that control the frame length and element length for each phase for both the receiver and the transmitter. The maximum number of elements per frame is 128 for a single-phase frame and 256 elements in a dual-phase frame. The number of bits per element can be 8, 12, 16, 20, 24, or 32.

---

**NOTE:** For a dual-phase frame with internally generated frame synchronization, the maximum number of elements per phase depends on the word length. This is because the frame period, FPER, is only 12-bits wide and, therefore, provides 4096 bits per frame. Hence, the maximum number of 256 elements per dual-phase frame applies only when the WDLEN is 16 bits. However, any combination of element numbers and element size (defined by the FRLN and WDLEN bits, respectively) is valid as long as their product is less than or equal to 4096 bits. This limitation does not apply for dual-phase with external frame sync.

---

**Figure 21-10. Dual-Phase Frame Example**

**Table 21-7. RCR/XCR Fields Controlling Elements per Frame and Bits per Element**

Serial Port	Frame Phase	RCR/XCR Field Control	
		Elements per Frame	Bits per Element
Receive	1	RFLEN1	RWDLEN1
Receive	2	RFLEN2	RWDLEN2
Transmit	1	XFLEN1	XWDLEN1
Transmit	2	XFLEN2	XWDLEN2

### 21.2.5.5.2 Frame Length: RFLEN1/2 and XFLEN1/2

Frame length specifies the maximum number of serial elements or logical time slots or channels that are available for transfer per frame synchronization signal. In multichannel selection mode, the frame length value is independent of, and perhaps different from, the actual number of channels that the device is programmed to receive or transmit per frame via the MCR, RCEREN, and XCEREN registers. See [Section 21.2.9](#) for details on multichannel selection mode operation. The 7-bit (R/X)FLEN1/2 bits in (R/X)CR support up to 128 elements per phase in a frame, as shown in [Table 21-8](#). (R/X)PHASE = 0 selects a single-phase data frame, and (R/X)PHASE = 1 selects a dual-phase frame for the data stream. For a single-phase frame, the value of (R/X)FLEN2 does not matter. Program the frame length fields with (w minus 1), where w represents the number of elements per frame. For [Figure 21-10](#), (R/X)FLEN1 = 1 or 000 0001b and (R/X)FLEN2 = 2 or 000 0010b.

**Table 21-8. Receive/Transmit Frame Length Configuration**

(R/X)PHASE	(R/X)FLEN1	(R/X)FLEN2	Frame Length
0	$0 \leq n \leq 127$	x	Single-phase frame; (n + 1) elements per frame
1	$0 \leq n \leq 127$	$0 \leq m \leq 127$	Dual-phase frame; (n + 1) plus (m + 1) elements per frame

### 21.2.5.5.3 Element Length: RWDLEN1/2 and XWDLEN1/2

The (R/X)WDLEN1/2 fields in the receive/transmit control register (RCR and XCR) determine the element length in bits per element for the receiver and the transmitter for each phase of the frame, as indicated in [Table 21-7](#). [Table 21-9](#) shows how the value of these fields selects particular element lengths in bits. For the example in [Figure 21-10](#), (R/X)WDLEN1 = 001b and (R/X)WDLEN2 = 000b. If (R/X)PHASE = 0, indicating a single-phase frame, then (R/X)WDLEN2 is not used by the McBSP and its value does not matter.

**Table 21-9. Receive/Transmit Element Length Configuration**

(R/X)WDLEN1/2	Element Length (Bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	Reserved
111	Reserved

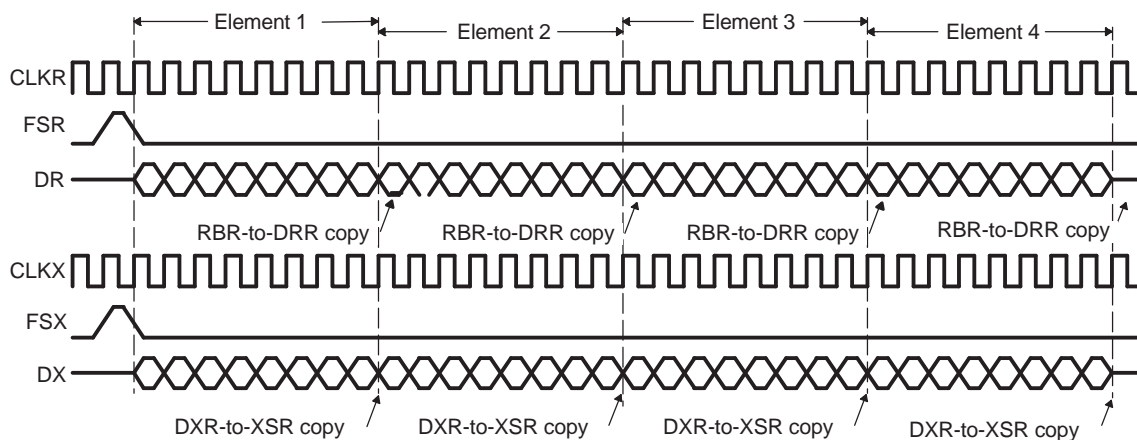
**21.2.5.5.4 Data Packing Using Frame Length and Element Length**

The frame length and element length can be manipulated to effectively pack data. For example, consider a situation in which four 8-bit elements are transferred in a single-phase frame, as shown in [Figure 21-11](#). In this case:

- (R/X)PHASE = 0, indicating a single-phase frame
- (R/X)FRLLEN1 = 000 0011b, indicating a 4-element frame
- (R/X)WDLEN1 = 000b, indicating 8-bit elements

In this example, four 8-bit data elements are transferred to and from the McBSP by the CPU or the EDMA controller. Four reads of DRR and four writes of DXR are necessary for each frame.

**Figure 21-11. Single-Phase Frame of Four 8-Bit Elements**

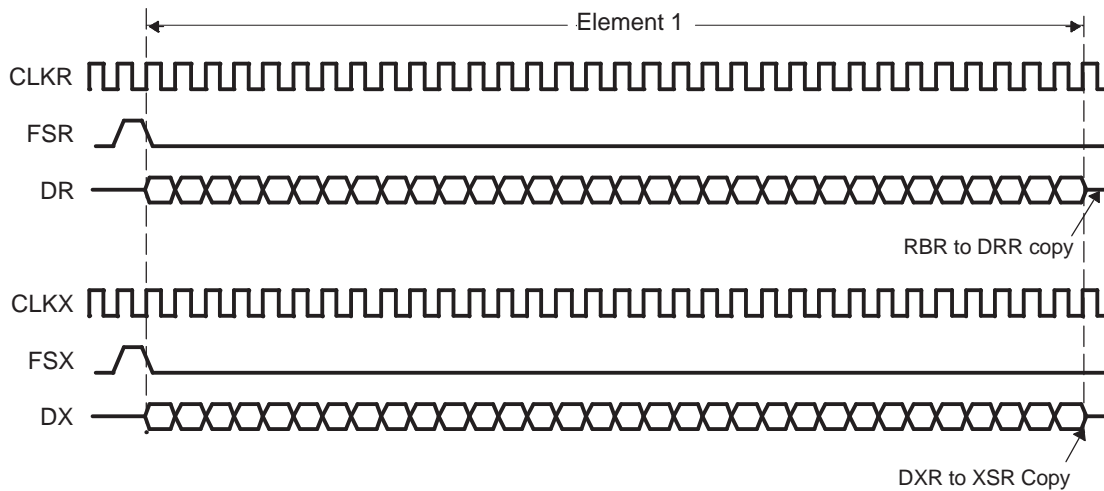


The example in [Figure 21-11](#) can also be viewed as a data stream of a single-phase frame of one 32-bit data element, as shown in [Figure 21-12](#). In this case:

- (R/X)PHASE = 0, indicating a single phase frame
- (R/X)FRLLEN1 = 0, indicating a 1-element frame
- (R/X)WDLEN1 = 101b, indicating 32-bit elements

In this example, one 32-bit data element is transferred to and from the McBSP by the CPU or the EDMA controller. Thus, one read of DRR and one write of DXR is necessary for each frame. As a result, the number of transfers is one-fourth that of the previous example ([Figure 21-11](#)). This manipulation reduces the percentage of bus time required for serial port data movement.

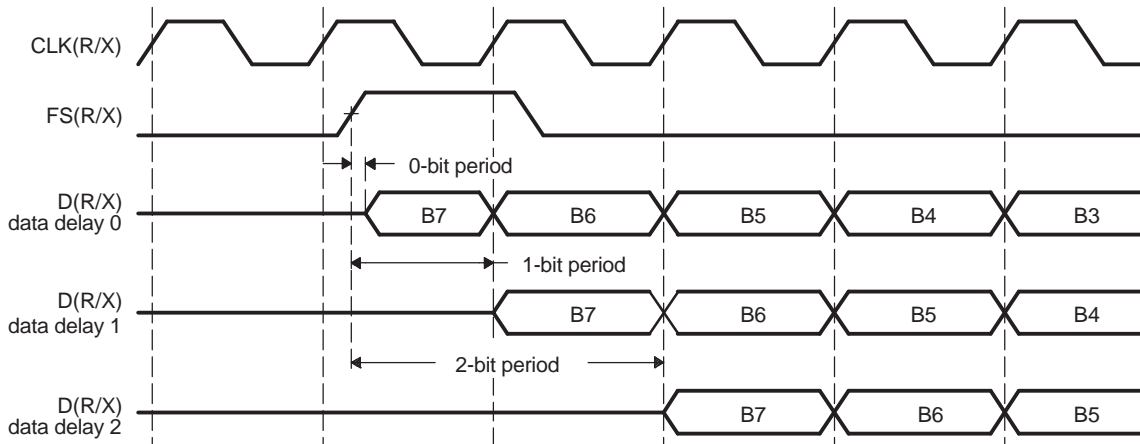
Figure 21-12. Single-Phase Frame of One 32-Bit Element



21.2.5.5.5 Data Delay: RDATDLY and XDATDLY

The start of a frame is defined by the first clock cycle in which frame synchronization is active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay. RDATDLY (in RCR) and XDATDLY (in XCR) specify the data delay for reception and transmission, respectively. The range of programmable data delay is zero to two bit clocks ((R/X)DATDLY = 00b to 10b), as shown in Figure 21-13. Typically, a 1-bit delay is selected because data often follows a 1-cycle active frame sync pulse.

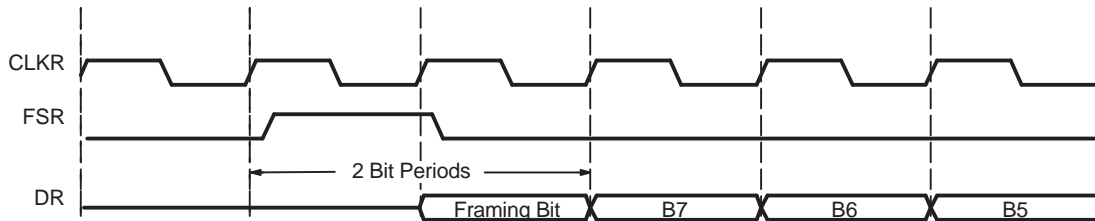
Figure 21-13. Data Delay



Normally, a frame sync pulse is detected or sampled with respect to an edge of serial clock CLK(R/X). Thus, on a subsequent cycle (depending on data delay value), data can be received or transmitted. However, in the case of a 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle. For reception, this problem is solved by receive data being sampled on the first falling edge of CLKR when an active (high) FSR is detected. However, data transmission must begin on the rising edge of CLKX that generated the frame synchronization. Therefore, the first data bit is assumed to be in the XSR and DX. The transmitter then asynchronously detects the frame synchronization, FSX goes active, and it immediately starts driving the first bit to be transmitted on the DX pin.

Another common operation uses a data delay of 2. This configuration allows the serial port to interface to different types of T1 framing devices in which the data stream is preceded by a framing bit. During the reception of such a stream with a data delay of two bits, the framing bit appears after a 1-bit delay and data appears after a 2-bit delay). The serial port essentially discards the framing bit from the data stream, as shown in Figure 21-14. In transmission, by delaying the first transfer bit, the serial port essentially inserts a blank period (high-impedance period) in place of the framing bit. Here, it is expected that the framing device inserts its own framing bit or that the framing bit is generated by another device. Alternatively, you may pull up or pull down DX to achieve the desired value.

**Figure 21-14. 2-Bit Data Delay Used to Discard Framing Bit**



**21.2.5.5.6 Receive Data Justification and Sign Extension: RJUST**

The RJUST bit in the serial port control register (SPCR) selects whether data in RBR is right- or left-justified (with respect to the MSB) in the DRR. If right justification is selected, RJUST further selects whether the data is sign-extended or zero-filled. Table 21-10 and Table 21-11 summarize the effect that various values of RJUST have on example receive data.

**Table 21-10. Effect of RJUST Bit Values With 12-Bit Example Data ABCCh**

RJUST Bit in SPCR	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	0000 0ABCCh
01	Right	Sign-extend MSBs	FFFF FABCh
10	Left	Zero-fill LSBs	ABC0 0000h
11	Reserved	Reserved	Reserved

**Table 21-11. Effect of RJUST Bit Values With 20-Bit Example Data ABCDEh**

RJUST Bit in SPCR	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	000A BCDEh
01	Right	Sign-extend MSBs	FFFA BCDEh
10	Left	Zero-fill LSBs	ABCD E000h
11	Reserved	Reserved	Reserved

### 21.2.5.5.7 32-Bit Bit Reversal: RWDREVRS, XWDREVRS

Normally all transfers are sent and received with the MSB first; however, you can reverse the receive/transmit bit ordering of a 32-bit element (LSB first) using the 32-bit reversal feature of the McBSP by setting all of the following:

- (R/X)WDREVRS = 1 in the receive/transmit control register (RCR/XCR).
- (R/X)COMPAND = 01b in RCR/XCR.
- (R/X)WDLEN(1/2) = 101b in RCR/XCR to indicate 32-bit elements.

When you set the register fields as above, the bit ordering of the 32-bit element is reversed before being received by or sent from the serial port. If the (R/W)WDREVRS and (R/X)COMPAND fields are set as above, but the element size is not set to 32-bit, operation is undefined.

### 21.2.6 McBSP Buffer FIFO (BFIFO)

The McBSP Buffer FIFO (BFIFO) provides additional data buffering for the McBSP. The time it takes the host CPU or DMA controller to respond to DMA requests from the McBSP may vary; the additional buffering provided by the BFIFO allows greater tolerance to such variations. For convenience, the BFIFO is treated here as a block between the McBSP and the host/DMA controller (see [Figure 21-1](#)). Details on configuration of the BFIFO are provided in [Section 21.2.7.6](#).

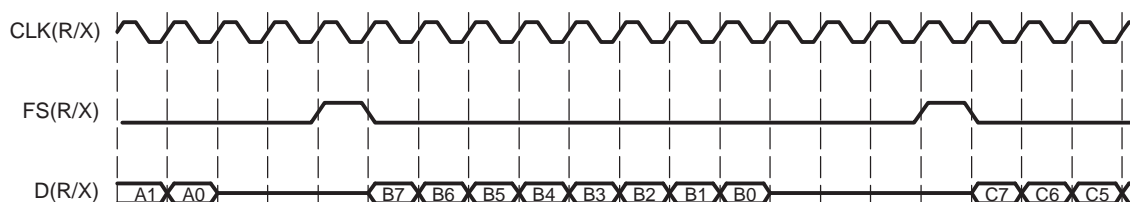
### 21.2.7 McBSP Standard Operation

During a serial transfer, there are typically periods of serial port inactivity between packets or transfers. The receive and transmit frame synchronization pulse occurs for every serial transfer. When the McBSP is not in the reset state and has been configured for the desired operation, a serial transfer can be initiated by programming (R/X)PHASE = 0 for a single-phase frame with the required number of elements programmed in (R/X)FRLLEN1. The number of elements can range from 1 to 128 ((R/X)FRLLEN1 = 00h to 7Fh). The required serial element length is set in the (R/X)WDLEN1 field in the (R/X)CR. If a dual-phase frame is required for the transfer, RPHASE = 1 and each (R/X)FRLLEN1/2 can be set to any value between 0h and 7Fh.

[Figure 21-15](#) shows a single-phase data frame of one 8-bit element. Since the transfer is configured for a 1-bit data delay, the data on the DX and DR pins are available one bit clock after FS(R/X) goes active. This figure, as well as all others in this section, use the following assumptions:

- (R/X)PHASE = 0, specifying a single-phase frame
- (R/X)FRLLEN1 = 0b, specifying one element per frame
- (R/X)WDLEN1 = 000b, specifying eight bits per element
- (R/X)FRLLEN2 = (R/X)WDLEN2 = Value is ignored
- CLK(R/X)P = 0, specifying that the receive data is clocked on the falling edge and that transmit data is clocked on the rising edge
- FS(R/X)P = 0, specifying that active (high) frame sync signals are used
- (R/X)DATDLY = 01b, specifying a 1-bit data delay

**Figure 21-15. McBSP Standard Operation**

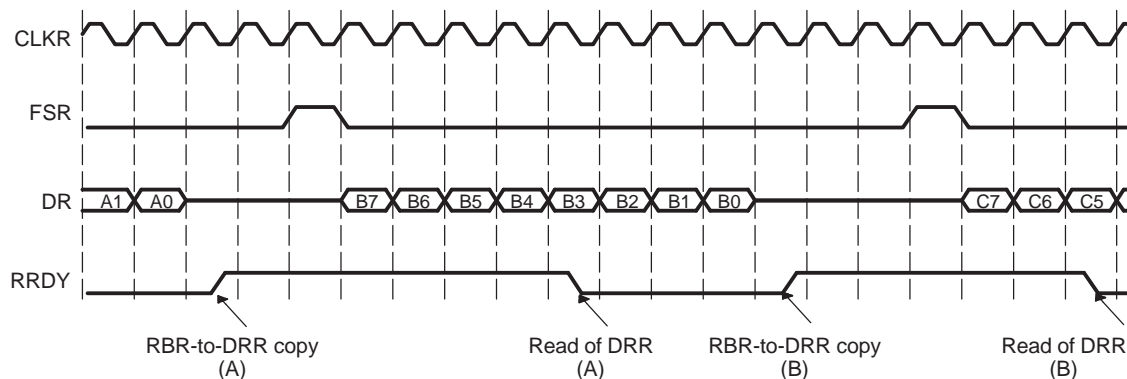




### 21.2.7.1 Receive Operation

Figure 21-16 shows serial reception. Once the receive frame synchronization signal (FSR) transitions to its active state, it is detected on the first falling edge of the receivers CLKR. The data on the DR pin is then shifted into the receive shift register (RSR) after the appropriate data delay as set by the RDATDLY bit in the receive control register (RCR). The contents of RSR is copied to RBR at the end of every element on the rising edge of the clock, provided RBR is not full with the previous data. Then, an RBR-to-DRR copy activates the RRDY status bit in the serial port control register (SPCR) to 1 on the following falling edge of CLKR. This indicates that the receive data register (DRR) is ready with the data to be read by the CPU or the EDMA controller. RRDY is deactivated when the DRR is read by the CPU or the EDMA controller. See also Section 21.2.13.1.2 and Section 21.2.14.1.

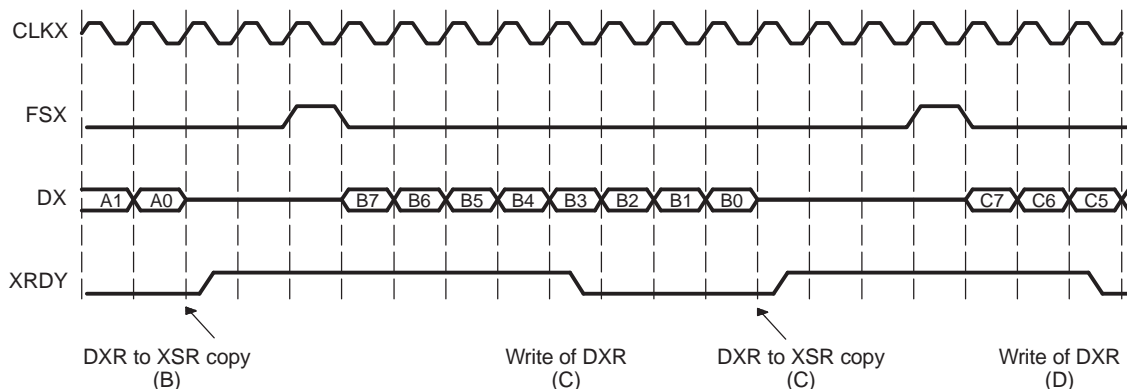
Figure 21-16. Receive Operation



### 21.2.7.2 Transmit Operation

Once transmit frame synchronization occurs, the value in the transmit shift register (XSR) is shifted out and driven on the DX pin after the appropriate data delay as set by the XDATDLY bit in the transmit control register (XCR). The XRDY bit in the serial port control register (SPCR) is activated after every DXR-to-XSR copy on the following falling edge of CLKX, indicating that the data transmit register (DXR) can be written with the next data to be transmitted. XRDY is deactivated when the DXR is written by the CPU or the EDMA controller. Figure 21-17 illustrates serial transmission. See Section 21.2.7.5.4 for information on transmit operation when the transmitter is pulled out of reset (XRST = 1). See also Section 21.2.13.1.3 and Section 21.2.14.2.

Figure 21-17. Transmit Operation





### 21.2.7.3 Maximum Frame Frequency

The frame frequency is determined by the following equation, which calculates the period between frame synchronization signals:

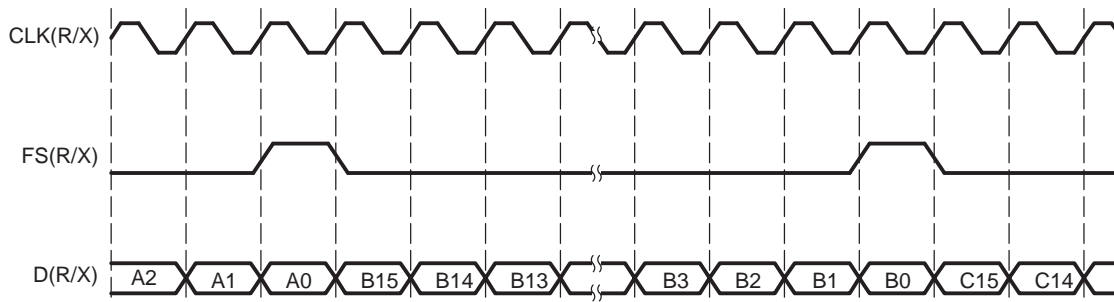
$$\text{Frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bit clocks between frame sync signals}}$$

The frame frequency may be increased by decreasing the time between frame synchronization signals in bit clocks (which is limited only by the number of bits per frame). As the frame transmit frequency is increased, the inactivity period between the data frames for adjacent transfers decreases to 0. The minimum time between frame synchronization pulses is the number of bits transferred per frame. This time also defines the maximum frame frequency, which is calculated by the following equation:

$$\text{Maximum frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bits per frame}}$$

Figure 21-18 shows the McBSP operating at maximum frame frequency. The data bits in consecutive frames are transmitted continuously with no inactivity between bits. If there is a 1-bit data delay, as shown, the frame synchronization pulse overlaps the last bit transmitted in the previous frame.

**Figure 21-18. Maximum Frame Frequency for Transmit and Receive**



**NOTE:** For (R/X)DATDLY = 0, the first bit of data transmitted is asynchronous to CLKX, as shown in Figure 21-13.

Maximum frame frequency may not be possible when the word length is 8-bits, depending on the clock divide value CLKGDV. The CPU or EDMA may not be able to service serial port requests that occur as frequently as every 8-bit clocks. This situation can be resolved by allowing additional space between words or choosing a slower bit clock (larger CLKGDV value).

### 21.2.7.4 Frame Synchronization Ignore

The McBSP can be configured to ignore transmit and receive frame synchronization pulses. The (R/X)FIG bit in (R/X)CR can be cleared to 0 to recognize frame sync pulses, or it can be set to 1 to ignore frame sync pulses. In this way, you can use (R/X)FIG either to pack data, if operating at maximum frame frequency, or to ignore unexpected frame sync pulses.

#### 21.2.7.4.1 Frame Sync Ignore and Unexpected Frame Sync Pulses

RFIG and XFIG are used to ignore unexpected internal or external frame sync pulses. Any frame sync pulse is considered unexpected if it occurs one or more bit clocks earlier than the programmed data delay from the end of the previous frame specified by ((R/X)DATDLY). Setting the frame ignore bits to 1 causes the serial port to ignore these unexpected frame sync signals.

In reception, if not ignored (RFIG = 0), an unexpected FSR pulse discards the contents of RSR in favor of the incoming data. Therefore, if RFIG = 0, an unexpected frame synchronization pulse aborts the current data transfer, sets RSYNCERR in SPCR to 1, and begins the reception of a new data element. When RFIG = 1, the unexpected frame sync pulses are ignored.

In transmission, if not ignored (XFIG = 0), an unexpected FSX pulse aborts the ongoing transmission, sets the XSYNCERR bit in SPCR to 1, and reinitiates transmission of the current element that was aborted. When XFIG = 1, unexpected frame sync signals are ignored.

Figure 21-19 shows that element B is interrupted by an unexpected frame sync pulse when (R/X)FIG = 0. The reception of B is aborted (B is lost), and a new data element (C) is received after the appropriate data delay. This condition causes a receive synchronization error and thus sets the RSYNCERR bit. However, for transmission, the transmission of B is aborted and the same data (B) is retransmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR bit. Synchronization errors are discussed in Section 21.2.7.5.2 and Section 21.2.7.5.5.

Figure 21-19. Unexpected Frame Synchronization With (R/X)FIG = 0

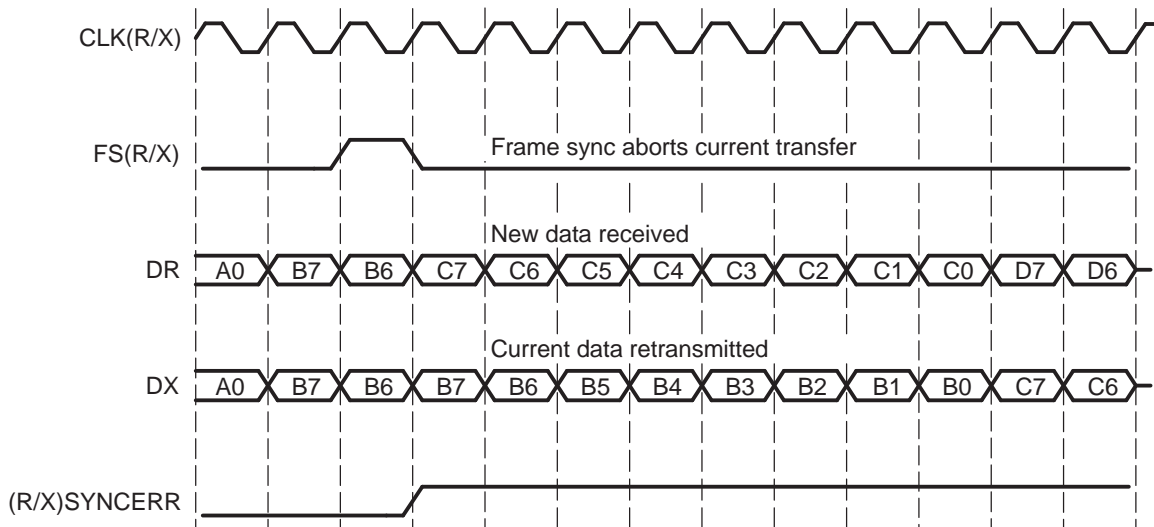
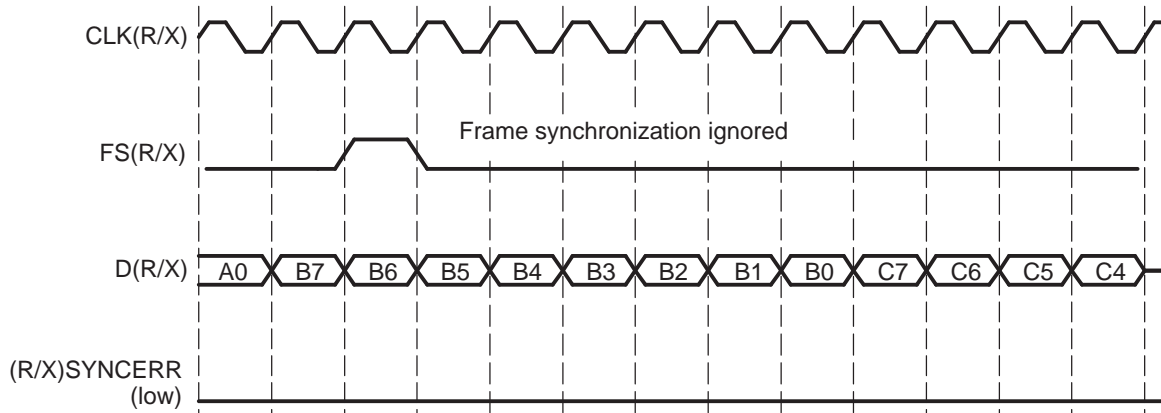


Figure 21-20 shows McBSP operation when unexpected internal or external frame synchronization signals are ignored by setting (R/X)FIG = 1. Here, the transfer of element B is not affected by an unexpected frame synchronization.

**Figure 21-20. Unexpected Frame Synchronization With (R/X)FIG = 1**



**21.2.7.4.2 Data Packing using Frame Sync Ignore Bits**

Section 21.2.5.5.4 describes one method of changing the element length and frame length to simulate 32-bit serial element transfers, thus requiring much less bus bandwidth than four 8-bit transfers require. This example works when there are multiple elements per frame.

Now consider the case of the McBSP operating at maximum packet frequency, as shown in Figure 21-21. Here, each frame has only a single 8-bit element. This stream takes one read transfer and one write transfer for each 8-bit element. Figure 21-22 shows the McBSP configured to treat this stream as a continuous stream of 32-bit elements. In this example, (R/X)FIG is set to 1 to ignore unexpected subsequent frames. Only one read transfer and one write transfer is needed every 32 bits. This configuration effectively reduces the required bus bandwidth to one-fourth of the bandwidth needed to transfer four 8-bit blocks.

**Figure 21-21. Maximum Frame Frequency Operation With 8-Bit Data**

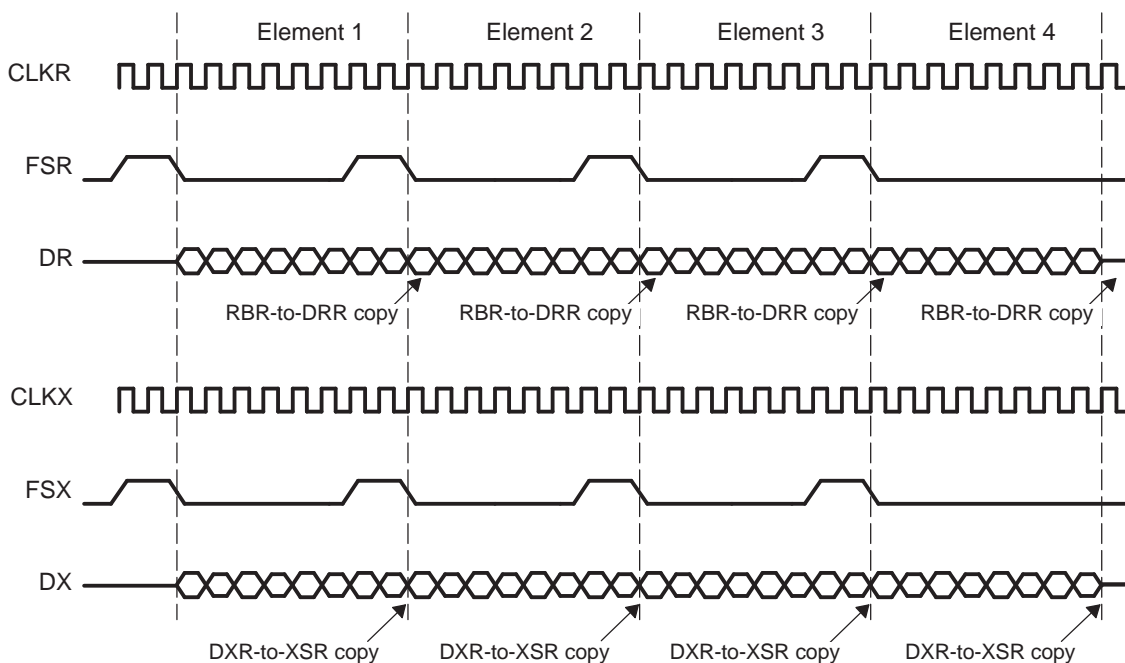
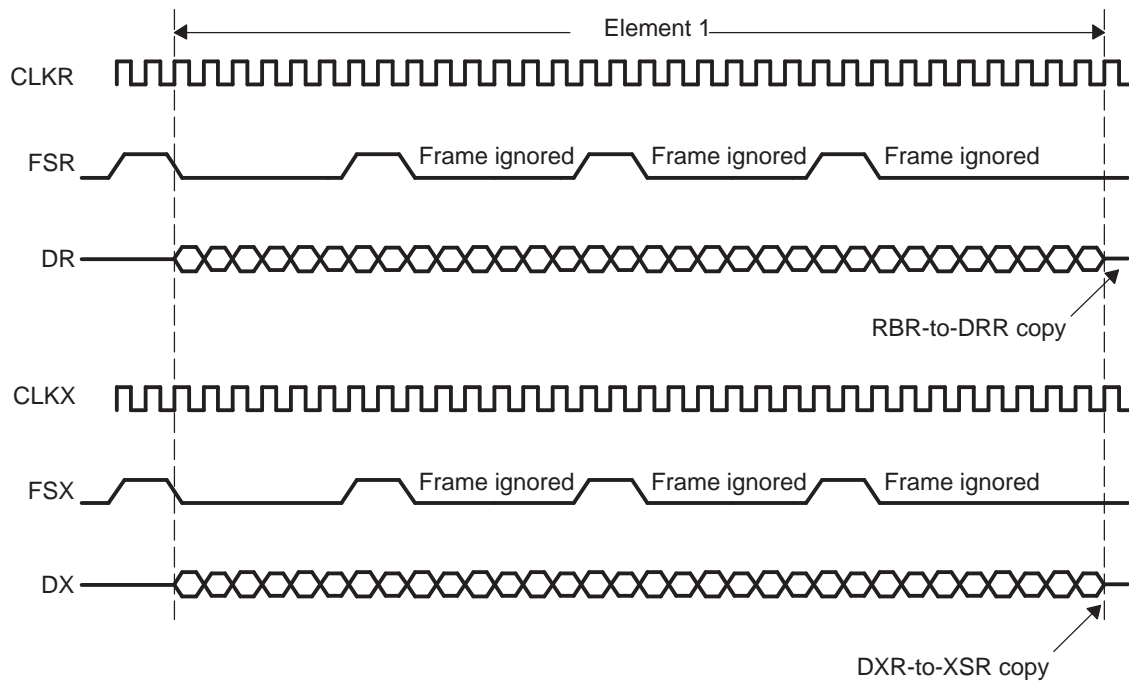


Figure 21-22. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1



### 21.2.7.5 Serial Port Exception Conditions

There are five serial port events that can constitute a system error:

- Receive overrun (RFULL = 1 in SPCR)
- Unexpected receive frame synchronization (RSYNCERR = 1 in SPCR)
- Transmit data overwrite
- Transmit empty (XEMPTY = 0 in SPCR)
- Unexpected transmit frame synchronization (XSYNCERR = 1 in SPCR)

#### 21.2.7.5.1 Receive Overrun: RFULL

RFULL = 1 in the serial port control register (SPCR) indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when the following conditions are met:

- DRR has not been read since the last RBR-to-DRR transfer.
- RBR is full and an RBR-to-DRR copy has not occurred.
- RSR is full and an RSR-to-RBR transfer has not occurred.

The data arriving on DR is continuously shifted into RSR (Figure 21-23). Once a complete element is shifted into RSR, an RSR-to-RBR transfer can occur only if an RBR-to-DRR copy is complete. Therefore, if DRR has not been read by the CPU or the EDMA controller since the last RBR-to-DRR transfer (RRDY = 1), an RBR-to-DRR copy does not take place until RRDY = 0. This prevents an RSR-to-RBR copy. New data arriving on the DR pin is shifted into RSR, and the previous contents of RSR are lost. After the receiver starts running from reset, a minimum of three elements must be received before RFULL can be set, because there was no last RBR-to-DRR transfer before the first element.

This data loss can be avoided if DRR is read no later than two and a half CLKR cycles before the end of the third element (data C) in RSR, as shown in Figure 21-24.

Either of the following events clears the RFULL bit to 0 and allows subsequent transfers to be read properly:

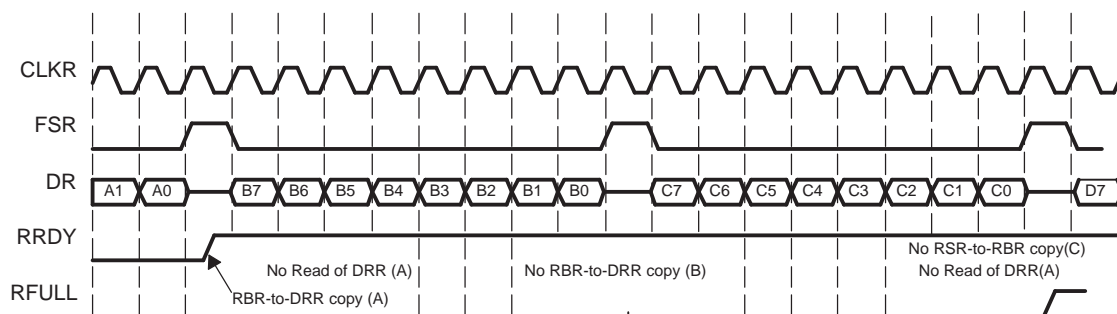
- Reading DRR
- Resetting the receiver (RRST = 0) or the device

Another frame synchronization is required to restart the receiver.

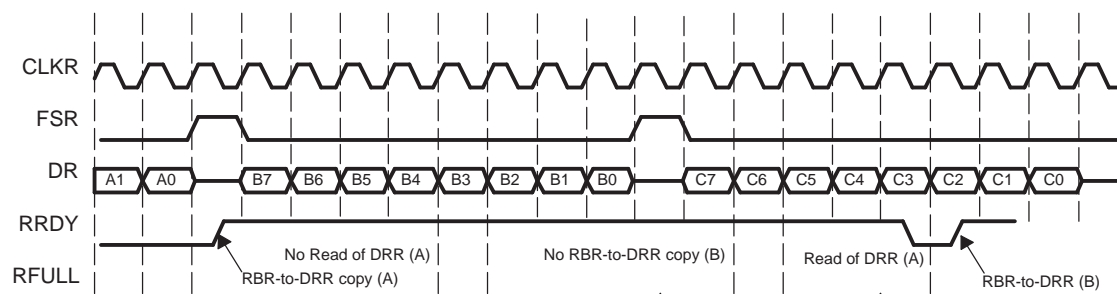
Figure 21-23 shows the receive overrun condition. Because element A is not read before the reception of element B is complete, B is not transferred to DRR yet. Another element, C, arrives and fills RSR. DRR is finally read, but not earlier than two and one half cycles before the end of element C. New data D overwrites the previous element C in RSR. If RFULL is still set after the DRR is read, the next element can overwrite D if DRR is not read in time.

Figure 21-24 shows the case in which RFULL is set but the overrun condition is averted by reading the contents of DRR at least two and a half cycles before the next element, C, is completely shifted into RSR. This ensures that a RBR-to-DRR copy of data B occurs before the next element is transferred from RSR to RBR.

**Figure 21-23. Serial Port Receive Overrun**



**Figure 21-24. Serial Port Receive Overrun Avoided**



**21.2.7.5.2 Unexpected Receive Frame Synchronization: RSYNCERR**

Figure 21-25 shows the decision tree that the receiver uses to handle all incoming frame synchronization pulses. The diagram assumes that the receiver has been activated (RRST = 1). Unexpected frame sync pulses can originate from an external source or from the internal sample rate generator. An unexpected frame sync pulse is defined as a sync pulse which occurs RDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSR pulses with RFIG = 1. This case is discussed in Section 21.2.7.4.1 and shown in Figure 21-20. Here, receive frame sync pulses are ignored and the reception continues.
- Case 2: Normal serial port reception. There are three reasons for a receive not to be in progress:
  - This FSR is the first after RRST = 1.
  - This FSR is the first after DRR has been read clearing an RFULL condition.
  - The serial port is in the inter-packet intervals. The programmed data delay (RDATDLY) for reception may start during these inter-packet intervals for the first bit of the next element to be received. Thus, at maximum frame frequency, frame synchronization can still be received RDATDLY bit clocks before the first bit of the associated element.

For this case, reception continues normally, because these are not unexpected frame sync pulses.

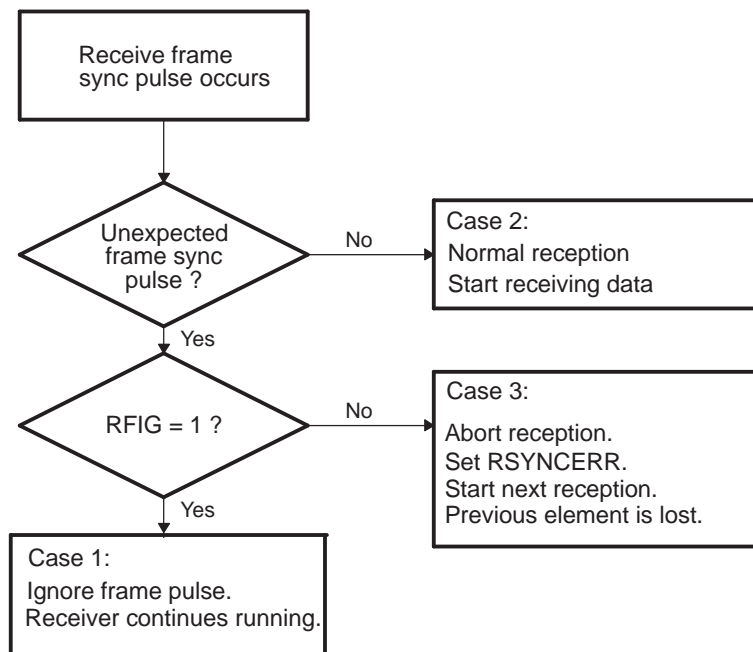
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (unexpected frame not ignored). This case was shown in Figure 21-19 for maximum packet frequency. Figure 21-26 shows this case during normal operation of the serial port with time intervals between packets. Unexpected frame sync pulses are detected when they occur the value in RDATDLY bit clocks before the last bit of the previous element is received on DR. In both cases, RSYNCERR in SPCR is set. RSYNCERR can be cleared only by receiver reset or by writing a 0 to this bit in SPCR. If RINTM = 11b in SPCR, RSYNCERR drives the receive interrupt (RINT) to the CPU.

---

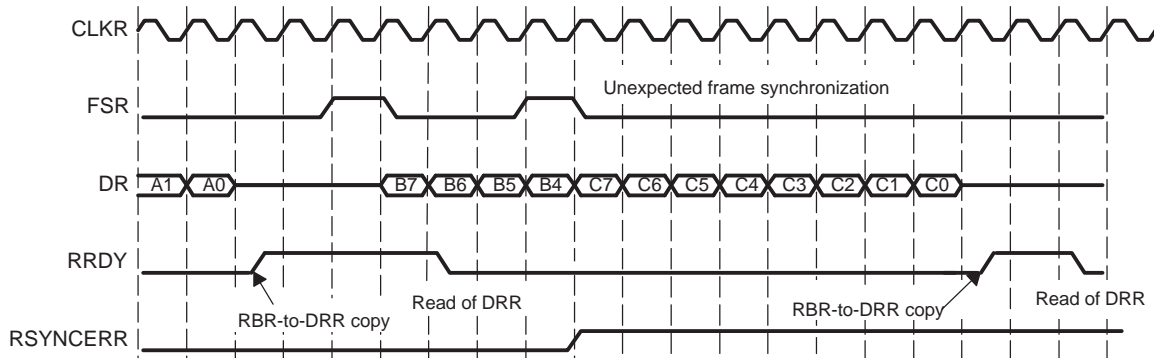
**NOTE:** Note that the RSYNCERR bit in SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

---

**Figure 21-25. Decision Tree Response to Receive Frame Synchronization Pulse**



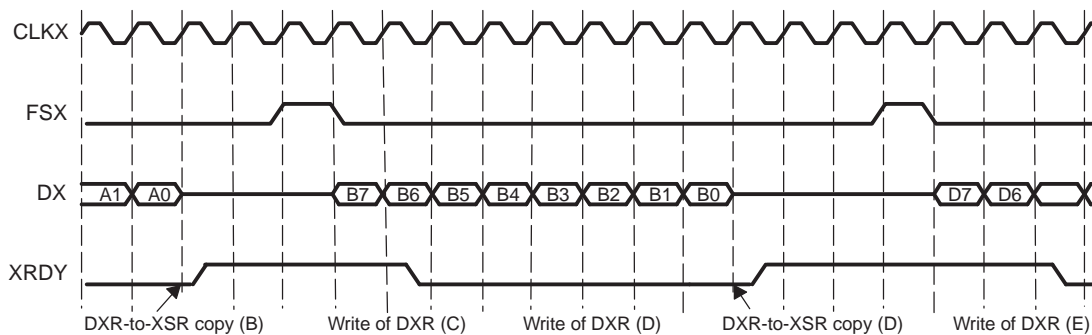
**Figure 21-26. Unexpected Receive Frame Synchronization Pulse**



**21.2.7.5.3 Transmit With Data Overwrite**

Figure 21-27 shows what happens if the data in DXR is overwritten before it is transmitted. Suppose you load the DXR with data C. A subsequent write to the DXR overwrites C with D before C is copied to the XSR. Thus, C is never transmitted on DX. The CPU can avoid overwriting data by polling XRDY before writing to DXR or by waiting for a programmed XINT to be triggered by XRDY (XINTM = 00b). The EDMA controller can avoid overwriting by synchronizing data writes with XEVT. See also Section 21.2.13.1.3.

**Figure 21-27. Transmit With Data Overwrite**



### 21.2.7.5.4 Transmit Empty: XEMPTY

XEMPTY indicates whether the transmitter has experienced underflow. Either of the following conditions causes XEMPTY to become active (XEMPTY = 0):

- During transmission, DXR has not been loaded since the last DXR-to-XSR copy, and all bits of the data element in XSR have been shifted out on DX.
- The transmitter is reset (XRST = 0 or the device is reset), and then restarted.

During an underflow condition, the transmitter continues to transmit the old data in DXR for every new frame sync signal FSX (generated by an external device, or by the internal sample rate generator) until a new element is loaded into DXR by the CPU or the EDMA controller. XEMPTY is deactivated (XEMPTY = 1) when this new element in DXR is transferred to XSR. In the case when the FSX is generated by a DXR-to-XSR copy (FSXM = 1 in PCR and FSGM = 0 in SRGR), the McBSP does not generate any new frame sync until new data is written to the DXR and a DXR-to-XSR copy occurs.

When the transmitter is taken out of reset (XRST = 1), it is in a transmit ready (XRDY = 1) and transmit empty (XEMPTY = 0) condition. If DXR is loaded by the CPU or the EDMA controller before FSX goes active, a valid DXR-to-XSR transfer occurs. This allows for the first element of the first frame to be valid even before the transmit frame sync pulse is generated or detected. Alternatively, if a transmit frame sync is detected before DXR is loaded, 0s are output on DX.

Figure 21-28 shows a transmit underflow condition. After B is transmitted, B is retransmitted on DX if you fail to reload the DXR before the subsequent frame synchronization. Figure 21-29 shows the case of writing to DXR just before a transmit underflow condition that would otherwise occur. After B is transmitted, C is written to DXR before the next transmit frame sync pulse occurs.

Figure 21-28. Transmit Empty

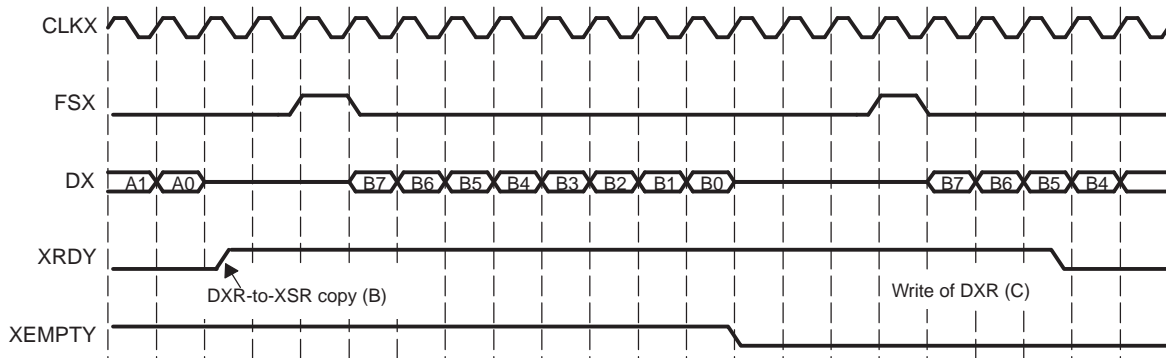
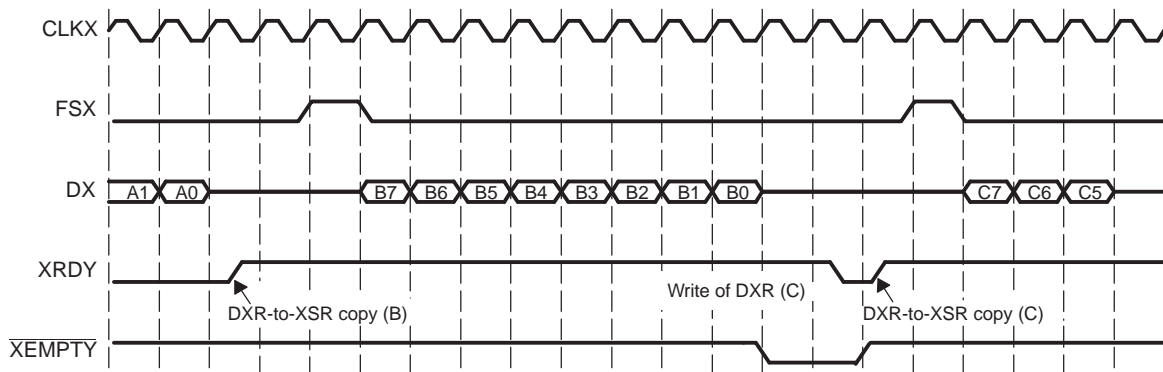


Figure 21-29. Transmit Empty Avoided





### 21.2.7.5.5 Unexpected Transmit Frame Synchronization: XSYNCERR

A transmit frame sync error (XSYNCERR) may occur the first time the transmitter is enabled (XRST = 1) after a device reset. To avoid this, after enabling the transmitter for the first time, the following procedure must be followed:

1. Wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
2. Disable the transmitter (XRST = 0). This clears any XSYNCERR.
3. Re-enable the transmitter (XRST = 1).

See also [Section 21.2.12](#) for details on initialization procedure.

[Figure 21-30](#) shows the decision tree that the transmitter uses to handle all incoming frame synchronization signals. The diagram assumes that the transmitter has been started (XRST = 1). An unexpected transmit frame sync pulse is defined as a sync pulse that occurs XDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSX pulses with XFIG = 1. This case is discussed in [Section 21.2.7.4.1](#) and shown in [Figure 21-20](#). In this case, unexpected FSX pulses are ignored, and the transmission continues.
- Case 2: FSX pulses with normal serial port transmission. This situation is discussed in [Section 21.2.7.2](#). There are two possible reasons for a transmit not to be in progress:
  - This FSX pulse is the first one to occur after XRST = 1.
  - The serial port is in the interpacket intervals. The programmed data delay (XDATDLY) may start during these interpacket intervals before the first bit of the next element is transmitted. Thus, if operating at maximum packet frequency, frame synchronization can still be received XDATDLY bit clocks before the first bit of the associated element.
- Case 3: Unexpected transmit frame synchronization with XFIG = 0. The case was shown in [Figure 21-19](#) for frame synchronization with XFIG = 0 at maximum packet frequency. [Figure 21-31](#) shows the case for normal operation of the serial port with interpacket intervals. In both cases, XSYNCERR in SPCR is set. XSYNCERR can be cleared only by transmitter reset or by writing a 0 to this bit in SPCR. If XINTM = 11b in SPCR, XSYNCERR drives the receive interrupt (XINT) to the CPU.

---

**NOTE:** The XSYNCERR bit in SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

---

Figure 21-30. Decision Tree Response to Transmit Frame Synchronization Pulse

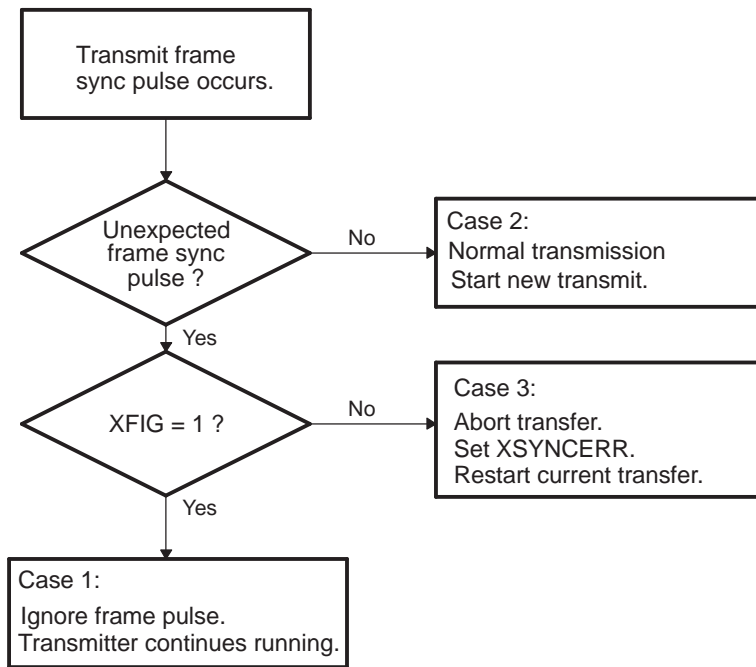
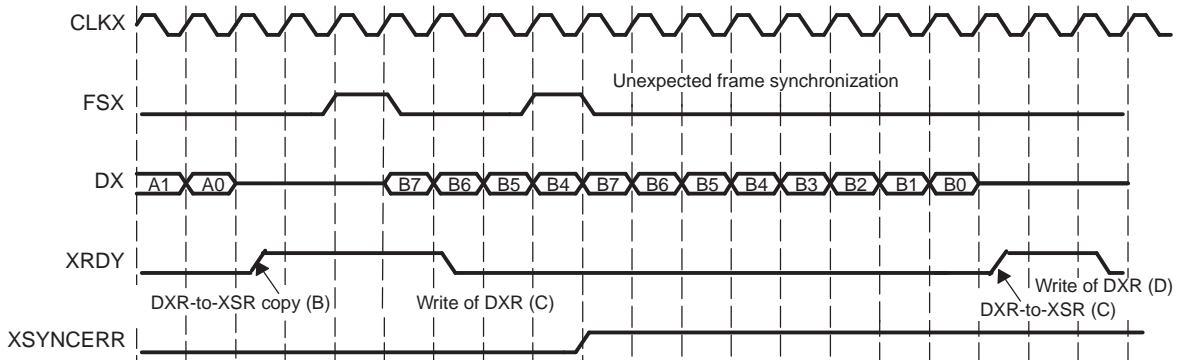


Figure 21-31. Unexpected Transmit Frame Synchronization Pulse

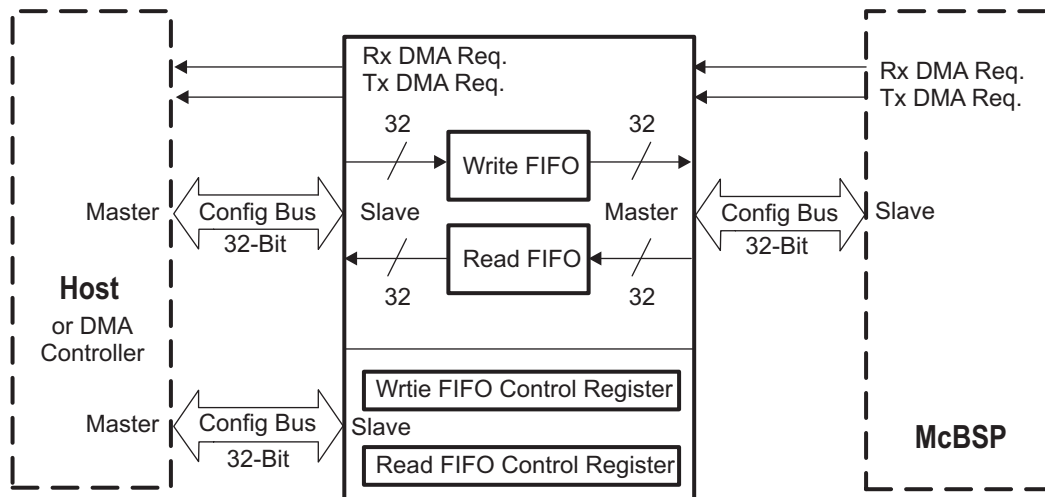


### 21.2.7.6 McBSP Buffer FIFO (BFIFO)

The BFIFO contains two FIFOs: one Read FIFO (RFIFO) and one Write FIFO (WFIFO). To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See [Figure 21-32](#) for a high-level block diagram of the BFIFO.

The BFIFO may be enabled/disabled and configured via the Write FIFO control register (WFIFCTL) and the Read FIFO control register (RFIFCTL). Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the McBSP (see [Section 21.2.12](#) for details).

**Figure 21-32. McBSP Buffer FIFO (BFIFO) Block Diagram**



Note that the McBSP Buffer FIFO (BFIFO) has a different memory-map (see your device-specific data manual) than the McBSP memory-mapped registers (MMRs); hence, the BFIFO is accessible by way of a different Configuration Bus.

#### 21.2.7.6.1 BFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the McBSP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the McBSP generates transmit DMA requests as needed; the BFIFO is "invisible" to the McBSP.

When the Write FIFO is enabled, transmit DMA requests from the McBSP are sent to the BFIFO, which in turn generates transmit DMA requests to the host/DMA controller.

If the Write FIFO is enabled, upon a transmit DMA request from the McBSP, the WFIFO writes *WNUMDMA* 32-bit words to the McBSP, if and when there are at least *WNUMDMA* words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied; at this point, it writes *WNUMDMA* words to the McBSP.

If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded.

Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

### **21.2.7.6.1.1 Transmit DMA Event Pacer**

The BFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the host or DMA controller is reduced.

If the Write FIFO has space to accept *WNUM EVT* 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once *WNUM EVT* words have been written to the WFIFO, the WFIFO checks again to see if there is space for *WNUM EVT* 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled.

Note that if transmit DMA event pacing is desired, the *WNUM EVT* bits in *WFIFOCTL* should be set to a non-zero integer multiple of the value in the *WNUM DMA* bits. If transmit DMA event pacing is not desired, then the value in the *WNUM EVT* bits should be set equal to the value in the *WNUM DMA* bits.

### **21.2.7.6.2 BFIFO Data Reception**

When the Read FIFO is disabled, receive DMA requests pass through directly from McBSP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the McBSP generates receive DMA requests as needed; the BFIFO is "invisible" to the McBSP.

When the Read FIFO is enabled, receive DMA requests from the McBSP are sent to the BFIFO, which in turn generates receive DMA requests to the host/DMA controller.

If the Read FIFO is enabled and the McBSP makes a receive DMA request, the RFIFO reads *RNUM DMA* 32-bit words from the McBSP, if and when the RFIFO has space for *RNUM DMA* words. If the RFIFO does not have space, the RFIFO waits until this condition has been satisfied; at this point, it reads *RNUM DMA* words from the McBSP.

If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than *RNUM EVT* words, those words will be read correctly, emptying the RFIFO.

#### **21.2.7.6.2.1 Receive DMA Event Pacer**

The BFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced.

If the Read FIFO contains at least *RNUM EVT* 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once *RNUM EVT* 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another *RNUM EVT* words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty.

Note that if receive DMA event pacing is desired, the *RNUM EVT* bits in *RFIFOCTL* should be set to a non-zero integer multiple of the value in *RNUM DMA* bits. If receive DMA event pacing is not desired, then the value in the *RNUM EVT* bits should be set equal to the value in the *RNUM DMA* bits.

### **21.2.7.6.3 Arbitration Between Transmit and Receive DMA Requests**

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

### 21.2.8 $\mu$ -Law/A-Law Companding Hardware Operation

Companding (compressing and expanding) hardware allows compression and expansion of data in either  $\mu$ -law or A-law format. The specification for  $\mu$ -law and A-law log PCM is part of the CCITT G.711 recommendation. The companding standard employed in the United States and Japan is  $\mu$ -law and allows 14 bits of dynamic range. The European companding standard is A-law and allows 13 bits of dynamic range. Any values outside these ranges are set to the most positive or most negative value. Thus, for companding to work best here, the data transferred to and from the McBSP via the CPU or the EDMA controller must be at least 16 bits wide.

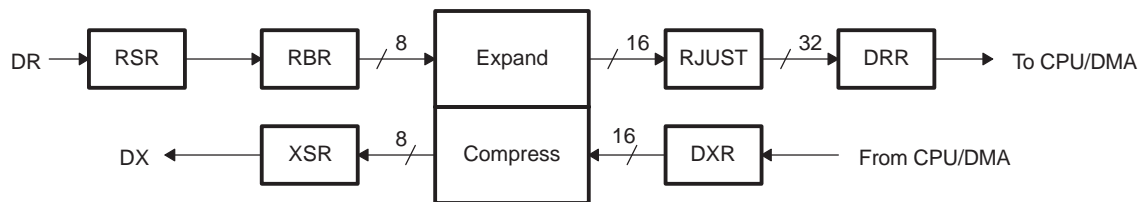
The  $\mu$ -law and A-law formats encode data into 8-bit code elements. Companded data is always 8 bits wide, so the appropriate (R/X)WDLEN1/2 must be cleared to 0, indicating an 8-bit serial data stream. If companding is enabled and either phase of the frame does not have an 8-bit element length, companding continues as if the element length is eight bits.

When companding is used, transmit data is encoded according to the specified companding law, and receive data is decoded to 2s-complement format. Companding is enabled and the desired format is selected by appropriately setting (R/X)COMPAND in the (R/X)CR. Compression occurs during the process of copying data from DXR to XSR and expansion occurs from RBR to DRR, as shown in [Figure 21-33](#).

For transmit data to be compressed, it should be 16-bit, left-justified data, such as LAW16, as shown in [Figure 21-34](#). The value can be either 13 or 14 bits wide, depending on the companding law. This 16-bit data is aligned in DXR, as shown in [Figure 21-35](#).

For reception, the 8-bit compressed data in RBR is expanded to a left-justified 16-bit data, LAW16. This can be further justified to 32-bit data by programming the RJUST bits in the serial port control register (SPCR), as shown in [Table 21-12](#).

**Figure 21-33. Companding Flow**



**Figure 21-34. Companding Data Formats**

LAW16	15		2	1	0	
$\mu$ -Law		Value		0	0	
LAW16	15		3	2	1	0
A-Law		Value		0	0	0

**Figure 21-35. Transmit Data Companding Format in DXR**

31		16	15		0
	Don't care		LAW16		

**Table 21-12. Justification of Expanded Data in DRR**

RJUST Bit in SPCR	DRR Bits			
	31	16	15	0
00		0		LAW16
01		sign		LAW16
10		LAW16		0
11			Reserved	

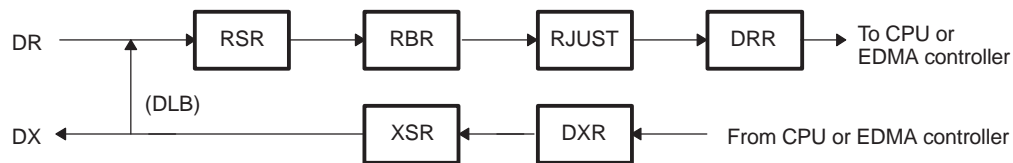
**21.2.8.1 Companding Internal Data**

If the McBSP is unused, the companding hardware can compand internal data. This hardware can be used to:

- Convert linear data to the appropriate  $\mu$ -law or A-law format.
- Convert  $\mu$ -law or A-law data to the linear format.
- Observe the quantization effects in companding by transmitting linear data and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

Figure 21-36 shows the method by which the McBSP can compand internal data. The data path is indicated by the (DLB) arrow. The McBSP is enabled in digital loopback (DLB) mode with companding appropriately enabled by the RCOMPAND and XCOMPAND bits. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or the EDMA controller to these conversions, respectively. The time for this companding depends on the serial bit rate selected.

**Figure 21-36. Companding of Internal Data**



**21.2.8.2 Bit Ordering**

Normally, all transfers on the McBSP are sent and received with the MSB first. However, certain 8-bit data protocols (that do not use companded data) require the LSB to be transferred first. By setting the (R/X)COMPAND = 01b in (R/X)CR, the bit ordering of 8-bit elements is reversed (LSB first) before being sent to the serial port. Like the companding feature, this feature is enabled only if the appropriate (R/X)WDLEN1/2 bit is cleared to 0, indicating that 8-bit elements are to be serially transferred. A 32-bit bit reversal feature is also available, as shown in Section 21.2.5.5.7

## 21.2.9 Multichannel Selection Modes

This section defines and provides the functions and all related information concerning the multichannel selection modes.

### 21.2.9.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission. In the receiver and in the transmitter, the 128 available channels are divided into eight blocks that each contain 16 contiguous channels:

Block 0: Channels 0 through 15	Block 4: Channels 64 through 79
Block 1: Channels 16 through 31	Block 5: Channels 80 through 95
Block 2: Channels 32 through 47	Block 6: Channels 96 through 111
Block 3: Channels 48 through 63	Block 7: Channels 112 through 127

The blocks are assigned to partitions according to the selected partition mode. In the 2-partition mode, you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode, blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use 2 receive partitions (A and B) and 8 transmit partitions (A through H).

### 21.2.9.2 Multichannel Selection

When McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels. The McBSP has one receive multichannel selection mode and three transmit multichannel selection modes.

Each channel partition has a dedicated channel enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

### 21.2.9.3 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.
- Set a frame length (RFRLN1/XFRLN1) that includes the highest-numbered channel that is to be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLN1 = 39). If XFRLN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

---

**NOTE:** The frame-sync pulse can be generated internally by the sample rate generator or it can be supplied externally by another source. In a multichannel mode configuration with external frame-sync generation, the McBSP transmitter will ignore the first frame-sync pulse after it is taken out of reset. The transmitter will transmit only on the second frame-sync pulse. The receiver will shift in data on the first frame-sync pulse, regardless of whether it is generated internally or externally.

---



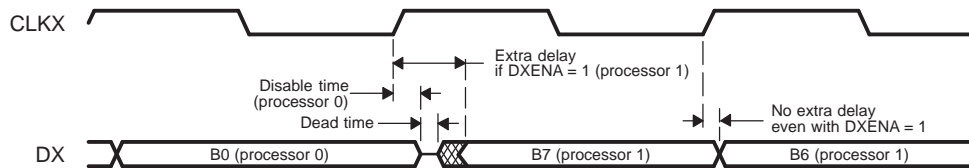
### 21.2.9.4 DX Enabler: DXENA

The DXENA bit in the serial port control register (SPCR) controls the high-impedance enable on the DX pin. When DXENA = 1, the McBSP enables extra delay for the DX pin turn-on time. This feature is useful for McBSP multichannel operations, such as in a time-division multiplexed (TDM) system. The McBSP supports up to 128 channels in a multichannel operation. These channels can be driven by different devices in a TDM data communication line, such as the T1/E1 line. In any multichannel operation where multiple devices transmit over the same DX line, you need to ensure that no two devices transmit data simultaneously, which results in bus contention. Enough dead time should exist between the transmission of the first data bit of the current device and the transmission of the last data bit of the previous device. In other words, the last data bit of the previous device needs to be disabled to a high-impedance state before the next device begins transmitting data to the same data line, as shown in [Figure 21-37](#).

When two McBSPs are used to transmit data over the same TDM line, bus contention occurs if DXENA = 0. The first McBSP turns off the transmission of the last data bit (changes DX from valid to a high-impedance state) after a disable time specified in the data manual. As shown in [Figure 21-37](#), this disable time is measured from the CLKX active clock edge. The next McBSP turns on its DX pin (changes from a high-impedance state to valid) after a delay time. Again, this delay time is measured from the CLKX active clock edge. Bus contention occurs because the dead time between the two devices is not enough. You need to apply alternative software or hardware methods to ensure proper multichannel operation in this case.

If you set DXENA = 1 in the second McBSP, the second McBSP turns on its DX pin after some extra delay time. This ensures that the previous McBSP on the same DX line is disabled before the second McBSP starts driving out data. The DX enabler controls only the high-impedance enable on the DX pin, not the data itself. Data is shifted out to the DX pin at the same time as in the case when DXENA = 0. The only difference is that with DXENA = 1, the DX pin is masked to a high-impedance state for some extra CPU cycles before the data is seen on the TDM data line. Therefore, only the first bit of data is delayed. Refer to the specific device datasheet for the exact amount of delay.

**Figure 21-37. DX Timing for Multichannel Operation**



### 21.2.9.5 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions. If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A.



### 21.2.9.5.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B (see [Table 21-13](#)), which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up to 32 enabled transmit channels) can be assigned to transmit partitions A and B (see [Table 21-14](#)). You can dynamically change which blocks of channels are assigned to the partitions, see [Section 21.2.9.5.2](#).

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bit in the multichannel control register (MCR). In the receive multichannel selection mode, the channels in this partition are controlled by the enhanced receive channel enable register partition A/B (RCERE0).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bit in MCR. In the receive multichannel selection mode, the channels in this partition are controlled by the enhanced receive channel enable register partition A/B (RCERE0).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bit in the multichannel control register (MCR). In one of the transmit multichannel selection modes, the channels in this partition are controlled by the enhanced transmit channel enable register partition A/B (XCERE0).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bit in MCR. In one of the transmit multichannel selection modes, the channels in this partition are controlled by the enhanced transmit channel enable register partition A/B (XCERE0).

**Table 21-13. Receive Channel Assignment and Control When Two Receive Partitions are Used**

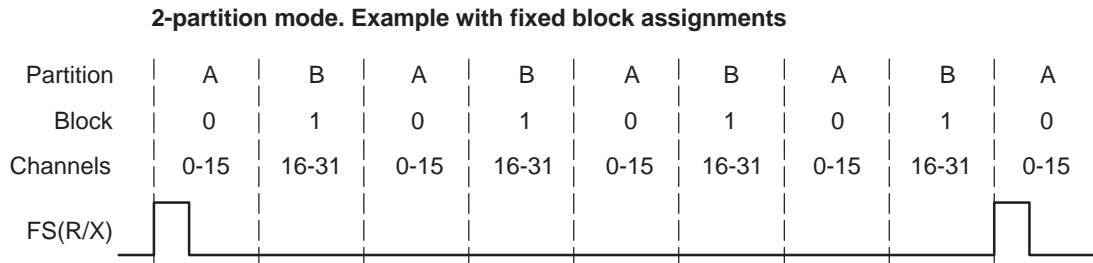
Receive Partition	Assigned Block of Receive Channels	RPABLK Bit in MCR	RPBBLK Bit in MCR	RCERE0 Bits
A	Block 0: channels 0 through 15	0	–	RCE0-RCE15
A	Block 2: channels 32 through 47	1h	–	RCE0-RCE15
A	Block 4: channels 64 through 79	2h	–	RCE0-RCE15
A	Block 6: channels 96 through 111	3h	–	RCE0-RCE15
B	Block 1: channels 16 through 31	–	0	RCE16-RCE31
B	Block 3: channels 48 through 63	–	1h	RCE16-RCE31
B	Block 5: channels 80 through 95	–	2h	RCE16-RCE31
B	Block 7: channels 112 through 127	–	3h	RCE16-RCE31

**Table 21-14. Transmit Channel Assignment and Control When Two Transmit Partitions are Used**

Transmit Partition	Assigned Block of Transmit Channels	XPABLK Bit in MCR	XPBBLK Bit in MCR	XCERE0 Bits
A	Block 0: channels 0 through 15	0	–	XCE0-XCE15
A	Block 2: channels 32 through 47	1h	–	XCE0-XCE15
A	Block 4: channels 64 through 79	2h	–	XCE0-XCE15
A	Block 6: channels 96 through 111	3h	–	XCE0-XCE15
B	Block 1: channels 16 through 31	–	0	XCE16-XCE31
B	Block 3: channels 48 through 63	–	1h	XCE16-XCE31
B	Block 5: channels 80 through 95	–	2h	XCE16-XCE31
B	Block 7: channels 112 through 127	–	3h	XCE16-XCE31

Figure 21-38 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0-15 have been assigned to partition A, and channels 16-31 have been assigned to partition B. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

**Figure 21-38. Alternating Between the Channels of Partition A and the Channels of Partition B**



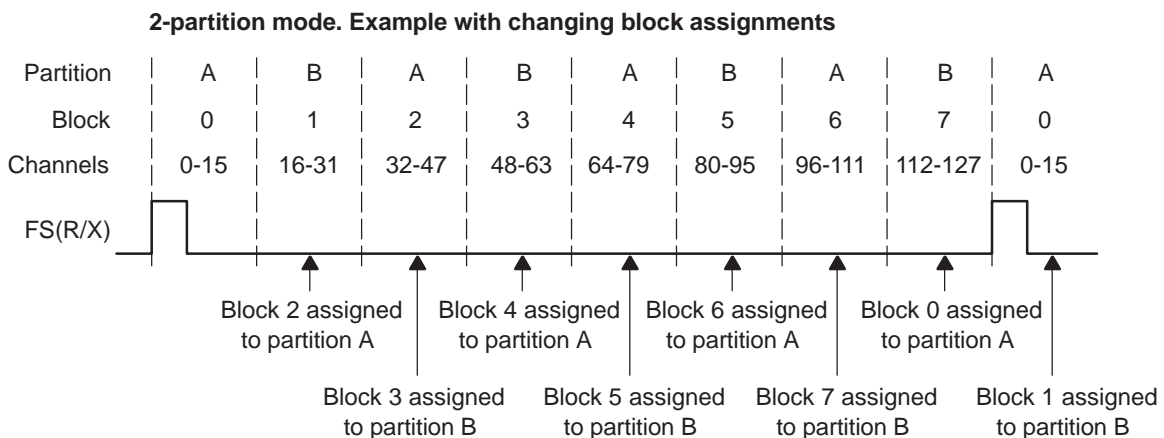
**21.2.9.5.2 Reassigning Blocks During Reception/Transmission**

If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, it's the associated block assignment bits cannot be modified, and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you cannot modify (R/X)PABLK to assign different channels to partition A, and you cannot modify (R/X)CERE0 to change the channel configuration for partition A. Several features of the McBSP helps to time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition.

Figure 21-39 shows an example of reassigning channels throughout a data transfer. In response to a frame-sync pulse, the McBSP alternates between partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever partition A is active, the CPU changes the block assignment for partition B.

**Figure 21-39. Reassigning Channel Blocks Throughout a McBSP Data Transfer**



### 21.2.9.6 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions. If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP partitions are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A. In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in [Table 21-15](#) and [Table 21-16](#). These assignments cannot be changed. [Table 21-15](#) and [Table 21-16](#) also show the registers used to control the channels in the partitions.

**Table 21-15. Receive Channel Assignment and Control When Eight Receive Partitions are Used**

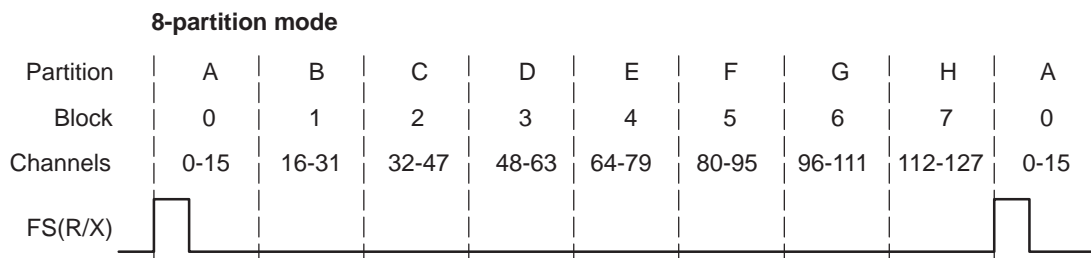
Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCERE0
B	Block 1: channels 16 through 31	RCERE0
C	Block 2: channels 32 through 47	RCERE1
D	Block 3: channels 48 through 63	RCERE1
E	Block 4: channels 64 through 79	RCERE2
F	Block 5: channels 80 through 95	RCERE2
G	Block 6: channels 96 through 111	RCERE3
H	Block 7: channels 112 through 127	RCERE3

**Table 21-16. Transmit Channel Assignment and Control When Eight Transmit Partitions are Used**

Transmit Partition	Assigned Block of Transmit Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	XCERE0
B	Block 1: channels 16 through 31	XCERE0
C	Block 2: channels 32 through 47	XCERE1
D	Block 3: channels 48 through 63	XCERE1
E	Block 4: channels 64 through 79	XCERE2
F	Block 5: channels 80 through 95	XCERE2
G	Block 6: channels 96 through 111	XCERE3
H	Block 7: channels 112 through 127	XCERE3

[Figure 21-40](#) shows an example of the McBSP using the 8-partition mode. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

**Figure 21-40. McBSP Data Transfer in the 8-Partition Mode**



### 21.2.9.7 Receive Multichannel Selection Mode

The RMCM bit in the multichannel control register (MCR) determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When RMCM = 1, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate enhanced receive channel enable register (RCEREN). The way channels are assigned to the RCEREN depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit in MCR.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register (RBR). The receiver does not copy the content of the RBR to the DRR, and as a result, does not set the receiver ready bit (RRDY). Therefore, no DMA synchronization event (REVT) is generated, and if the receiver interrupt mode depends on RRDY (RINTM = 0), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1. Accepts bits shifted in from the DR pin in channel 0.
2. Ignores bits received in channels 1-14.
3. Accepts bits shifted in from the DR pin in channel 15.
4. Ignores bits received in channels 16-38.
5. Accepts bits shifted in from the DR pin in channel 39.

### 21.2.9.8 Transmit Multichannel Selection Mode

The XMCM bit in the multichannel control register (MCR) determines whether all channels or only selected channels are enabled and unmasked for transmission. The McBSP has three transmit multichannel selection modes (XMCM = 1, XMCM = 2h, and XMCM = 3h), which are described in [Table 21-17](#).

**Table 21-17. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits**

XMCM Bit in MCR	Transmit Multichannel Selection Mode
0	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
1h	All channels are disabled unless they are selected in the appropriate enhanced transmit channel enable register (XCEREN). If enabled, a channel in this mode is also unmasked. The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in XCEREN.
2h	All channels are enabled, but they are masked unless they are selected in the appropriate enhanced transmit channel enable register (XCEREN). The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in XCEREN.
3h	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate enhanced receive channel enable register (RCEREN). Once enabled, they are masked unless they are also selected in the appropriate enhanced transmit channel enable register (XCEREN). The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in RCEREN and XCEREN.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 1 (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP:

1. Shifts data to the DX pin in channel 0.
2. Places the DX pin in the high-impedance state in channels 1–14.
3. Shifts data to the DX pin in channel 15.
4. Places the DX pin in the high-impedance state in channels 16–38.
5. Shifts data to the DX pin in channel 39.

### 21.2.9.8.1 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- Enabled and unmasked (transmission can begin and can be completed)
- Enabled but masked (transmission can begin but cannot be completed)
- Disabled (transmission cannot occur)

The following definitions explain the channel control options:

<b>Enabled channel</b>	A channel that can begin transmission by passing data from the data transmit register (DXR) to the transmit shift register (XSR).
<b>Masked channel</b>	A channel that cannot complete transmission. The DX pin is held in the high-impedance state; data cannot be shifted out on the DX pin.  In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.
<b>Disabled channel</b>	A channel that is not enabled. A disabled channel is also masked.  Because no DXR-to-XSR copy occurs, the XRDY bit in SPCR is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 0 in SPCR), no interrupt is generated.  The XEMPTY bit in SPCR is not affected.
<b>Unmasked channel</b>	A channel that is not masked. Data in the XSR is shifted out on the DX pin.

### 21.2.9.8.2 Activity on McBSP Pins for Different Values of XMCM

Figure 21-41 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- In transmit control register (XCR):
  - XPHASE = 0: Single-phase frame (required for multichannel selection modes)
  - XFRLEN1 = 3h: 4 words per frame
  - XWDLEN1 = 0: 8 bits per word
- In multichannel control register (MCR):
  - XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 3h, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

In Figure 21-41, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

### 21.2.9.9 Using Interrupts Between Block Transfers

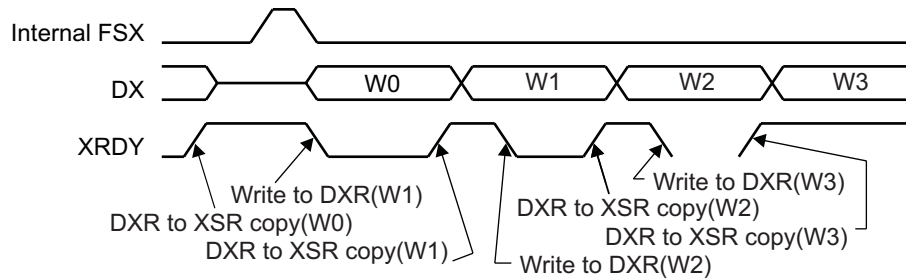
When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if the RINTM bit in the serial port control register (SPCR) is set to 1. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if the XINTM bit in SPCR is set to 1. When RINTM/XINTM = 1, no interrupt is generated unless a multichannel selection mode is on.

These interrupt pulses are active high and last for two McBSP internal input clock cycles.

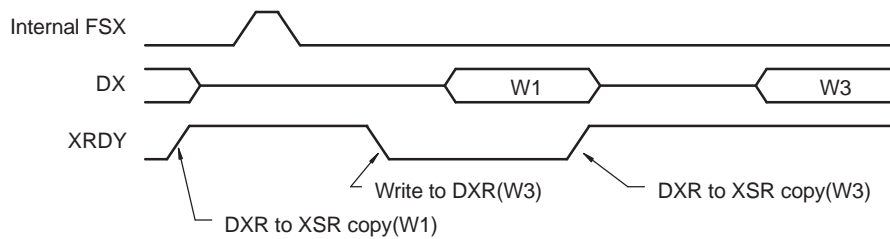
This type of interrupt is especially helpful if you are using the two-partition mode and you want to know when you can assign a different block of channels to partition A or B.

**Figure 21-41. Activity on McBSP Pins for the Possible Values of XMCM**

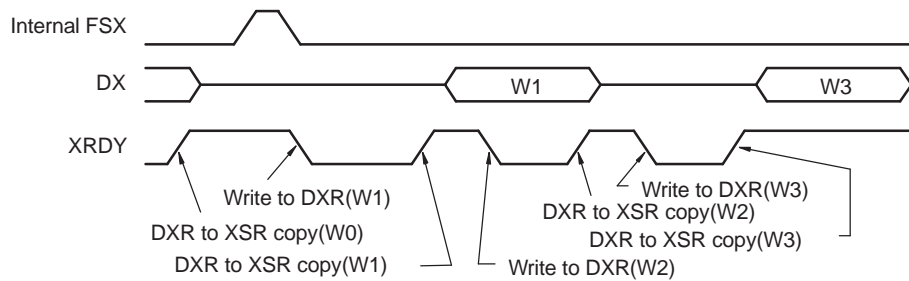
(a)  $XMCM = 0$ : All channels enabled and unmasked



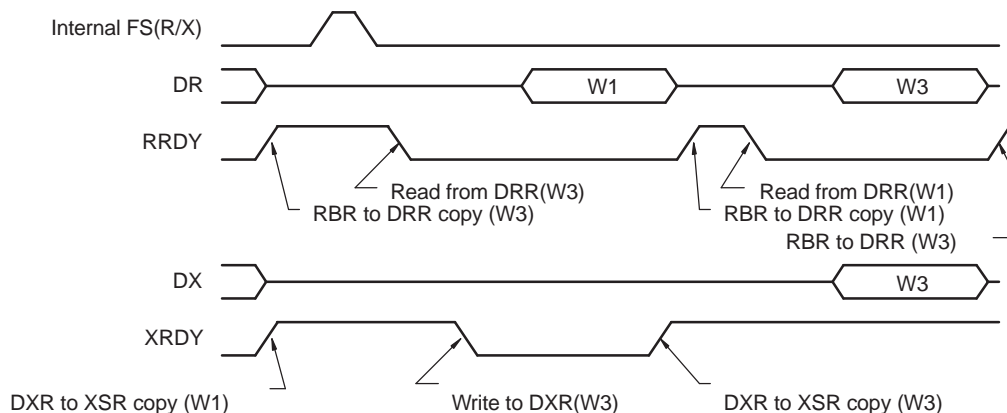
(b)  $XMCM = 1h$ ,  $XPABLK = 0$ ,  $XCERE0 = 0000\ 000Ah$ : Only channels 1 and 3 enabled and unmasked



(c)  $XMCM = 2h$ ,  $XPABLK = 0$ ,  $XCERE0 = 0000\ 000Ah$ : All channels enabled, only 1 and 3 unmasked



(d)  $XMCM = 3h$ ,  $RPABLK = 0$ ,  $XPABLK = x$ ,  $RCERE0 = 0000\ 0008h$ ,  $XCERE0 = 0000\ 000Ah$ : Receive channels: 1 and 3 enabled; transmit channels: 1 and 3 enabled, but only 3 unmasked



### 21.2.10 SPI Operation Using the Clock Stop Mode

The McBSP on this device does not support the SPI protocol.

### 21.2.11 Resetting the Serial Port: Rrst, Xrst, Grst, and Reset

Device reset or McBSP reset: When the McBSP is reset by device reset or McBSP reset, the state machine is reset to its initial state. All counters and status bits are reset. This includes the receive status bits RFULL, RRDY, and RSYNCERR, and the transmit status bits XEMPTY, XRDY, and XSYNCERR in the serial port control register (SPCR).

The serial port can be reset in the following two ways:

- Device reset (RESET pin is low) places the receiver, the transmitter, and the sample rate generator in reset. When the device reset is removed (RESET = 1), FRST = GRST = Rrst = Xrst = 0 in SPCR, keeping the entire serial port in the reset state.
- The serial port transmitter and receiver can be independently reset by the Xrst and Rrst bits in SPCR. The sample rate generator is reset by the GRST bit in SPCR.

Table 21-18 shows the state of the McBSP pins when the serial port is reset by these methods.

**Table 21-18. Reset State of McBSP Pins**

Pin	Direction	Device Reset ( RESET = 0)	McBSP Reset
<b>Receiver Reset (Rrst = 0 and Grst = 1)</b>			
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR if output
FSR	I/O/Z	Input	Known state if input; FSRP(inactive state) if output
CLKS	I	Input	Input
<b>Transmitter Reset (Xrst = 0 and Grst = 1)</b>			
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if input; CLKX if output
FSX	I/O/Z	Input	Known state if input; FSXP(inactive state) if output
CLKS	I	Input	Input

#### 21.2.11.1 Software Reset Considerations

**McBSP reset:** When the receiver and transmitter reset bits, Rrst and Xrst in SPCR, are written with 0, the respective portions of the McBSP are reset and activity in the corresponding section stops. All input-only pins, such as DR, and all other pins that are configured as inputs are in a known state. FS(R/X) is driven to its inactive state (same as its polarity bit, FS(R/X)P) if it is an output. If CLK(R/X) are programmed as outputs, they are driven by CLKG, provided that GRST = 1. The DX pin is in the high-impedance state when the transmitter is reset. During normal operation, the sample rate generator can be reset by writing a 0 to GRST. The sample rate generator should only be reset when not being used by the transmitter or the receiver. In this case, the internal sample rate generator clock CLKG, and its frame sync signal (FSG) is driven inactive (low). When the sample rate generator is not in the reset state (GRST = 1), FSR and FSX are in an inactive state when Rrst = 0 and Xrst = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when FRST = 1 and frame sync is driven by FSG.

**Sample-rate generator reset:** As mentioned previously, the sample rate generator is reset when the device is reset or when its reset bit, GRST in SPCR, is written with 0.

**Emulator software reset:** In the event of an emulator software reset initiated from the DSP, the McBSP register values are reset to their default values.



### 21.2.11.2 Hardware Reset Considerations

When the McBSP is reset due to device reset, the entire serial port (including the transmitter, receiver, and the sample rate generator) is reset. All input-only pins and 3-state pins should be in a known state. The output-only pin, DX, is in the high-impedance state. When the device is pulled out of reset, the serial port remains in the reset condition (RRST = XRST = FRST = GRST = 0 in SPCR).

### 21.2.12 McBSP Initialization Procedure

The McBSP initialization procedure varies depending on the specific system setup. [Section 21.2.12.1](#) provides a general initialization sequence.

The transmitter and the receiver of the McBSP can operate independently from each other. Therefore, they can be placed in or taken out of reset individually by modifying only the desired bit in the registers without disrupting the other portion. The steps in the following sections discuss the initialization procedure for taking both the transmitter and the receiver out of reset. To initialize only one portion, configure only the portion desired.

The McBSP internal sample rate generator and internal frame sync generator are shared between the transmitter and the receiver. [Table 21-19](#) and [Table 21-20](#) describe their usage base upon the clock and frame sync configurations of the receiver and transmitter, respectively.

**Table 21-19. Receiver Clock and Frame Configurations**

CLKR Source	FSR Source	Comment on Configuration
Internal	Internal	The McBSP internal sample rate generator and internal frame sync generator are used by the receiver.
External	Internal	The McBSP internal sample rate generator and internal frame sync generator are used by the receiver.
Internal	External	The McBSP internal sample rate generator is used but the internal frame sync generator is not used by the receiver.
External	External	The McBSP internal sample rate generator and internal frame sync generator are not used by the receiver.

**Table 21-20. Transmitter Clock and Frame Configurations**

CLKX Source	FSX Source	Comment on Configuration
Internal	Internal	The McBSP internal sample rate generator is used by the transmitter. The transmitter can generate frame sync FSX in one of two ways. First, it can generate FSX by using the internal frame sync generator (FSGM = 1). Alternatively, it can generate FSX upon each DXR-to-XSR copy (FSGM = 0). In this case, the internal frame sync generator can be kept in reset (FRST = 0) if it is not used by the receiver. You can follow the general initialization sequence in <a href="#">Section 21.2.12.1</a> .
External	Internal	The McBSP internal sample rate generator and internal frame sync generator are not used by the transmitter. This configuration is only valid with FSGM = 0 where the McBSP transmitter generates FSX upon each DXR-to-XSR copy. You can follow the general initialization sequence in <a href="#">Section 21.2.12.1</a> .
Internal	External	The McBSP internal sample rate generator is used by the transmitter but the internal frame sync generator is not.
External	External	The McBSP internal sample rate generator and internal frame sync generator are not used by the transmitter.



### 21.2.12.1 General Initialization Procedure

This section provides the general initialization procedure.

1. With the McBSP still in reset (Power and Sleep Controller (PSC) in the default state):
  - (a) Program the PSC registers in the System Module to put the McBSP in the enable state (see the *Power and Sleep Controller (PSC)* chapter).
  - (b) Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Ensure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).
3. Program the control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0). Also ensure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0).
4. Wait for proper internal synchronization. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR and the SCLKME bit in PCR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit clock master and the internal sample rate generator is used.
  - (a) Start the sample rate generator by setting the GRST bit to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
  - (b) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to  $1/(\text{CLKGDV} + 1)$  of the sample rate generator source clock CLKSRG.
6. Skip this step if the transmitter is not used. If the transmitter is used, a transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
  - (a) Set the XRST bit to 1 to enable the transmitter.
  - (b) Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
  - (c) Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
  - (a) If the EDMA is used to service the McBSP, setup data acquisition as desired and start the EDMA in this step, before the McBSP is taken out of reset.
  - (b) If CPU interrupt is used to service the McBSP, enable the transmit and/or receive interrupt as required.
  - (c) If CPU polling is used to service the McBSP, no action is required in this step.
8. Set the XRST bit and/or the RRST bit to 1 to enable the corresponding section of the McBSP. The McBSP is now ready to transmit and/or receive.
  - (a) If the EDMA is used to service the McBSP, it services the McBSP automatically upon receiving the XEVT and/or REVT.
  - (b) If CPU interrupt is used to service the McBSP, the interrupt service routine is automatically entered upon receiving the XINT and/or RINT.
  - (c) If CPU polling is used to service the McBSP, it can do so now by polling the XRDY and/or RRDY bit.
9. If the internal frame sync generator is used (FSGM = 1), proceed to the additional steps to turn on the internal frame sync generator. Initialization is complete if any one of the following is true:
  - (a) The external device generates frame sync FSX and/or FSR. The McBSP is now ready to transmit and/or receive upon receiving external frame sync.
  - (b) The McBSP generates transmit frame sync FSX upon each DXR-to-XSR copy. The internal frame sync generator is not used (FSGM = 0).

The following additional steps to turn on the internal frame sync generator apply only if FSGM = 1:

10. Skip this step if the transmitter is not used. If the transmitter is used, ensure that DXR is serviced before you start the internal frame sync generator. You can do so by checking XEMPTY = 1 (XSR is not empty) in SPCR.
11. Set the FRST bit to 1 to start the internal frame sync generator. The internal frame sync signal FSG is generated on a CLKG active edge after 7 to 8 CLKG clocks have elapsed.

### 21.2.12.2 Special Case: External Device is the Transmit Frame Master

Care must be taken if the transmitter expects a frame sync from an external device. After the transmitter comes out of reset (XRST = 1), it waits for a frame sync from the external device. If the first frame sync arrives very shortly after the transmitter is enabled, the CPU or EDMA controller may not have a chance to service the data transmit register (DXR). In this case, the transmitter shifts out the default data in the transmit shift register (XSR) instead of the desired value, which has not yet arrived in DXR. This causes problems in some applications, as the first data element in the frame is invalid. The data stream appears element-shifted (the first data word may appear in the second channel instead of the first).

To ensure proper operation when the external device is the frame master, you must assure that DXR is already serviced with the first word when a frame sync occurs. To do so, you can keep the transmitter in reset until the first frame sync is detected. Software is setup such that it will only take the transmitter out of reset (XRST = 1) promptly after detecting the first frame sync. This assures that the transmitter does not begin data transfers at the data pin during the first frame sync period. This also provides almost an entire frame period for the DSP to service DXR with the first word before the second frame sync occurs. The transmitter only begins data transfers upon receiving the second frame sync. At this point, DXR is already serviced with the first word.

#### 21.2.12.2.1 How to Detect First Frame Sync

Although the McBSP is capable of generating an interrupt to the CPU upon the detection of frame synchronization (XINTM = 2h and/or RINTM = 2h in the serial port control register (SPCR)), the McBSP requires the associated portion (receiver/transmitter) of the McBSP to be out of reset in order for the interrupt to be generated. Therefore, instead of directly using the McBSP interrupt to detect the first frame sync, you can use the GPIO peripheral. This can be achieved by connecting the frame sync signal to a GPIO pin. Software can either poll the GPIO pin to detect the first frame sync or program the GPIO peripheral to generate an interrupt to the CPU upon detecting the first frame sync edge. For more information on the GPIO peripheral, see the *General-Purpose Input/Output (GPIO)* chapter.

The following are some recommended GPIO pin(s) on the device that you can use to detect the first McBSP external frame sync:

- **GPIO pin located near the McBSP pins.** Connect the external frame sync to both the McBSP FSX/FSR pin(s) and the dedicated GPIO pin.
- **GPIO pin multiplexed with the McBSP FSX signal.** Note that on the device, the GPIO pins (of the GPIO peripheral) are multiplexed with the McBSP pins. Software can program the device's pin multiplexing register (PINMUX) to default these pins to the GPIO function, and only switch them to the McBSP function upon detecting the first frame sync. This method is only recommended if the external device is both the frame sync and clock master; that is, the external device drives both the FSX and CLKX signals. This method is not recommended if the McBSP is the clock master (driving CLKX and/or CLKR), as the "on-the-fly" pin multiplexed switching can cause a glitch on the CLKX/CLKR pin. For more details on pin multiplexing, see the device-specific data manual.

#### 21.2.12.2.2 Initialization Procedure When External Device is Frame Sync Master

The initialization procedure assumes the following:

- Using a GPIO pin multiplexed with the McBSP FSX signal. If a dedicated GPIO pin is used instead, skip step 1 and step 8b.
- Software polls the GPIO pin to detect the first frame sync. If the GPIO interrupt is used instead to detect the first frame sync, step 8 can be performed within an interrupt service routine (ISR).

1. The GPIO and McBSP signals are multiplexed together on the device. Start by programming the pin multiplexing register (PINMUX) to select the GPIO function on the GPIO/McBSP multiplexed pins. Program the GPIO peripheral so that these pins function as GPIO inputs.
2. Ensure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).
3. Program the sample rate generator register (SRGR) and other control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0 in SPCR). Also ensure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0 in SPCR).
4. Wait for proper McBSP internal synchronization:
  - (a) If the external device provides the bit clock, wait for two CLKR or CLKX cycles. Skip step 5.
  - (b) If the McBSP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit clock master and the internal sample rate generator is used.
  - (a) Start the sample rate generator by setting the GRST bit in SPCR to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
  - (b) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to  $1/(\text{CLKGDV} + 1)$  of the sample rate generator source clock CLKSRG.
6. A transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
  - (a) Set the XRST bit in SPCR to 1 to enable the transmitter.
  - (b) Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
  - (c) Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
  - (a) If the EDMA controller is used to service the McBSP, setup data acquisition as desired and start the EDMA controller in this step, before the McBSP is taken out of reset.
  - (b) If the CPU interrupt is used to service the McBSP, no action is required in this step.
  - (c) If CPU polling is used to service the McBSP, no action is required in this step.
8. Poll the GPIO pin (through reading the appropriate registers in the GPIO peripheral) to detect the first transmit frame sync from the external device. Upon detection of the first frame sync, perform the following in this order:
  - (a) Set the XRST bit and/or the RRST bit to 1 to enable the respective portion of the McBSP. The McBSP is now ready to transmit and/or receive.
  - (b) Program PINMUX to switch the GPIO/McBSP multiplexed pins to the McBSP function.
9. Service the McBSP:
  - (a) If CPU polling is used to service the McBSP in normal operations, it can do so upon exit from the ISR.
  - (b) If the CPU interrupt is used to service the McBSP in normal operations, upon XRDY interrupt service routine is entered. The ISR should be setup to verify that XRDY = 1 and service the McBSP accordingly.
  - (c) If the EDMA controller is used to service the McBSP in normal operations, it services the McBSP automatically upon receiving the XEVT and/or REVT.
10. Upon detection of the second frame sync, DXR is already serviced and the transmitter is ready to transmit the valid data. The receiver is also serviced properly by the DSP.

### 21.2.13 Interrupt Support

The McBSP can send both receive and transmit interrupts to the DSP controller. For more details on the Interrupt Controller, see the *TMS320C674x DSP Megamodule Reference Guide* ([SPRUFK5](#)).

#### 21.2.13.1 Interrupt Events and Requests

The RRDY and XRDY bits in the serial port control register (SPCR) indicate the ready state of the McBSP receiver and transmitter, respectively. Writes and reads from the serial port can be synchronized by any of the following methods:

- Polling RRDY and XRDY bits in SPCR
- Using the events sent to the EDMA controller (REVT and XEVT)
- Using the interrupts to the CPU (RINT and XINT) that the events generate

Reading DRR and writing to DXR affects RRDY and XRDY, respectively.

##### 21.2.13.1.1 Interrupt Events: RINT and XINT

The receive interrupt (RINT) and transmit interrupt (XINT) signals inform the DSP of changes to the serial port status. Three options exist for configuring these interrupts. These options are set by the receive/transmit interrupt mode bits (RINTM and XINTM) in SPCR. The possible values of the mode, and the configurations they represent, are:

- (R/X)INTM = 00b: Interrupt on every serial element by tracking the (R/X)RDY bits in SPCR.
- (R/X)INTM = 01b: Interrupt at the end of a subframe (16 elements or less) within a frame. See [Section 21.2.9.9](#) for more details.
- (R/X)INTM = 10b: Interrupt on detection of frame synchronization pulses. The associated portion (receiver/transmitter) of the McBSP must be out of reset.
- (R/X)INTM = 11b: Interrupt on frame synchronization error. Note that if any of the other interrupt modes are selected, (R/X)SYNCERR may be read when servicing the interrupts to detect this condition. See [Section 21.2.7.5.2](#) and [Section 21.2.7.5.5](#) for more details on synchronization error.

##### 21.2.13.1.2 Receive Ready Status: RINT and RRDY

RRDY = 1 indicates that the RBR contents have been copied to DRR and that the data can now be read by either the CPU or the EDMA controller. Once that data has been read by either the CPU, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the McBSP receive interrupt (RINT) to the CPU if RINTM = 00b (default value) in SPCR.

##### 21.2.13.1.3 Transmit Ready Status: XINT and XRDY

XRDY = 1 indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the CPU, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The CPU can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the McBSP transmit interrupt (XINT) to the CPU if XINTM = 00b (default value) in SPCR.

---

**NOTE:** If the polling method is used to service the transmitter, the CPU should wait for one McBSP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to overwrites.

---

### 21.2.13.2 Interrupt Multiplexing

The RINT and XINT interrupts generated by the McBSP peripheral to the CPU are multiplexed with other interrupt sources. Refer to the device-specific data manual to determine how pin multiplexing affects the McBSP.

## 21.2.14 EDMA Event Support

### 21.2.14.1 Receive Ready Status: REVT and RRDY

RRDY = 1 in the serial port control register (SPCR) indicates that the RBR contents have been copied to DRR and that the data can now be read by the EDMA controller. Once that data has been read by the EDMA controller, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0 in SPCR), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the McBSP receive event to the EDMA controller (via REVT).

For detailed information on using the EDMA to read or write to the McBSP, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.

### 21.2.14.2 Transmit Ready Status: XEVT and XRDY

XRDY = 1 in the serial port control register (SPCR) indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST transitions from 0 to 1 in SPCR), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the EDMA controller, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The EDMA controller can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the transmit synchronization event to the EDMA controller (via XEVT).

For detailed information on using the EDMA to read or write to the McBSP, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.

---

**NOTE:** If the polling method is used to service the transmitter, the CPU should wait for one McBSP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to overwrites.

---



### 21.2.15 Power Management

The McBSP can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device.

In order for the McBSP to be placed in power-down mode by the PSC, ensure that the XRDY and RRDY flags in the serial port control register (SPCR) are cleared by performing the following steps:

1. Place the McBSP in reset by clearing the XRST, RRST, FRST, and GRST bits to 0 in SPCR.  
If EDMA is being used to service the transmitter and/or the receiver, disable the associated EDMA channels.  
For detailed information on using the EDMA to read or write to the McBSP, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.
2. Switch the McBSP clocks and frames to internal clock source:
  - (a) Set the CLKSM and FSGM bits to 1 in the sample rate generator register (SRGR).
  - (b) Set the CLKXM, CLKRM, FSXM, and FSRM bits to 1 in the pin control register (PCR).
  - (c) Clear the SCLKME bit to 0 in PCR.
3. Bring the McBSP out of reset by setting the XRST, RRST, and GRST bits to 1 in SPCR.
4. Wait for two CLKSRG cycles for proper internal synchronization.
5. Write a dummy data value to the data transmit register (DXR) in order to clear the first XRDY flag.
6. Wait for at least one McBSP bit clock, since once the first dummy data value is internally copied from DXR to XSR, the XRDY flag transitions again from 0 to 1.
7. Write a second dummy data value to DXR in order to clear the second XRDY flag.
8. Check the RRDY flag in SPCR and if set to 1, read the data receive register (DRR) and discard the data to clear the RRDY flag.
9. Place the McBSP in power-down mode by issuing the proper PSC commands. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

---

**NOTE:** After waking up the McBSP from a power-down mode using the proper PSC commands, remember to reconfigure the SPCR, SRGR, and PCR registers to the clock and frame combination that they were in before entering the power-down sequence and discard the two dummy data values that were used to clear the XRDY flags. If EDMA is used, re-enable the corresponding EDMA channels.

---

### 21.2.16 Emulation Considerations

The FREE and SOFT bits are special emulation bits in the serial port control register (SPCR) that determine the state of the McBSP when an emulation suspend event occurs in the emulator. An emulation suspend event corresponds to any type of emulator access to the DSP, such as a hardware or software breakpoint, a probe point, or a printf instruction.

Table 21-21 shows the effects of the FREE and SOFT bits on the response of the McBSP to emulation suspend events.

**Table 21-21. McBSP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR**

FREE Bit in SPCR	SOFT Bit in SPCR	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition). The transmitter and receiver stop immediately in response to an emulation suspend event.
0	1	Soft stop mode. When an emulation suspend event occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode. The transmitter and receiver continue to run when an emulation suspend event occurs.

## 21.3 Registers

Table 21-22 lists the memory-mapped registers for the McBSP. See the device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 21-22 should be considered as reserved locations and the register contents should not be modified.

The McBSP control registers are accessible by the DSP CPU. You should halt the McBSP before making changes to the serial port control register (SPCR), receive control register (RCR), transmit control register (XCR), and pin control register (PCR). Changes made to these registers without halting the McBSP could result in an undefined state.

**Table 21-22. McBSP Registers**

Offset	Acronym	Register Name	Section
-	RBR <sup>(1)</sup>	Receive buffer register	—
-	RSR <sup>(1)</sup>	Receive shift register	—
-	XSR <sup>(1)</sup>	Transmit shift register	—
0h	DRR <sup>(2) (3)</sup>	Data receive register	Section 21.3.1
4h	DXR <sup>(3)</sup>	Data transmit register	Section 21.3.2
8h	SPCR	Serial port control register	Section 21.3.3
Ch	RCR	Receive control register	Section 21.3.4
10h	XCR	Transmit control register	Section 21.3.5
14h	SRGR	Sample rate generator register	Section 21.3.6
18h	MCR	Multichannel control register	Section 21.3.7
1Ch	RCERE0	Enhanced receive channel enable register partition A/B	Section 21.3.8
20h	XCERE0	Enhanced transmit channel enable register partition A/B	Section 21.3.9
24h	PCR	Pin control register	Section 21.3.10
28h	RCERE1	Enhanced receive channel enable register partition C/D	Section 21.3.8
2Ch	XCERE1	Enhanced transmit channel enable register partition C/D	Section 21.3.9
30h	RCERE2	Enhanced receive channel enable register partition E/F	Section 21.3.8
34h	XCERE2	Enhanced transmit channel enable register partition E/F	Section 21.3.9
38h	RCERE3	Enhanced receive channel enable register partition G/H	Section 21.3.8
3Ch	XCERE3	Enhanced transmit channel enable register partition G/H	Section 21.3.9
0h	BFIFOREV <sup>(4)</sup>	BFIFO Revision Identification Register	Section 21.3.11
10h	WFIFOCTL <sup>(4)</sup>	Write FIFO Control Register	Section 21.3.12
14h	WFIFOSTS <sup>(4)</sup>	Write FIFO Status Register	Section 21.3.13
18h	RFIFOCTL <sup>(4)</sup>	Read FIFO Control Register	Section 21.3.14
1Ch	RFIFOSTS <sup>(4)</sup>	Read FIFO Status Register	Section 21.3.15

<sup>(1)</sup> The RBR, RSR, and XSR are not directly accessible via the CPUs or the EDMA controller.

<sup>(2)</sup> The CPUs and EDMA controller can only read this register; they cannot write to it.

<sup>(3)</sup> The DRR and DXR are accessible via the CPUs or the EDMA controller.

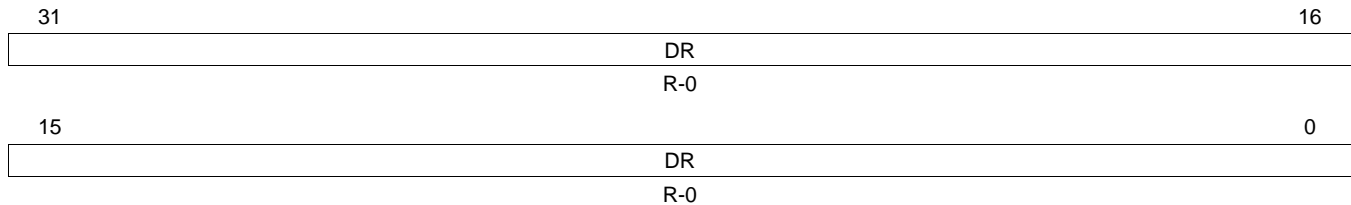
<sup>(4)</sup> The McBSP Buffer FIFO (BFIFO) has a different memory-map (see your device-specific data manual) than the McBSP memory-mapped registers (MMRs); hence, the BFIFO is accessible by way of a different Configuration Bus.

### 21.3.1 Data Receive Register (DRR)

The data receive register (DRR) contains the value to be written to the data bus. The DRR is shown in [Figure 21-42](#) and described in [Table 21-23](#).

See the device-specific data manual for the memory address of these registers. Both the CPUs and the EDMA can access DRR in all the memory-mapped locations. An access to *any* EDMA bus location is equivalent to an access to DRR of the corresponding McBSP.

**Figure 21-42. Data Receive Register (DRR)**



LEGEND: R = Read only; -n = value after reset

**Table 21-23. Data Receive Register (DRR) Field Descriptions**

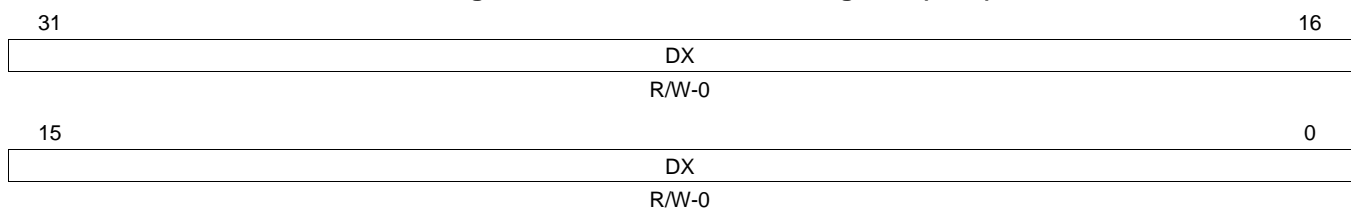
Bit	Field	Value	Description
31-0	DR	0-FFFF FFFFh	Data receive register value to be written to the data bus.

### 21.3.2 Data Transmit Register (DXR)

The data transmit register (DXR) contains the value to be loaded into the data transmit shift register (XSR). The DXR is shown in [Figure 21-43](#) and described in [Table 21-24](#).

See the device-specific data manual for the memory address of these registers. DXR is accessible via the peripheral bus and via the EDMA bus. Both the CPUs and the EDMA can access DXR in all the memory-mapped locations.

**Figure 21-43. Data Transmit Register (DXR)**



LEGEND: R/W = Read/ Write; -n = value after reset

**Table 21-24. Data Transmit Register (DXR) Field Descriptions**

Bit	Field	Value	Description
31-0	DX	0-FFFF FFFFh	Data transmit register value to be loaded into the data transmit shift register (XSR).



### 21.3.3 Serial Port Control Register (SPCR)

The serial port is configured via the serial port control register (SPCR) and the pin control register (PCR). The SPCR contains McBSP status control bits. The SPCR is shown in Figure 21-44 and described in Table 21-25.

**Figure 21-44. Serial Port Control Register (SPCR)**

31				26				25	24
Reserved							FREE	SOFT	
R-0							R/W-0	R/W-0	
23	22	21	20	19	18	17	16		
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST		
R/W-0	R/W-0	R/W-0		R/W-0	R-0	R-0	R/W-0		
15	14	13	12	11	10	8			
DLB	RJUST		CLKSTP		Reserved				
R/W-0	R/W-0		R-0		R-0				
7	6	5	4	3	2	1	0		
DXENA	Reserved	RINTM		RSYNCERR	RFULL	RRDY	RRST		
R/W-0	R-0	R/W-0		R/W-0	R-0	R-0	R/W-0		

LEGEND: R = Read only; R/W = Read/ Write; -n = value after reset

**Table 21-25. Serial Port Control Register (SPCR) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
25	FREE	0 1	Free-running enable mode bit. This bit is used in conjunction with SOFT bit to determine state of serial port clock during emulation halt. 0 Free-running mode is disabled. During emulation halt, SOFT bit determines operation of McBSP. 1 Free-running mode is enabled. During emulation halt, serial clocks continue to run.
24	SOFT	0 1	Soft bit enable mode bit. This bit is used in conjunction with FREE bit to determine state of serial port clock during emulation halt. This bit has no effect if FREE = 1. 0 Soft mode is disabled. Serial port clock stops immediately during emulation halt, thus aborting any transmissions. 1 Soft mode is enabled. During emulation halt, serial port clock stops after completion of current transmission.
23	FRST	0 1	Frame-sync generator reset bit. 0 Frame-synchronization logic is reset. Frame-sync signal (FSG) is not generated by the sample-rate generator. 1 Frame-sync signal (FSG) is generated after (FPER + 1) number of CLKG clocks; that is, all frame counters are loaded with their programmed values.
22	GRST	0 1	Sample-rate generator reset bit. 0 Sample-rate generator is reset. 1 Sample-rate generator is taken out of reset. CLKG is driven as per programmed value in sample-rate generator register (SRGR).
21-20	XINTM	0-3h 0 1h 2h 3h	Transmit interrupt (XINT) mode bit. 0 XINT is driven by XRDY (end-of-word). 1h Reserved 2h XINT is generated by a new frame synchronization. 3h XINT is generated by XSYNCERR.
19	XSYNCERR	0 1	Transmit synchronization error bit. Writing a 1 to XSYNCERR sets the error condition when the transmitter is enabled (XRST = 1). Thus, it is used mainly for testing purposes or if this operation is desired. 0 No synchronization error is detected. 1 Synchronization error is detected.

**Table 21-25. Serial Port Control Register (SPCR) Field Descriptions (continued)**

Bit	Field	Value	Description
18	XEMPTY	0 1	Transmit shift register empty bit. XSR is empty. XSR is not empty.
17	XRDY	0 1	Transmitter ready bit. Transmitter is not ready. Transmitter is ready for new data in DXR.
16	XRST	0 1	Transmitter reset bit resets or enables the transmitter. Serial port transmitter is disabled and in reset state. Serial port transmitter is enabled.
15	DLB	0 1	Digital loop back mode enable bit. Digital loop back mode is disabled. Digital loop back mode is enabled.
14-13	RJUST	0-3h 0 1h 2h 3h	Receive sign-extension and justification mode bit. Right-justify and zero-fill MSBs in DRR. Right-justify and sign-extend MSBs in DRR. Left-justify and zero-fill LSBs in DRR. Reserved
12-11	CLKSTP	0-3h 0 1h-3h	Clock stop mode bit. Clock stop mode is disabled. Normal clocking for non-SPI mode. Reserved
10-8	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
7	DXENA	0 1	DX enabler bit. See <a href="#">Section 21.2.9.4</a> for details on the DX enabler bit. DX enabler is off. DX enabler is on.
6	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
5-4	RINTM	0-3h 0 1h 2h 3h	Receive interrupt (RINT) mode bit. RINT is driven by RRDY (end-of-word). Reserved RINT is generated by a new frame synchronization. RINT is generated by RSYNCERR.
3	RSYNCERR	0 1	Receive synchronization error bit. Writing a 1 to RSYNCERR sets the error condition when the receiver is enabled (RRST = 1). Thus, it is used mainly for testing purposes or if this operation is desired. No synchronization error is detected. Synchronization error is detected.
2	RFULL	0 1	Receive shift register full bit. RBR is not in overrun condition. DRR is not read, RBR is full, and RSR is also full with new word.
1	RRDY	0 1	Receiver ready bit. Receiver is not ready. Receiver is ready with data to be read from DRR.
0	RRST	0 1	Receiver reset bit resets or enables the receiver. The serial port receiver is disabled and in reset state. The serial port receiver is enabled.

### 21.3.4 Receive Control Register (RCR)

The receive control register (RCR) configures parameters of the receive operations. The RCR is shown in [Figure 21-45](#) and described in [Table 21-26](#).

**Figure 21-45. Receive Control Register (RCR)**

31	30	24	23	21	20	19	18	17	16
RPHASE	RFRLLEN2		RWDLEN2		RCOMPAND		RFIG	RDATDLY	
R/W-0	R/W-0		R/W-0		R/W-0		R/W-0	R/W-0	
15	14	8	7	5	4	3	0		
Reserved		RFRLLEN1		RWDLEN1		RWDREVRS		Reserved	
R-0		R/W-0		R/W-0		R/W-0		R-0	

LEGEND: R = Read only; R/ W = Read/Write; -n = value after reset

**Table 21-26. Receive Control Register (RCR) Field Descriptions**

Bit	Field	Value	Description
31	RPHASE	0 1	Receive phases bit. Single-phase frame Dual-phase frame
30-24	RFRLLEN2	0-7Fh 0 1h 2h ... 7Fh	Specifies the receive frame length (number of words) in phase 2. 1 word in phase 2 2 words in phase 2 3 words in phase 2 ... 128 words in phase 2
23-21	RWDLEN2	0-7h 0 1h 2h 3h 4h 5h 6h-7h	Specifies the receive word length (number of bits) in phase 2. Receive word length is 8 bits. Receive word length is 12 bits. Receive word length is 16 bits. Receive word length is 20 bits. Receive word length is 24 bits. Receive word length is 32 bits. Reserved
20-19	RCOMPAND	0-3h 0 1h 2h 3h	Receive companding mode bit. Modes other than 00 are only enabled when RWDLEN1/2 bit is 000 (indicating 8-bit data). No companding, data transfer starts with MSB first. No companding, 8-bit data transfer starts with LSB first. Compand using $\mu$ -law for receive data. Compand using A-law for receive data.
18	RFIG	0 1	Receive frame ignore bit. Receive frame-synchronization pulses after the first pulse restarts the transfer. Receive frame-synchronization pulses after the first pulse are ignored.
17-16	RDATDLY	0-3h 0 1h 2h 3h	Receive data delay bit. 0-bit data delay 1-bit data delay 2-bit data delay Reserved
15	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

**Table 21-26. Receive Control Register (RCR) Field Descriptions (continued)**

Bit	Field	Value	Description
14-8	RFRLLEN1	0-7Fh 0 1h 2h ... 7Fh	Specifies the receive frame length (number of words) in phase 1. 1 word in phase 1 2 words in phase 1 3 words in phase 1 ... 128 words in phase 1
7-5	RWDLEN1	0-7h 0 1h 2h 3h 4h 5h 6h-7h	Specifies the receive word length (number of bits) in phase 1. Receive word length is 8 bits. Receive word length is 12 bits. Receive word length is 16 bits. Receive word length is 20 bits. Receive word length is 24 bits. Receive word length is 32 bits. Reserved
4	RWDREVRS	0 1	Receive 32-bit bit reversal enable bit. 32-bit bit reversal is disabled. 32-bit bit reversal is enabled. 32-bit data is received LSB first. RWDLEN1/2 bit should be set to 5h (32-bit operation); RCOMPAND bit should be set to 1h (transfer starts with LSB first); otherwise, operation is undefined.
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

### 21.3.5 Transmit Control Register (XCR)

The transmit control register (XCR) configures parameters of the transmit operations. The XCR is shown in [Figure 21-46](#) and described in [Table 21-27](#).

**Figure 21-46. Transmit Control Register (XCR)**

31	30	24	23	21	20	19	18	17	16
XPHASE	XFRLEN2		XWDLEN2		XCOMPAND		XFIG	XDATDLY	
R/W-0	R/W-0		R/W-0		R/W-0		R/W-0	R/W-0	
15	14	8	7	5	4	3	0		
Reserved		XFRLEN1		XWDLEN1		XWDREVRVS		Reserved	
R-0		R/W-0		R/W-0		R/W-0		R-0	

LEGEND: R = Read only; R/ W = Read/Write; -n = value after reset

**Table 21-27. Transmit Control Register (XCR) Field Descriptions**

Bit	Field	Value	Description
31	XPHASE	0 1	Transmit phases bit. Single-phase frame Dual-phase frame
30-24	XFRLEN2	0-7Fh 0 1h 2h ... 7Fh	Specifies the transmit frame length (number of words) in phase 2. 1 word in phase 2 2 words in phase 2 3 words in phase 2 ... 128 words in phase 2
23-21	XWDLEN2	0-7h 0 1h 2h 3h 4h 5h 6h-7h	Specifies the transmit word length (number of bits) in phase 2. Transmit word length is 8 bits. Transmit word length is 12 bits. Transmit word length is 16 bits. Transmit word length is 20 bits. Transmit word length is 24 bits. Transmit word length is 32 bits. Reserved
20-19	XCOMPAND	0-3h 0 1h 2h 3h	Transmit companding mode bit. Modes other than 00 are only enabled when XWDLEN1/2 bit is 000 (indicating 8-bit data). No companding, data transfer starts with MSB first. No companding, 8-bit data transfer starts with LSB first. Compand using $\mu$ -law for transmit data. Compand using A-law for transmit data.
18	XFIG	0 1	Transmit frame ignore bit. Transmit frame-synchronization pulses after the first pulse restarts the transfer. Transmit frame-synchronization pulses after the first pulse are ignored.
17-16	XDATDLY	0-3h 0 1h 2h 3h	Transmit data delay bit. 0-bit data delay 1-bit data delay 2-bit data delay Reserved
15	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

**Table 21-27. Transmit Control Register (XCR) Field Descriptions (continued)**

Bit	Field	Value	Description
14-8	XFRLEN1	0-7Fh	Specifies the transmit frame length (number of words) in phase 1.
		0	1 word in phase 1
		1h	2 words in phase 1
		2h	3 words in phase 1
		...	...
		7Fh	128 words in phase 1
7-5	XWDLEN1	0-7h	Specifies the transmit word length (number of bits) in phase 1.
		0	Transmit word length is 8 bits.
		1h	Transmit word length is 12 bits.
		2h	Transmit word length is 16 bits.
		3h	Transmit word length is 20 bits.
		4h	Transmit word length is 24 bits.
		5h	Transmit word length is 32 bits.
		6h-7h	Reserved
4	XWDREVRS		Transmit 32-bit bit reversal feature enable bit.
		0	32-bit bit reversal is disabled.
		1	32-bit bit reversal is enabled. 32-bit data is transmitted LSB first. XWDLEN1/2 bit should be set to 5h (32-bit operation); XCOMPAND bit should be set to 1h (transfer starts with LSB first); otherwise, operation is undefined.
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

### 21.3.6 Sample Rate Generator Register (SRGR)

The sample rate generator register (SRGR) controls the operation of various features of the sample rate generator. The SRGR is shown in Figure 21-47 and described in Table 21-28.

**Figure 21-47. Sample Rate Generator Register (SRGR)**

31	30	29	28	27	16
GSYNC	CLKSP	CLKSM	FSGM	FPER	
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	
			8	7	0
FWID			CLKGDV		
R/W-0			R/W-1		

LEGEND: R/W = Read/Write; -n = value after reset

**Table 21-28. Sample Rate Generator Register (SRGR) Field Descriptions**

Bit	Field	Value	Description																		
31	GSYNC	0 1	<p>Sample-rate generator clock synchronization bit is only used when the external clock (CLKS) drives the sample-rate generator clock (CLKSM = 0).</p> <p>0 The sample-rate generator clock (CLKG) is free running.</p> <p>1 The sample-rate generator clock (CLKG) is running; however, CLKG is resynchronized and frame-sync signal (FSG) is generated only after detecting the receive frame-synchronization signal (FSR). Also, frame period (FPER) is a don't care because the period is dictated by the external frame-sync pulse.</p>																		
30	CLKSP	0 1	<p>CLKS polarity clock edge select bit is only used when the external clock (CLKS) drives the sample-rate generator clock (CLKSM = 0).</p> <p>0 Rising edge of CLKS generates CLKG and FSG.</p> <p>1 Falling edge of CLKS generates CLKG and FSG.</p>																		
29	CLKSM	0 1	<p>Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:</p> $CLKG \text{ frequency} = \text{Input clock frequency} / (\text{CLKGDV} + 1)$ <p>CLKSM is used in conjunction with the SCLKME bit in the pin control register (PCR) to determine the source for the input clock.</p> <p>A DSP reset selects the McBSP internal input clock as the input clock and forces the CLKG frequency to 1/2 the McBSP internal input clock frequency.</p> <p>0 The input clock for the sample rate generator is taken from the CLKS pin or from the CLKR pin, depending on the value of the SCLKME bit in PCR:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>CLKSM</th> <th>SCLKME</th> <th>Input Clock for Sample Rate Generator</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td>Signal on CLKS pin</td> </tr> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td>Signal on CLKR pin</td> </tr> </tbody> </table> <p>1 The input clock for the sample rate generator is taken from the McBSP internal input clock or from the CLKX pin, depending on the value of the SCLKME bit in PCR:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>CLKSM</th> <th>SCLKME</th> <th>Input Clock for Sample Rate Generator</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td>McBSP internal input clock</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td>Signal on CLKX pin</td> </tr> </tbody> </table>	CLKSM	SCLKME	Input Clock for Sample Rate Generator	0	0	Signal on CLKS pin	0	1	Signal on CLKR pin	CLKSM	SCLKME	Input Clock for Sample Rate Generator	1	0	McBSP internal input clock	1	1	Signal on CLKX pin
CLKSM	SCLKME	Input Clock for Sample Rate Generator																			
0	0	Signal on CLKS pin																			
0	1	Signal on CLKR pin																			
CLKSM	SCLKME	Input Clock for Sample Rate Generator																			
1	0	McBSP internal input clock																			
1	1	Signal on CLKX pin																			
28	FSGM	0 1	<p>Sample-rate generator transmit frame-synchronization mode bit is only used when FSXM = 1 in PCR.</p> <p>0 Transmit frame-sync signal (FSX) is generated on every DXR-to-XSR copy. When FSGM = 0, FWID bit and FPER bit are ignored.</p> <p>1 Transmit frame-sync signal (FSX) is driven by the sample-rate generator frame-sync signal (FSG).</p>																		
27-16	FPER	0-FFFh	Frame period value plus 1 specifies when the next frame-sync signal becomes active. Range is 1 to 4096 sample-rate generator clock (CLKG) periods.																		
15-8	FWID	0-FFh	Frame width value plus 1 specifies the width of the frame-sync pulse (FSG) during its active period.																		

**Table 21-28. Sample Rate Generator Register (SRGR) Field Descriptions (continued)**

Bit	Field	Value	Description
7-0	CLKGDV	0-FFh	Sample-rate generator clock (CLKG) divider value is used as the divide-down number to generate the required sample-rate generator clock frequency.

### 21.3.7 Multichannel Control Register (MCR)

The multichannel control register (MCR) has control and status bits for multichannel selection operation in the receiver (with an R prefix) and the same type of bits for the transmitter (with an X prefix). The MCR is shown in [Figure 21-48](#) and described in [Table 21-29](#). This I/O-mapped register enables you to:

- Enable all channels or only selected channels for reception (RMCM).
- Choose which channels are enabled/disabled and masked/unmasked for transmission (XMCM).
- Specify whether two partitions (32 channels at a time) or eight partitions (128 channels at a time) can be used (RMCME for reception, XMCME for transmission).
- Assign blocks of 16 channels to partitions A and B when the 2-partition mode is selected (RPABLK and RPBBLK for reception, XPABLK and XPBBLK for transmission).
- Determine which block of 16 channels is currently involved in a data transfer (RCBLK for reception, XCBLK for transmission).

**Figure 21-48. Multichannel Control Registers (MCR)**

31	26	25	24	23	22	21	20	18	17	16
Reserved		XMCME	XPBBLK	XPABLK	XCBLK	XMCM				
R-0		R/W-0	R/W-0	R/W-0	R-0	R/W-0				
15	10	9	8	7	6	5	4	2	1	0
Reserved		RMCME	RPBBLK	RPABLK	RCBLK	Reserved		RMCM		
R-0		R/W-0	R/W-0	R/W-0	R-0	R-0		R/W-0		

LEGEND: R = Read only; R/ W = Read/Write; -n = value after reset

**Table 21-29. Multichannel Control Register (MCR) Field Descriptions**

Bit	Field	Value	Description
31-26	Reserved	0	Reserved.
25	XMCME	0	Transmit multichannel partition mode bit. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). XMCME determines whether only 32 channels or all 128 channels are to be individually selectable. 2-partition mode. Only partitions A and B are used. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bit. <b>If XMCM = 1 or 2h:</b> assign 16 channels to partition A with the XPABLK bit and assign 16 channels to partition B with the XPBBLK bit. <b>If XMCM = 3h:</b> (for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bit and assign 16 channels to receive partition B with the RPBBLK bit.
		1	8-partition mode. All partitions (A through H) are used. You can control up to 128 channels in the transmit multichannel selection mode selected with the XMCM bit. You control the channels with the appropriate enhanced transmit channel enable register (XCEREn): XCERE0: Channels 0 through 31 XCERE1: Channels 32 through 63 XCERE2: Channels 64 through 95 XCERE3: Channels 96 through 127



**Table 21-29. Multichannel Control Register (MCR) Field Descriptions (continued)**

Bit	Field	Value	Description
24-23	XPBBLK	0-3h	<p>Transmit partition B block bit.</p> <p>XPBBLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter.</p> <p>The 128 transmit channels of the McBSP are divided equally among 8 blocks (0 through 7). When XPBBLK is applicable, use the XPBBLK bit to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B; use the XPABLK bit to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.</p> <p>If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the transmitter is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The XCBLK bit is regularly updated to indicate which block is active.</p> <p>When XMCM = 3h (for symmetric transmission and reception), the transmitter uses the receive block bits (RPABLK and RPBBLK) rather than the transmit block bits (XPABLK and XPBBLK).</p> <p>0 Block 1: channels 16 through 31</p> <p>1h Block 3: channels 48 through 63</p> <p>2h Block 5: channels 80 through 95</p> <p>3h Block 7: channels 112 through 127</p>
22-21	XPABLK	0-3h	<p>Transmit partition A block bit.</p> <p>XPABLK is only applicable if channels can be individually disabled/enabled and masked/unmasked (XMCM is nonzero) and the 2-partition mode is selected (XMCME = 0). Under these conditions, the McBSP transmitter can transmit or withhold data in any of the 32 channels that are assigned to partitions A and B of the transmitter. See the XPBBLK bit description for more information about assigning blocks to partitions A and B.</p> <p>0 Block 0: channels 0 through 15</p> <p>1h Block 2: channels 32 through 47</p> <p>2h Block 4: channels 64 through 79</p> <p>3h Block 6: channels 96 through 111</p>
20-18	XCBLK	0-7h	<p>Transmit current block indicator. XCBLK indicates which block of 16 channels is involved in the current McBSP transmission.</p> <p>0 Block 0: channels 0 through 15</p> <p>1h Block 1: channels 16 through 31</p> <p>2h Block 2: channels 32 through 47</p> <p>3h Block 3: channels 48 through 63</p> <p>4h Block 4: channels 64 through 79</p> <p>5h Block 5: channels 80 through 95</p> <p>6h Block 6: channels 96 through 111</p> <p>7h Block 7: channels 112 through 127</p>

**Table 21-29. Multichannel Control Register (MCR) Field Descriptions (continued)**

Bit	Field	Value	Description
17-16	XMCM	0-3h 0 1h 2h 3h	<p>Transmit multichannel selection mode bit. XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission.</p> <p>0 Transmit multichannel selection is off. All channels are enabled and unmasked. No channels can be disabled or masked.</p> <p>1h All channels are disabled unless they are selected in the appropriate enhanced transmit channel enable register (XCEREn). If enabled, a channel in this mode is also unmasked. The XMCM bit determines whether 32 channels or 128 channels are selectable in XCEREn.</p> <p>2h All channels are enabled, but they are masked unless they are selected in the appropriate enhanced transmit channel enable register (XCEREn). The XMCM bit determines whether 32 channels or 128 channels are selectable in XCEREn.</p> <p>3h This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate enhanced receive channel enable register (RCEREn). Once enabled, they are masked unless they are also selected in the appropriate enhanced transmit channel enable register (XCEREn). The XMCM bit determines whether 32 channels or 128 channels are selectable in RCEREn and XCEREn.</p>
15-10	Reserved	0	Reserved.
9	RMCME	0 1	<p>Receive multichannel partition mode bit. RMCME is only applicable if channels can be individually disabled/enabled for reception (RMCM = 1). RMCME determines whether only 32 channels or all 128 channels are to be individually selectable.</p> <p>0 2-partition mode. Only partitions A and B are used. You can control up to 32 channels in the receive multichannel selection mode (RMCM = 1). Assign 16 channels to partition A with the RPBLK bit and assign 16 channels to partition B with the RPBBLK bit. You control the channels with the enhanced receive channel enable register partition A/B (RCERE0).</p> <p>1 You can control up to 128 channels in the receive multichannel selection mode. You control the channels with the appropriate enhanced receive channel enable register (RCEREn): RCERE0: Channels 0 through 31 RCERE1: Channels 32 through 63 RCERE2: Channels 64 through 95 RCERE3: Channels 96 through 127</p>
8-7	RPBBLK	0-3h 0 1h 2h 3h	<p>Receive partition B block bit.</p> <p>RPBBLK is only applicable if channels can be individually disabled/enabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver.</p> <p>The 128 receive channels of the McBSP are divided equally among 8 blocks (0 through 7). When RPBBLK is applicable, use the RPBBLK bit to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B; use the RPBLK bit to assign one of the even-numbered blocks (0, 2, 4, or 6) to partition A.</p> <p>If you want to use more than 32 channels, you can change block assignments dynamically. You can assign a new block to one partition while the receiver is handling activity in the other partition. For example, while the block in partition A is active, you can change which block is assigned to partition B. The RCBLLK bit is regularly updated to indicate which block is active.</p> <p>When XMCM = 3h (for symmetric transmission and reception), the transmitter uses the receive block bits (RPBLK and RPBBLK) rather than the transmit block bits (XPBLK and XPBBLK).</p> <p>0 Block 1: channels 16 through 31 1h Block 3: channels 48 through 63 2h Block 5: channels 80 through 95 3h Block 7: channels 112 through 127</p>

**Table 21-29. Multichannel Control Register (MCR) Field Descriptions (continued)**

Bit	Field	Value	Description
6-5	RPABLK	0-3h 0 1h 2h 3h	<p>Receive partition A block bit.</p> <p>RPABLK is only applicable if channels can be individually disabled/enabled (RMCM = 1) and the 2-partition mode is selected (RMCME = 0). Under these conditions, the McBSP receiver can accept or ignore data in any of the 32 channels that are assigned to partitions A and B of the receiver. See the RPBBLK bit description for more information about assigning blocks to partitions A and B.</p> <p>Block 0: channels 0 through 15                      Block 2: channels 32 through 47                      Block 4: channels 64 through 79                      Block 6: channels 96 through 111</p>
4-2	RCBLK	0-7h 0 1h 2h 3h 4h 5h 6h 7h	<p>Receive current block indicator. RCBLK indicates which block of 16 channels is involved in the current McBSP reception.</p> <p>Block 0: channels 0 through 15                      Block 1: channels 16 through 31                      Block 2: channels 32 through 47                      Block 3: channels 48 through 63                      Block 4: channels 64 through 79                      Block 5: channels 80 through 95                      Block 6: channels 96 through 111                      Block 7: channels 112 through 127</p>
1	Reserved	0	Reserved.
0	RMCM	0 1	<p>Receive multichannel selection mode bit. RMCM determines whether all channels or only selected channels are enabled for reception.</p> <p>0 All 128 channels are enabled.                      1 Multichannel selection mode. Channels can be individually enabled or disabled.</p> <p>The only channels enabled are those selected in the appropriate enhanced receive channel enable register (RCEREn). The way channels are assigned to RCEREn depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.</p>

### 21.3.8 Enhanced Receive Channel Enable Registers (RCERE0-RCERE3)

The enhanced receive channel enable register (RCEREn) is shown in [Figure 21-49](#) and described in [Table 21-30](#). The RCEREn is used to enable any of 128 elements for receive. RCERE0 is the only register used in normal mode (up to 32 channels can be selected in partitions A and B, RMCME = XMCME = 0 in MCR). RCERE0-RCERE3 are used when in enhanced mode (up to 128 channels can be selected in all partitions, RMCME = XMCME = 1 in MCR).

The receive multichannel partition mode (RMCME) bit in the multichannel control register (MCR) is only applicable if channels can be individually disabled/enabled for reception (RMCM = 1). The RMCME bit determines whether only 32 channels or all 128 channels are to be individually selectable:

- **When RMCME = 0:** Only partitions A and B are used. RCERE0 is used to enable any of the 32 elements for a receive. Of the 32 elements, 16 channels belong to a subframe in partition A and 16 channels belong to a subframe in partition B. The RCE0-RCE15 bits enable elements within the 16-channel elements in partition A and the RCE16-RCE31 bits enable elements within the 16-channel elements in partition B. The 16 channels in partition A are assigned with the RPABLK bit in MCR and the 16 channels in partition B are assigned with the RPBBLK bit in MCR.
- **When RMCME = 1:** All partitions are used. RCERE0 is used to enable any of the 32 elements in channels 0 through 31 for a receive. Of the 32 elements, channels 0 to 15 belong to a subframe in partition A and channels 16 to 31 belong to a subframe in partition B. The RCE0-RCE15 bits enable elements within the 16-channel elements in partition A and the RCE16-RCE31 bits enable elements within the 16-channel elements in partition B.

[Section 21.3.8.1](#) shows the 128 channels in a multichannel data stream and their corresponding enable bits in RCEREn.

**Figure 21-49. Enhanced Receive Channel Enable Register n (RCEREn)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCE31	RCE30	RCE29	RCE28	RCE27	RCE26	RCE25	RCE24	RCE23	RCE22	RCE21	RCE20	RCE19	RCE18	RCE17	RCE16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCE15	RCE14	RCE13	RCE12	RCE11	RCE10	RCE9	RCE8	RCE7	RCE6	RCE5	RCE4	RCE3	RCE2	RCE1	RCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 21-30. Enhanced Receive Channel Enable Register n (RCEREn) Field Descriptions**

Bit	Field	Value	Description
31-0	RCERn	0	Receive channel enable bit. For receive multichannel selection mode (RMCM = 1 in MCR). Disable the channel that is mapped to RCERn.
		1	Enable the channel that is mapped to RCERn.

### 21.3.8.1 RCEREN Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the enhanced receive channel enable register (RCEREN) depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit in the multichannel control register (MCR). For each of these two cases, Table 21-31 shows which block of channels is assigned to each RCEREN used. For each RCEREN, Table 21-31 shows which channel is assigned to each of the bits.

**Table 21-31. Use of the Receive Channel Enable Registers**

Number of selectable channels	Block Assignments		Channel Assignments		
	RCEREN	Block assigned <sup>(1)</sup>	Bit in RCEREN	Channel assigned <sup>(1)</sup>	
32 (RMCME = 0)	RCERE0	Channels $n$ to $(n + 15)$	RCE0	Channel $n$	
		The block of channels (0, 2, 4, or 6) is selected with the RPABLK bit in MCR	RCE1	Channel $(n + 1)$	
			...	...	
			RCE15	Channel $(n + 15)$	
		Channels $m$ to $(m + 15)$	The block of channels (1, 3, 5, or 7) is selected with the RPBBLK bit in MCR	RCE16	Channel $m$
				RCE17	Channel $(m + 1)$
				...	...
	RCE31	Channel $(m + 15)$			
	128 (RMCME = 1)	RCERE0	Block 0	RCE0	Channel 0
				...	...
RCE15			Channel 15		
Block 1			RCE16	Channel 16	
		...	...		
RCE31		Channel 31			
RCERE1		Block 2	RCE0	Channel 32	
			...	...	
		RCE15	Channel 47		
		Block 3	RCE16	Channel 48	
...			...		
RCE31		Channel 63			
RCERE2		Block 4	RCE0	Channel 64	
			...	...	
	RCE15	Channel 79			
	Block 5	RCE16	Channel 80		
...		...			
RCE31	Channel 95				
RCERE3	Block 6	RCE0	Channel 96		
		...	...		
	RCE15	Channel 111			
	Block 7	RCE16	Channel 112		
...		...			
RCE31	Channel 127				

<sup>(1)</sup>  $n$  is any even-numbered block 0, 2, 4, or 6.  $m$  is any odd-numbered block 1, 3, 5, or 7.

### 21.3.9 Enhanced Transmit Channel Enable Registers (XCERE0-XCERE3)

The enhanced transmit channel enable register (XCEREn) is shown in [Figure 21-50](#) and described in [Table 21-32](#). The XCEREn is used to enable any of 128 elements for transmit. XCERE0 is the only register used in normal mode (up to 32 channels can be selected in partitions A and B, RMCME = XMCME = 0 in MCR). XCERE0-XCERE3 are used when in enhanced mode (up to 128 channels can be selected in all partitions, RMCME = XMCME = 1 in MCR).

The transmit multichannel partition mode (XMCME) bit in the multichannel control register (MCR) is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). The XMCME bit determines whether only 32 channels or all 128 channels are to be individually selectable:

- **When XMCME = 0:** Only partitions A and B are used. XCERE0 is used to enable any of the 32 elements for a transmit. Of the 32 elements, 16 channels belong to a subframe in partition A and 16 channels belong to a subframe in partition B. The XCE0-XCE15 bits enable elements within the 16-channel elements in partition A and the XCE16-XCE31 bits enable elements within the 16-channel elements in partition B. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bit in MCR.
- **When XMCME = 1:** All partitions are used. XCERE0 is used to enable any of the 32 elements in channels 0 through 31 for a transmit. Of the 32 elements, channels 0 to 15 belong to a subframe in partition A and channels 16 to 31 belong to a subframe in partition B. The XCE0-XCE15 bits enable elements within the 16-channel elements in partition A and the XCE16-XCE31 bits enable elements within the 16-channel elements in partition B.

[Section 21.3.9.1](#) shows the 128 channels in a multichannel data stream and their corresponding enable bits in XCEREn.

**Figure 21-50. Enhanced Transmit Channel Enable Register n (XCEREn)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XCE31	XCE30	XCE29	XCE28	XCE27	XCE26	XCE25	XCE24	XCE23	XCE22	XCE21	XCE20	XCE19	XCE18	XCE17	XCE16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCE15	XCE14	XCE13	XCE12	XCE11	XCE10	XCE9	XCE8	XCE7	XCE6	XCE5	XCE4	XCE3	XCE2	XCE1	XCE0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

**Table 21-32. Enhanced Transmit Channel Enable Register n (XCEREn) Field Descriptions**

Bit	Field	Value	Description
31-0	XCERn		Transmit channel enable bit. The role of this bit depends on which transmit multichannel mode is selected with the XMCM bit in MCR:
			<b>When XMCM = 1h (all channels disabled unless selected):</b>
		0	Disable and mask the channel that is mapped to XCERn.
		1	Enable and unmask the channel that is mapped to XCERn.
			<b>When XMCM = 2h (all channels enabled but masked unless selected):</b>
		0	Mask the channel that is mapped to XCERn.
	<b>When XMCM = 3h (all channels masked unless selected):</b>		
	0	Mask the channel that is mapped to XCERn. Even if this channel is enabled by the corresponding enhanced receive channel enable bit, this channel's data cannot appear on the DX pin.	
	1	Unmask the channel that is mapped to XCERn. Even if this channel is also enabled by the corresponding enhanced receive channel enable bit, full transmission can occur.	

### 21.3.9.1 XCEREn Used in the Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the enhanced transmit channel enable register (XCEREn) depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit in the multichannel control register (MCR). For each of these two cases, [Table 21-33](#) shows which block of channels is assigned to each XCEREn used. For each XCEREn, [Table 21-33](#) shows which channel is assigned to each of the bits.

When XMCM = 3h (symmetric transmission and reception), the transmitter uses the enhanced receive channel enable register (RCEREn) to enable channels and uses XCEREn to unmask channels for transmission.

**Table 21-33. Use of the Transmit Channel Enable Registers**

Number of selectable channels	Block Assignments		Channel Assignments			
	XCEREn	Block assigned <sup>(1)</sup>	Bit in XCEREn	Channel assigned <sup>(1)</sup>		
32 (XMCME = 0)	XCERE0	Channels $n$ to $(n + 15)$	XCE0	Channel $n$		
		When XMCM = 1h or 2h, the block of channels (0, 2, 4, or 6) is selected with the XPABLK bit in MCR. When XMCM = 3h, the block of channels (0, 2, 4, or 6) is selected with the RPABLK bit in MCR.	XCE1	Channel $(n + 1)$		
			...	...		
		XCE15	Channel $(n + 15)$			
		Channels $m$ to $(m + 15)$	XCE16	Channel $m$		
		When XMCM = 1h or 2h, the block of channels (1, 3, 5, or 7) is selected with the XPBBLK bit in MCR. When XMCM = 3h, the block of channels (1, 3, 5, or 7) is selected with the RPBBLK bit in MCR.	XCE17	Channel $(m + 1)$		
			...	...		
		XCE31	Channel $(m + 15)$			
		128 (XMCME = 1)	XCERE0	Block 0	XCE0	Channel 0
					...	...
XCE15	Channel 15					
XCE16	Channel 16					
...	...					
XCE31	Channel 31					
XCERE1	Block 2		XCE0	Channel 32		
			...	...		
			XCE15	Channel 47		
			XCE16	Channel 48		
			...	...		
			XCE31	Channel 63		
XCERE2	Block 4		XCE0	Channel 64		
			...	...		
			XCE15	Channel 79		
			XCE16	Channel 80		
			...	...		
			XCE31	Channel 95		

<sup>(1)</sup>  $n$  is any even-numbered block 0, 2, 4, or 6.  $m$  is any odd-numbered block 1, 3, 5, or 7.

**Table 21-33. Use of the Transmit Channel Enable Registers (continued)**

Number of selectable channels	Block Assignments		Channel Assignments	
	XCERE <sub>n</sub>	Block assigned <sup>(1)</sup>	Bit in XCERE <sub>n</sub>	Channel assigned <sup>(1)</sup>
	XCERE3	Block 6	XCE0	Channel 96
			...	...
			XCE15	Channel 111
		Block 7	XCE16	Channel 112
			...	...
			XCE31	Channel 127

**21.3.10 Pin Control Register (PCR)**

The serial port is configured via the serial port control register (SPCR) and the pin control register (PCR). The PCR contains McBSP status control bits. The PCR is shown in [Figure 21-51](#) and described in [Table 21-34](#).

**Figure 21-51. Pin Control Register (PCR)**

Reserved							
R-0							
31							16
15	14	13	12	11	10	9	8
Reserved	Reserved <sup>(1)</sup>	Reserved <sup>(1)</sup>	FSXM	FSRM	CLKXM	CLKRM	
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SCLKME	Reserved <sup>(1)</sup>	Reserved	Reserved	FSXP	FSRP	CLKXP	CLKRP
R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

<sup>(1)</sup> If writing to this field, always write the default value of 0 to ensure proper McBSP operation.

**Table 21-34. Pin Control Register (PCR) Field Descriptions**

Bit	Field	Value	Description
31-14	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
13-12	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper McBSP operation.
11	FSXM	0 1	Transmit frame-synchronization mode bit. 0 Frame-synchronization signal is derived from an external source. 1 Frame-synchronization signal is determined by FSGM bit in SRGR.
10	FSRM	0 1	Receive frame-synchronization mode bit. 0 Frame-synchronization signal is derived from an external source. FSR is an input pin. 1 Frame-synchronization signal is generated internally by the sample-rate generator. FSR is an output pin.
9	CLKXM	0 1	Transmit clock mode bit. When CLKSTP bit in SPCR is cleared to 0: 0 CLKX is an input pin and is driven by an external clock. 1 CLKX is an output pin and is driven by the internal sample-rate generator.



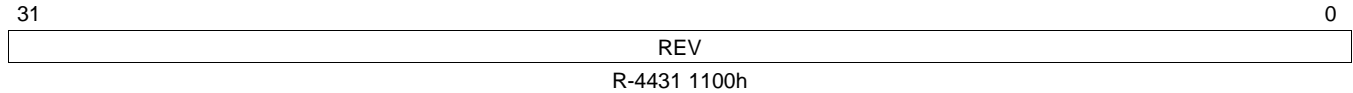
**Table 21-34. Pin Control Register (PCR) Field Descriptions (continued)**

Bit	Field	Value	Description									
8	CLKRM		Receive clock mode bit.									
		0	<b>Digital loop back mode is disabled (DLB = 0 in SPCR):</b> CLKR is an input pin and is driven by an external clock.									
		1	CLKR is an output pin and is driven by the internal sample-rate generator.									
			<b>Digital loop back mode is enabled (DLB = 1 in SPCR):</b>									
		0	Receive clock (not the CLKR pin) is driven by transmit clock (CLKX) that is based on CLKXM bit. CLKR pin is in high-impedance state.									
		1	CLKR is an output pin and is driven by the transmit clock. The transmit clock is based on CLKXM bit.									
7	SCLKME		Sample rate generator input clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is:  <i>CLKG frequency = Input clock frequency / (CLKGDV + 1)</i>  SCLKME is used in conjunction with the CLKSM bit in the sample rate generator register (SRGR) to select the input clock.  A DSP reset selects the McBSP internal input clock as the input clock and forces the CLKG frequency to 1/2 the McBSP internal input clock frequency.									
		0	The input clock for the sample rate generator is taken from the CLKS pin or from the McBSP internal input clock, depending on the value of the CLKSM bit in SRGR:  <table border="0"> <thead> <tr> <th>SCLKME</th> <th>CLKSM</th> <th>Input Clock for Sample Rate Generator</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Signal on CLKS pin</td> </tr> <tr> <td>0</td> <td>1</td> <td>McBSP internal input clock</td> </tr> </tbody> </table>	SCLKME	CLKSM	Input Clock for Sample Rate Generator	0	0	Signal on CLKS pin	0	1	McBSP internal input clock
		SCLKME	CLKSM	Input Clock for Sample Rate Generator								
		0	0	Signal on CLKS pin								
0	1	McBSP internal input clock										
1	The input clock for the sample rate generator is taken from the CLKR pin or from the CLKX pin, depending on the value of the CLKSM bit in SRGR:  <table border="0"> <thead> <tr> <th>SCLKME</th> <th>CLKSM</th> <th>Input Clock for Sample Rate Generator</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>0</td> <td>Signal on CLKR pin</td> </tr> <tr> <td>1</td> <td>1</td> <td>Signal on CLKX pin</td> </tr> </tbody> </table>	SCLKME	CLKSM	Input Clock for Sample Rate Generator	1	0	Signal on CLKR pin	1	1	Signal on CLKX pin		
SCLKME	CLKSM	Input Clock for Sample Rate Generator										
1	0	Signal on CLKR pin										
1	1	Signal on CLKX pin										
6	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper McBSP operation.									
5-4	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.									
3	FSXP		Transmit frame-synchronization polarity bit.									
		0	Transmit frame-synchronization pulse is active high.									
		1	Transmit frame-synchronization pulse is active low.									
2	FSRP		Receive frame-synchronization polarity bit.									
		0	Receive frame-synchronization pulse is active high.									
		1	Receive frame-synchronization pulse is active low.									
1	CLKXP		Transmit clock polarity bit.									
		0	Transmit data driven on rising edge of CLKX.									
		1	Transmit data driven on falling edge of CLKX.									
0	CLKRP		Receive clock polarity bit.									
		0	Receive data sampled on falling edge of CLKR.									
		1	Receive data sampled on rising edge of CLKR.									

### 21.3.11 BFIFO Revision Identification Register (BFIFOREV)

The Buffer FIFO (BFIFO) revision identification register (BFIFOREV) contains revision data for the Buffer FIFO (BFIFO). The BFIFOREV is shown in [Figure 21-52](#) and described in [Table 21-35](#).

**Figure 21-52. BFIFO Revision Identification Register (BFIFOREV)**



LEGEND: R = Read only; -n = value after reset

**Table 21-35. BFIFO Revision Identification Register (BFIFOREV) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4431 1100h	Identifies revision of Buffer FIFO.

### 21.3.12 Write FIFO Control Register (WFIFOCTL)

The Write FIFO control register (WFIFOCTL) is shown in [Figure 21-53](#) and described in [Table 21-36](#).

**NOTE:** The WNUMEVT and WNUMDMA values must be set prior to enabling the Write FIFO. If the Write FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.

**Figure 21-53. Write FIFO Control Register (WFIFOCTL)**

31	Reserved		17	16
	R-0			WENA
				R/W-0
15	8	7		0
	WNUMEVT		WNUMDMA	
	R/W-10h		R/W-4h	

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

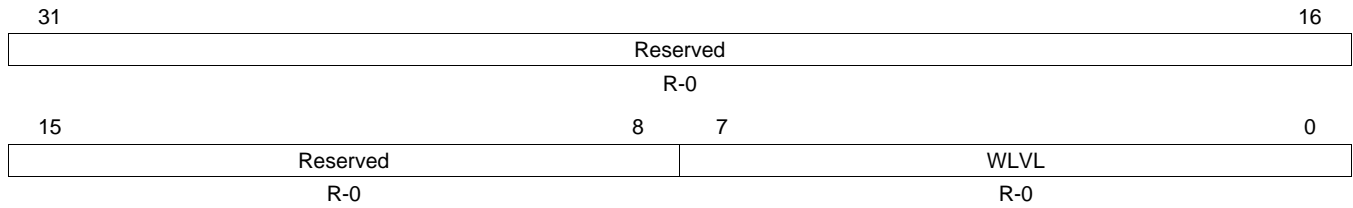
**Table 21-36. Write FIFO Control Register (WFIFOCTL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	WENA	0	Write FIFO is disabled. The WLVL bit in the Write FIFO status register (WFIFOSTS) is reset to 0 and the pointers are initialized, that is, the Write FIFO is “flushed.”
		1	Write FIFO is enabled. If the Write FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.
15-8	WNUMEVT	0-FFh	Write word count per DMA event (32-bit). When the Write FIFO has space for at least <i>WNUMEVT</i> words of data, then an XEVT (transmit DMA event) is generated to the host/DMA controller. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		...	...
		40h	64 words
		41h-FFh	Reserved
7-0	WNUMDMA	0-FFh	Write word count per transfer (32-bit words). Upon a transmit DMA event from the McBSP, <i>WNUMDMA</i> words are transferred from the Write FIFO to the McBSP. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1	1 word
		2h-FFh	Reserved

### 21.3.13 Write FIFO Status Register (WFIFOSTS)

The Write FIFO status register (WFIFOSTS) is shown in [Figure 21-54](#) and described in [Table 21-37](#).

**Figure 21-54. Write FIFO Status Register (WFIFOSTS)**



LEGEND: R = Read only; -n = value after reset

**Table 21-37. Write FIFO Status Register (WFIFOSTS) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	WLVL	0-FFh	Write level. Number of 32-bit words currently in the Write FIFO.
		0	0 words currently in the Write FIFO.
		1h	1 word currently in the Write FIFO.
		2h	2 words currently in the Write FIFO.
		...	...
		40h	64 words currently in the Write FIFO.
		41h-FFh	Reserved

### 21.3.14 Read FIFO Control Register (RFIFOCTL)

The Read FIFO control register (RFIFOCTL) is shown in [Figure 21-55](#) and described in [Table 21-38](#).

**NOTE:** The RNUMEVT and RNUMDMA values must be set prior to enabling the Read FIFO. If the Read FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.

**Figure 21-55. Read FIFO Control Register (RFIFOCTL)**

31	Reserved	17	16
	R-0		RENA
			R/W-0
15	RNUMEVT	8	7
	R/W-10h		RNUMDMA
			R/W-4h
			0

LEGEND: R = Read only; R/W = Read/Write; -n = value after reset

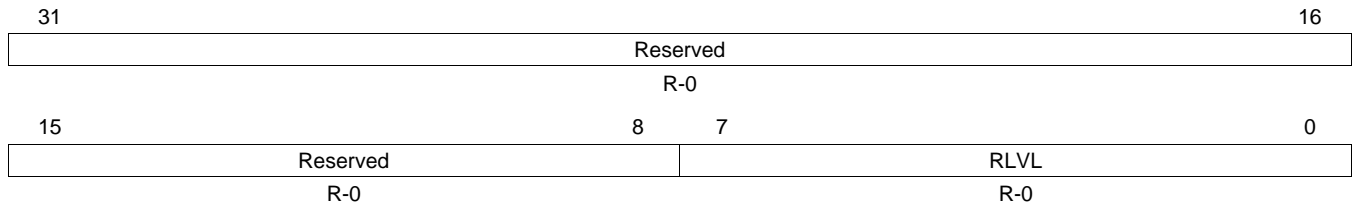
**Table 21-38. Read FIFO Control Register (RFIFOCTL) Field Descriptions**

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	RENA	0	Read FIFO enable bit. Read FIFO is disabled. The RLVL bit in the Read FIFO status register (RFIFOSTS) is reset to 0 and the pointers are initialized, that is, the Read FIFO is “flushed.”
		1	Read FIFO is enabled. If the Read FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.
15-8	RNUMEVT	0-FFh	Read word count per DMA event (32-bit). When the Read FIFO contains at least <i>RNUMEVT</i> words of data, then an REVT (receive DMA event) is generated to the host/DMA controller. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		...	...
		40h	64 words
		41h-FFh	Reserved
7-0	RNUMDMA	0-FFh	Read word count per transfer (32-bit words). Upon a receive DMA event from the McBSP, the Read FIFO reads <i>RNUMDMA</i> words from the McBSP. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1	1 word
		2h-FFh	Reserved

### 21.3.15 Read FIFO Status Register (RFIFOSTS)

The Read FIFO status register (RFIFOSTS) is shown in [Figure 21-56](#) and described in [Table 21-39](#).

**Figure 21-56. Read FIFO Status Register (RFIFOSTS)**



LEGEND: R = Read only; -n = value after reset

**Table 21-39. Read FIFO Status Register (RFIFOSTS) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RLVL	0-FFh	Read level. Number of 32-bit words currently in the Read FIFO.
		0	0 words currently in the Read FIFO.
		1h	1 word currently in the Read FIFO.
		2h	2 words currently in the Read FIFO.
		...	...
		40h	64 words currently in the Read FIFO.
		41h-FFh	Reserved

## ***Real-Time Clock (RTC)***

---

---

This chapter describes the real-time clock (RTC).

<b>Topic</b>	<b>Page</b>
<b>22.1 Introduction</b> .....	<b>965</b>
<b>22.2 Architecture</b> .....	<b>966</b>
<b>22.3 Registers</b> .....	<b>972</b>

## 22.1 Introduction

### 22.1.1 Purpose of the Peripheral

The real-time clock (RTC) provides a time reference to an application running on the device. The current date and time is tracked in a set of counter registers that update once per second. The time can be represented in 12-hour or 24-hour mode. The calendar and time registers are buffered during reads and writes so that updates do not interfere with the accuracy of the time and date.

Alarms are available to interrupt the CPU at a particular time, or at periodic time intervals, such as once per minute or once per day. In addition, the RTC can interrupt the CPU every time the calendar and time registers are updated, or at programmable periodic intervals.

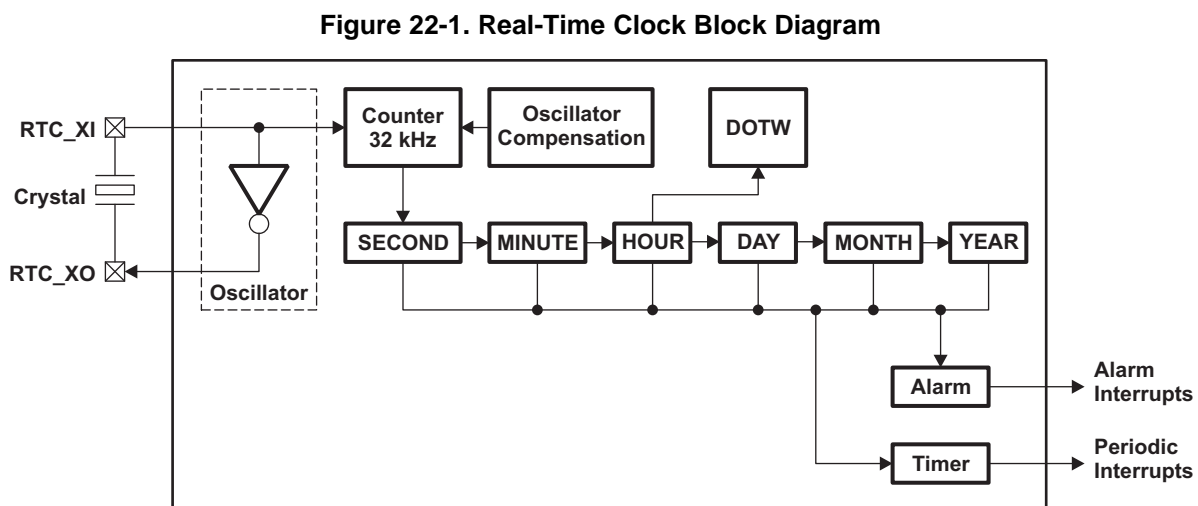
### 22.1.2 Features

The real-time clock (RTC) provides the following features:

- 100-year calendar (xx00 to xx99)
- Counts seconds, minutes, hours, day of the week, date, month, and year with leap year compensation
- Binary-coded-decimal (BCD) representation of time, calendar, and alarm
- 12-hour clock mode (with AM and PM) or 24-hour clock mode
- Alarm interrupt
- Periodic interrupt
- Single interrupt to the CPU
- Supports external 32.768-kHz crystal or external clock source of the same frequency
- Isolated power supply

### 22.1.3 Block Diagram

Figure 22-1 shows a block diagram of the RTC.





## 22.2 Architecture

### 22.2.1 Clock Source

The clock reference for the RTC is an external 32.768-kHz crystal or an external clock source of the same frequency. The RTC also has a separate power supply that is isolated from the rest of the system. When the CPU and other peripherals are without power, the RTC can remain powered to preserve the current time and calendar information.

The source for the RTC reference clock may be provided by a crystal or by an external clock source. The RTC has an internal oscillator buffer to support direct operation with a crystal. The crystal is connected between pins RTC\_XI and RTC\_XO. RTC\_XI is the input to the on-chip oscillator and RTC\_XO is the output from the oscillator back to the crystal. For more information about the RTC crystal connection, see your device-specific data manual.

An external 32.768-kHz clock source may be used instead of a crystal. In such a case, the clock source is connected to RTC\_XI, and RTC\_XO is left unconnected.

If the RTC is not used, the RTC\_XI pin should be held low and RTC\_XO should be left unconnected. The RTCDISABLE bit in the control register (CTRL) can be set to save power; however, the RTCDISABLE bit should not be cleared once it has been set. If the application requires the RTC module to stop and continue, the RUN bit in CTRL should be used instead.

### 22.2.2 Signal Descriptions

The RTC signals are listed in [Table 22-1](#).

**Table 22-1. Real-Time Clock Signals**

Signal	I/O	Description
RTC_XI	I	RTC time base input signal. RTC_XI can either be driven with a 32.768-kHz reference clock, or RTC_XI and RTC_XO can be connected to an external crystal. This signal is the input to the RTC internal oscillator.
RTC_XO	O	RTC time base output signal. RTC_XO is the output from the RTC internal oscillator. If a crystal is not used as the time base for RTC_XI, RTC_XO should be left unconnected.

### 22.2.3 Isolated Power Supply

The RTC has a power supply that is isolated from the rest of the system. This allows the RTC to continue to run while the rest of the system is not powered. In this state, the RTC time and calendar counters continue to run, but the powered down CPU is not able to receive RTC interrupts. Separate power supply pins for the RTC are provided on the device package.

#### 22.2.3.1 Split-Power Circuitry

To decrease power consumption, RTC includes leakage-isolation circuitry that is activated by setting the SPLITPOWER bit in the control register (CTRL). Because of its isolated power supply, RTC does not have a power-on hardware reset signal. Therefore, upon initial device power-on, the RTC is in an unknown state until it has been properly configured. After the RTC module has been configured once, it functions as programmed as long as its power supply and clock source are provided.

#### 22.2.3.2 Power Considerations

The RTC leakage-isolation circuitry requires that the CPU supply be powered down to VSS when the RTC is powered on while the rest of the device is powered off. A floating CPU supply creates undesired RTC leakage current. Also, the RTC power consumption is higher when the CPU is powered on versus the RTC power consumption when the CPU is powered off. Therefore, if the RTC module is expected to run from a small-capacity power supply (ex. watch battery) while the rest of the device is powered off, a power system should be implemented such that the RTC is powered from a high-capacity power supply when the CPU is powered on.

## 22.2.4 Operation

### 22.2.4.1 Using the Real-Time Clock Time and Calendar Registers

The current time and date are maintained in the RTC time and calendar registers.

#### 22.2.4.1.1 Time/Calendar Data Format

The time and calendar data in the RTC is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Although most of the time/calendar registers have 4 bits assigned to each BCD digit, some of the register fields are shorter since the range of valid numbers may be limited. For example, only 3 bits are required to represent the day since only BCD numbers 1 through 7 are required.

The following time and calendar registers are supported (BCD Format):

- SECOND - Second Count (00-59)
- MINUTE - Minute Count (00-59)
- HOUR - Hour Count (12HR: 01-12; 24HR: 00-23)
- DAY - Day of the Month Count (01-31)
- MONTH - Month Count (01-12; JAN = 1)
- YEAR - Year Count (00-99)
- DOTW - Day of the Week Count (0-6; SUN = 0)

Note that the ALARM registers which share the names above also share the same BCD formatting.

#### 22.2.4.1.2 12-Hour and 24-Hour Modes

The current time can be represented in 12-hour or 24-hour mode by configuring the HOURMODE bit in the control register (CTRL):

- When HOURMODE = 0, 24-hour mode is selected. The hours are represented as 00 through 23. The MERIDIEM bit in the HOURS register has no function and should be cleared.
- When HOURMODE = 1, 12-hour mode is selected. The hours are represented as 00 through 12. MERIDIEM = 0 indicates ante meridiem (AM), and MERIDIEM = 1 indicates post meridiem (PM).

#### 22.2.4.1.3 Reading from Time/Calendar Registers

The time/calendar registers are updated every second as the time changes. During a read of the SECOND register, the RTC copies the current values of the time/date registers into shadow read registers. This isolation assures that the CPU can capture all the time/date values at the moment of the SECOND read request and not be subject to changing register values from time updates.

If desired, the RTC also provides a one-time-triggered minute-rounding feature to round the MINUTE:SECOND registers to the nearest minute (with zero seconds). This feature is enabled by setting the ROUNDMIN bit in the control register (CTRL); the RTC automatically rounds the time values to the nearest minute upon the next read of the SECOND register.

#### 22.2.4.1.4 Writing to Time/Calendar Registers

When setting the RTC time and date, values are written directly to the time/calendar registers. Therefore, care must be taken to avoid writing to the time/calendar registers while the time is updating to the next second. This can be accomplished in one of two ways:

1. The RTC can be stopped by clearing the RUN bit in the control register (CTRL). When stopped, there is no danger of contention during writes because the registers do not auto-update.
2. The BUSY bit in the status register (STATUS) is low when a time update does not take place for at least 15  $\mu$ s. By checking for a low BUSY bit before writing to registers, the CPU is assured of a 15  $\mu$ s window of time during which multiple accesses to the time/calendar registers can be performed.

After writing to a time/calendar register, the RTC requires four peripheral clock cycles to update the register value. Any reads that take place within four peripheral clock cycles of a write returns old data.

Note that all registers in the RTC except for KICK $n$ R have write-protection. See [Section 22.2.6](#) for information on unlocking registers.

#### 22.2.4.2 Real-Time Clock Update Cycle

The RTC executes an update cycle once per second to update the current time in the time/calendar registers. The update cycle also compares each alarm register with the corresponding time register. These comparisons are done to determine when to trigger an alarm. The BUSY bit in the status register (STATUS) provides a mechanism to indicate when the time/calendar registers are updated. When the BUSY bit is high, an update takes place within 15  $\mu$ s. When BUSY returns low again, the update has been completed.

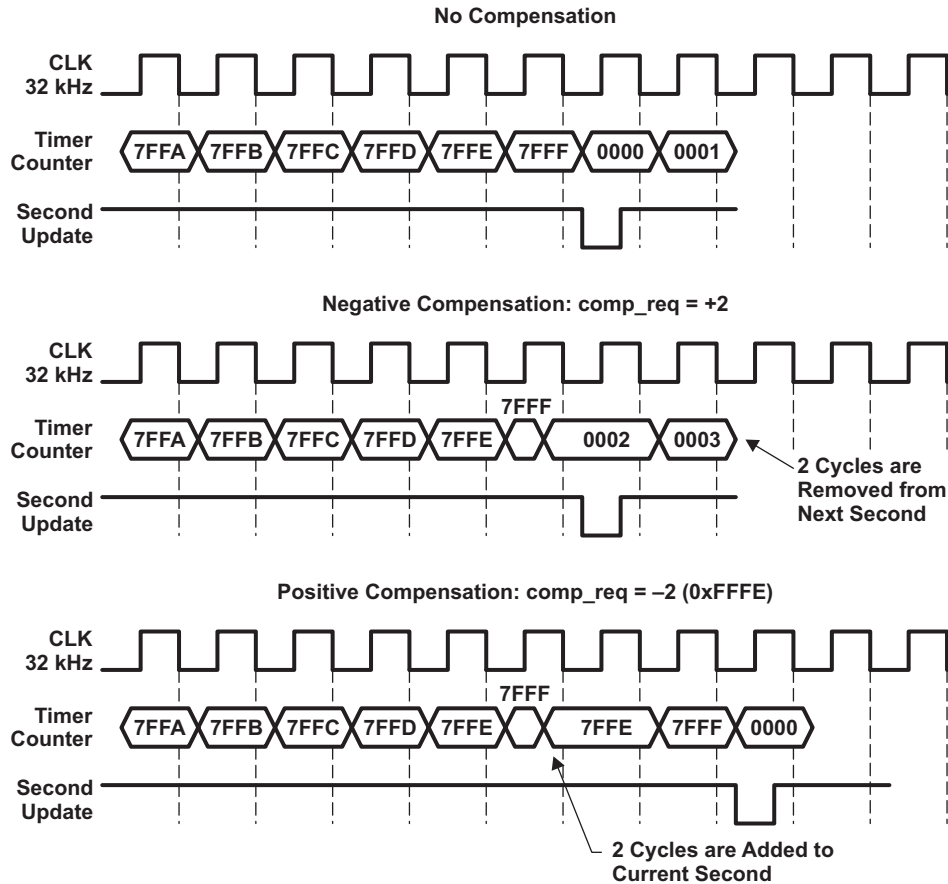
The BUSY bit should be checked when writing to any of the following registers while RTC is running:

- SECOND
- MINUTE
- HOUR
- DAY
- MONTH
- YEAR
- DOTW
- ALARMSECOND (when ALARM interrupt is enabled)
- ALARMMINUTE (when ALARM interrupt is enabled)
- ALARMHOUR (when ALARM interrupt is enabled)
- ALARMDAY (when ALARM interrupt is enabled)
- ALARMMONTH (when ALARM interrupt is enabled)
- ALARMYEAR (when ALARM interrupt is enabled)
- CTRL (SET32COUNTER field only -- the other fields in CTRL do not require BUSY to be low)
- INTERRUPT
- COMPLSB (when oscillator drift compensation is enabled)
- COMPMSB (when oscillator drift compensation is enabled)

#### 22.2.4.3 Oscillator Drift Compensation

If the RTC 32.768-kHz reference clock is susceptible to oscillator drift, the RTC provides the ability to compensate the update cycle by subtracting oscillator periods. The COMPMSB and COMPLSB registers hold the number of two's complement reference periods to subtract from the update cycle every hour. For example, [Figure 22-2](#) shows how programming the value of 2h into the compensation registers shortens the update cycle by two 32.786-kHz reference periods every hour. [Figure 22-2](#) also shows how programming the value of FFFEh (decimal negative 2) into the compensation register lengthens the update cycle by two reference periods every hour. To enable the oscillator compensation, the AUTOCOMP bit in the control register (CTRL) must be set.

**Figure 22-2. 32-kHz Oscillator Counter Compensation**



### 22.2.5 Interrupt Requests

The RTC provides the ability to interrupt the CPU based on two events: a periodic interrupt and an alarm interrupt. Although two interrupt sources are available, the RTC makes a single interrupt request to the CPU.

When the device is initially powered on, the RTC may issue spurious interrupt signals to the CPU. To avoid issues, a software reset should be performed on the RTC module before the CPU interrupt controller is initialized. See [Section 22.2.10](#) for more information on reset considerations.

#### 22.2.5.1 Alarm Interrupt Enable and Status Bits

The ALARM bit in the interrupt register (INTERRUPT) enables the alarm interrupt. When the current time and date match the ALARMSECOND, ALARMMINUTE, ALARMHOUR, ALARMDAY, ALARMMONTH, and ALARMYEAR registers, the RTC issues an interrupt to the CPU and sets the ALARM bit in the status register (STATUS). Once set, the ALARM status bit stays high until cleared by a write of 1 to the ALARM bit.

As with writing to time and calendar registers ([Section 22.2.4.1.4](#)), writes to the INTERRUPT and STATUS registers should only be done when the RTC is stopped or when the BUSY bit is low.

Note that all registers in the RTC except for KICKnR have write-protection. See [Section 22.2.6](#) for information on unlocking registers.

### 22.2.5.2 Periodic Interrupt Enable and Status Bits

The TIMER bit and EVERY field in the interrupt register (INTERRUPT) work together to enable periodic interrupts. When the TIMER bit is enabled, interrupts are issued at a time period indicated by the EVERY field (0 = Second, 1h = Minute, 2h = Hour, 3h = Day). Regardless of the period selected in the EVERY field, the periodic timer status bits (DAYEVT, HREVT, MINEVT, SECEVT) are set in the status register (STATUS) whenever they are valid. Note that the appropriate status bits are set when the TIME bit is enabled, not when the desired interrupt is generated. Active periodic status bits remain high as long as the TIMER bit is enabled.

For example, if daily periodic interrupts are enabled and the time (in HH:MM:SS format) transitions from 23:59:59 to 00:00:00, the STATUS register sets all four periodic status bits (DAYEVT, HREVT, MINEVT, and SECEVT) because all four time periods were incremented. These bits all remain high until:

1. The TIME bit is cleared and all four status bits clear to zero until TIME is set again **OR**
2. The current time reaches 00:00:01. At that point, the SECEVT remains set while the DAYEVT, HREVT, and MINEVT bits are cleared. The next interrupt is not generated until the next day transition.

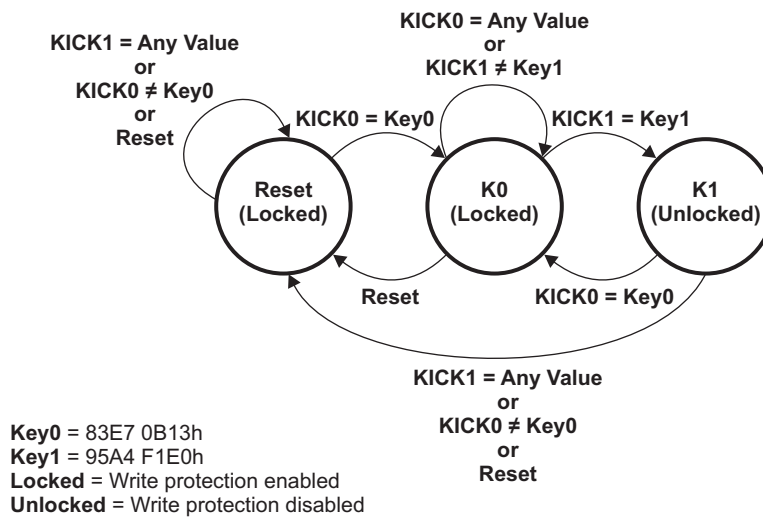
As with writing to time and calendar registers (Section 22.2.4.1.4), writes to the INTERRUPT and STATUS registers should only be done when the RTC is stopped or when the BUSY bit is low.

Note that all registers in the RTC except for KICK $n$ R have write-protection. See Section 22.2.6 for information on unlocking registers.

### 22.2.6 Register Protection Against Spurious Writes

All registers in the RTC except for the KICK $n$ R registers are protected from spurious writes. Out of reset, writes to protected registers are disabled until the registers are unlocked using the KICK $n$ R registers. To unlock the registers, a key of 83E7 0B13h needs to be written to KICK0R, followed by a write of 95A4 F1E0h to KICK1R. Registers remain unlocked until write protection is enabled again by writing any value to KICK0R or KICK1R. The write protection state machine is shown in Figure 22-3.

Figure 22-3. Kick State Machine



### 22.2.7 General-Purpose Scratch Registers

The RTC provides three general-purpose registers (SCRATCH $n$ ) that can be used to store 32-bit words -- these registers have no functional purpose for the RTC. Software using the RTC may find the SCRATCH $n$  registers to be useful in indicating RTC states. For example, the SCRATCH $n$  registers may be used to indicate write-protection lock status or unintentional power downs.

To indicate write-protection, the software should write a unique value to one of the SCRATCH $n$  registers when write-protection is disabled and another unique value when write-protection is enabled again. In this way, the lock-status of the registers can be determined quickly by reading the SCRATCH register.

To indicate unintentional power downs, the software should write a unique value to one of the SCRATCH $n$  registers when RTC is configured and enabled. If the RTC is unintentionally powered down, the value written to the SCRATCH register is cleared.

### 22.2.8 Real-Time Clock Response to Low Power Modes (Idle Configurations)

The device is divided into idle domains that can be programmed to be idle or active. The state of all domains is called the idle configuration. The RTC runs on its own external clock source and is not affected by any of the other device idle domains.

### 22.2.9 Emulation Modes of the Real-Time Clock

The RTC always continues to run regardless of the state (running/halted) of the emulation debugger software.

### 22.2.10 Reset Considerations

When the device is initially powered on, the RTC may issue spurious interrupt signals to the CPU. To avoid issues, a software reset should be performed on the RTC module before the CPU interrupt controller is initialized.

As the RTC is configured, the SPLITPOWER bit in the control register (CTRL) should be set.

A software reset is performed on the RTC by setting the SWRESET bit in the oscillator register (OSC). The software reset applies to all registers except the oscillator (OSC) and kick (KICK $n$ R) registers. The RTC requires three 32.768-kHz reference clocks to pass before RTC registers can be accessed.

## 22.3 Registers

Table 22-2 lists the memory-mapped registers for the RTC. See your device-specific data manual for the memory address of these registers.

**Table 22-2. Real-Time Clock (RTC) Registers**

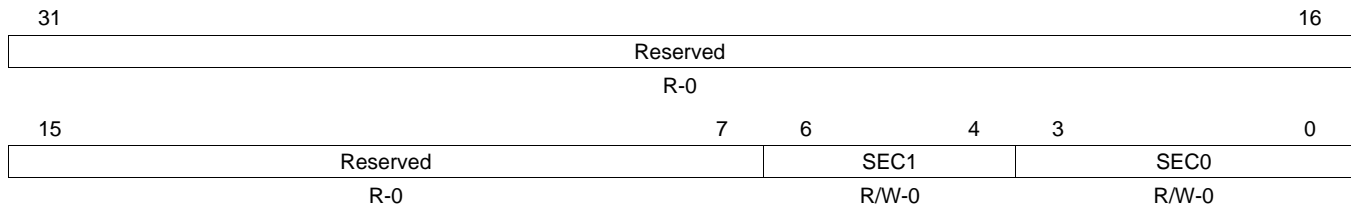
Address Offset	Acronym	Register Description	Section
0h	SECOND	Seconds Register	<a href="#">Section 22.3.1</a>
4h	MINUTE	Minutes Register	<a href="#">Section 22.3.2</a>
8h	HOUR	Hours Register	<a href="#">Section 22.3.3</a>
Ch	DAY	Day of the Month Register	<a href="#">Section 22.3.4</a>
10h	MONTH	Month Register	<a href="#">Section 22.3.5</a>
14h	YEAR	Year Register	<a href="#">Section 22.3.6</a>
18h	DOTW	Day of the Week Register	<a href="#">Section 22.3.7</a>
20h	ALARMSECOND	Alarm Seconds Register	<a href="#">Section 22.3.8</a>
24h	ALARMMINUTE	Alarm Minutes Register	<a href="#">Section 22.3.9</a>
28h	ALARMHOUR	Alarm Hours Register	<a href="#">Section 22.3.10</a>
2Ch	ALARMDAY	Alarm Days Register	<a href="#">Section 22.3.11</a>
30h	ALARMMONTH	Alarm Months Register	<a href="#">Section 22.3.12</a>
34h	ALARMYEAR	Alarm Years Register	<a href="#">Section 22.3.13</a>
40h	CTRL	Control Register	<a href="#">Section 22.3.14</a>
44h	STATUS	Status Register	<a href="#">Section 22.3.15</a>
48h	INTERRUPT	Interrupt Enable Register	<a href="#">Section 22.3.16</a>
4Ch	COMPLSB	Compensation (LSB) Register	<a href="#">Section 22.3.17</a>
50h	COMPMSB	Compensation (MSB) Register	<a href="#">Section 22.3.18</a>
54h	OSC	Oscillator Register	<a href="#">Section 22.3.19</a>
60h	SCRATCH0	Scratch 0 Register (General-Purpose)	<a href="#">Section 22.3.20</a>
64h	SCRATCH1	Scratch 1 Register (General-Purpose)	<a href="#">Section 22.3.20</a>
68h	SCRATCH2	Scratch 2 Register (General-Purpose)	<a href="#">Section 22.3.20</a>
6Ch	KICK0R	Kick 0 Register (Write Protect)	<a href="#">Section 22.3.21</a>
70h	KICK1R	Kick 1 Register (Write Protect)	<a href="#">Section 22.3.21</a>

### 22.3.1 Second Register (SECOND)

**NOTE:** Out of reset, the second register (SECOND) is write-protected. To disable write protection, correct keys must be written to the KICK<sub>n</sub>R registers (see [Section 22.2.6](#)).

The second register (SECOND) sets the second value of the current time. Seconds are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The SECOND register is shown in [Figure 22-4](#) and described in [Table 22-3](#).

**Figure 22-4. Second Register (SECOND)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-3. Second Register (SECOND) Field Descriptions**

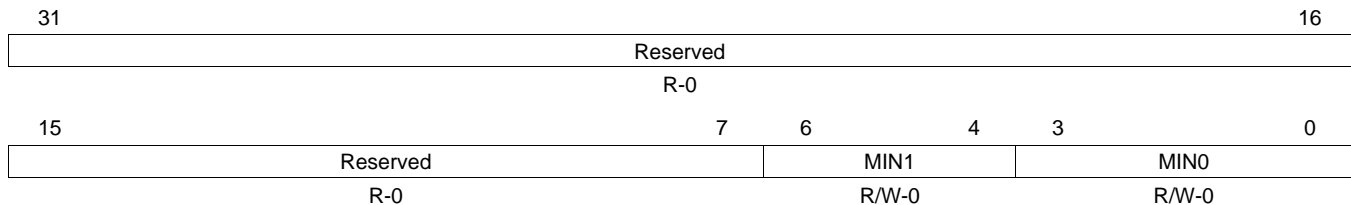
Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	SEC1	0-5h	Most significant digit of second value. Range for SEC1:SEC0 is 00-59.
3-0	SEC0	0-9h	Least significant digit of second value. Range for SEC1:SEC0 is 00-59.

### 22.3.2 Minute Register (MINUTE)

**NOTE:** Out of reset, the minute register (MINUTE) is write-protected. To disable write protection, correct keys must be written to the KICK<sub>n</sub>R registers (see [Section 22.2.6](#)).

The minute register (MINUTE) sets the minute value of the current time. Minutes are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The MINUTE register is shown in [Figure 22-5](#) and described in [Table 22-4](#).

**Figure 22-5. Minute Register (MINUTE)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-4. Minute Register (MINUTE) Field Descriptions**

Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	MIN1	0-5h	Most significant digit of minute value. Range for MIN1:MIN0 is 00-59.
3-0	MIN0	0-9h	Least significant digit of minute value. Range for MIN1:MIN0 is 00-59.

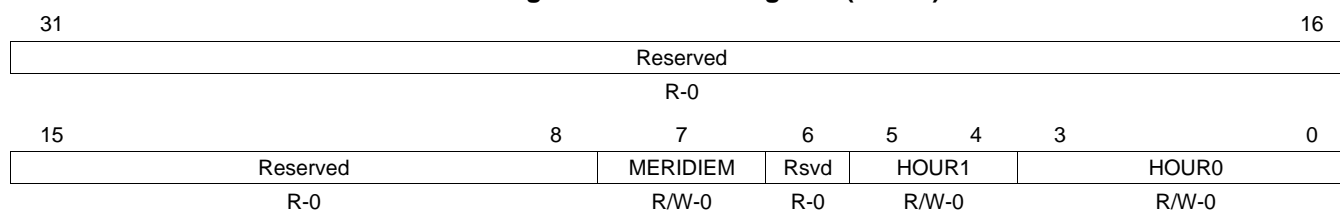


### 22.3.3 Hour Register (HOUR)

**NOTE:** Out of reset, the hour register (HOUR) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The hour register (HOUR) sets the hour value of the current time. Hours are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The HOUR register is shown in [Figure 22-6](#) and described in [Table 22-5](#).

**Figure 22-6. Hour Register (HOUR)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-5. Hour Register (HOUR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	MERIDIEM	0 1	Determines whether the hour is ante meridiem (AM) or post meridiem (PM) when 12-hour mode is enabled. Hour is AM Hour is PM
6	Reserved	0	Reserved.
5-4	HOUR1	0-2h	Most significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.
3-0	HOUR0	0-9h	Least significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.

### 22.3.4 Day of the Month Register (DAY)

**NOTE:** Out of reset, the day of the month register (DAY) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The day of the month register (DAY) sets the day of the month value of the current date. Days are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The DAY register is shown in [Figure 22-7](#) and described in [Table 22-6](#).

**Figure 22-7. Days Register (DAY)**

31	Reserved												16	
R-0														
15	Reserved						6	5	4	3				0
R-0							DAY1			DAY0				
R-0							R/W-0			R/W-1				

LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-6. Day Register (DAY) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5-4	DAY1	0-3h	Most significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.
3-0	DAY0	0-9h	Least significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.

### 22.3.5 Month Register (MONTH)

**NOTE:** Out of reset, the month register (MONTH) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The month register (MONTH) sets the month in the year value of the current date. The month is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The MONTH register is shown in [Figure 22-8](#) and described in [Table 22-7](#).

**Figure 22-8. Month Register (MONTH)**

31	Reserved												16
R-0													
15	Reserved						5	4	3				0
R-0							MONTH1			MONTH0			
R-0							R/W-0			R/W-1			

LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-7. Month Register (MONTH) Field Descriptions**

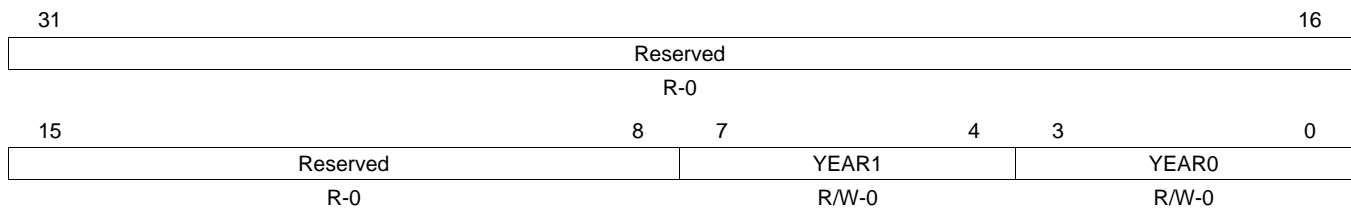
Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4	MONTH1	0-1h	Most significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.
3-0	MONTH0	0-9h	Least significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.

### 22.3.6 Year Register (YEAR)

**NOTE:** Out of reset, the year register (YEAR) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The year register (YEAR) sets the year value of the current date. The year is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The YEAR register is shown in [Figure 22-9](#) and described in [Table 22-8](#).

**Figure 22-9. Year Register (YEAR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-8. Year Register (YEAR) Field Descriptions**

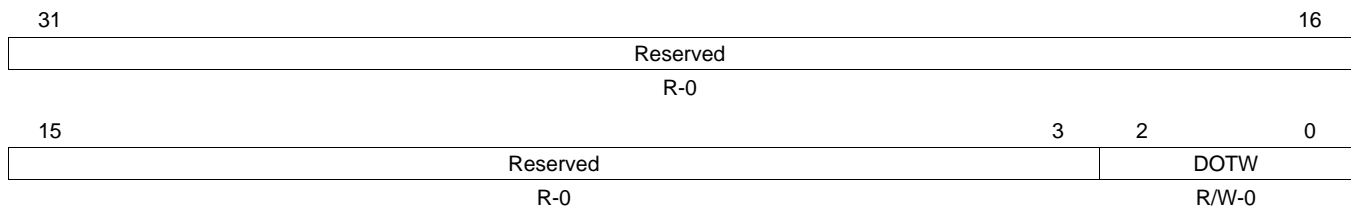
Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-4	YEAR1	0-9h	Most significant digit of years value. Range for YEAR1:YEAR0 is 00-99.
3-0	YEAR0	0-9h	Least significant digit of years value. Range for YEAR1:YEAR0 is 00-99.

### 22.3.7 Day of the Week Register (DOTW)

**NOTE:** Out of reset, the day of the week register (DOTW) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The day of the week register (DOTW) sets the day of the week value of the current date. The day of the week is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The DOTW register is shown in [Figure 22-10](#) and described in [Table 22-9](#).

**Figure 22-10. Day of the Week Register (DOTW)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-9. Day of the Week Register (DOTW) Field Descriptions**

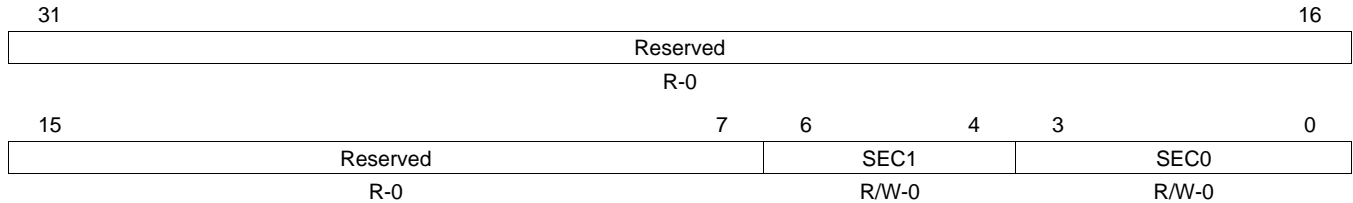
Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2-0	DOTW	0-6h	Day of the week. Sunday = 0, Saturday = 6h.

### 22.3.8 Alarm Second Register (ALARMSECOND)

**NOTE:** Out of reset, the alarm second register (ALARMSECOND) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The alarm second register (ALARMSECOND) sets the second value for the alarm interrupt. Seconds are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMSECOND register is shown in [Figure 22-11](#) and described in [Table 22-10](#).

**Figure 22-11. Alarm Second Register (ALARMSECOND)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-10. Alarm Second Register (ALARMSECOND) Field Descriptions**

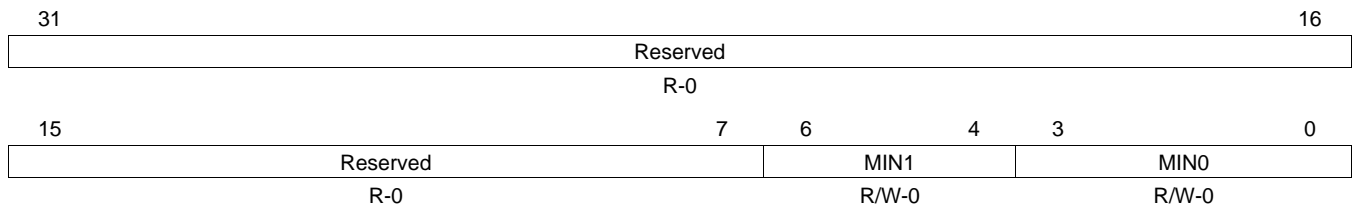
Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	SEC1	0-5h	Most significant digit of alarm seconds value. Range for SEC1:SEC0 is 00-59.
3-0	SEC0	0-9h	Least significant digit of alarm seconds value. Range for SEC1:SEC0 is 00-59.

### 22.3.9 Alarm Minute Register (ALARMMINUTE)

**NOTE:** Out of reset, the alarm minute register (ALARMMINUTE) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The alarm minute register (ALARMMINUTE) sets the minute value for the alarm interrupt. Minutes are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMMINUTE register is shown in [Figure 22-12](#) and described in [Table 22-11](#).

**Figure 22-12. Alarm Minute Register (ALARMMINUTE)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-11. Alarm Minute Register (ALARMMINUTE) Field Descriptions**

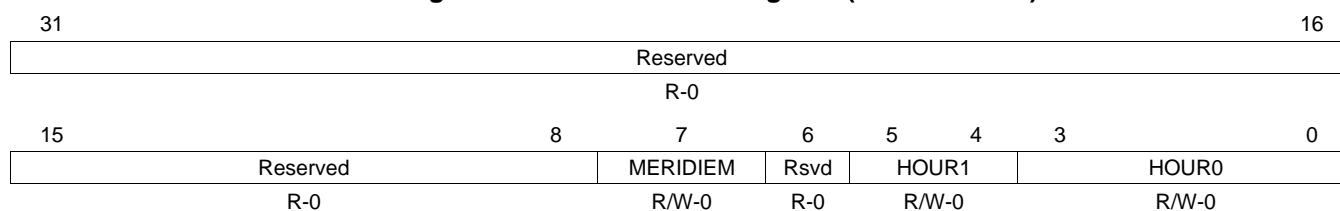
Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	MIN1	0-5h	Most significant digit of alarm minutes value. Range for MIN1:MIN0 is 00-59.
3-0	MIN0	0-9h	Least significant digit of alarm minutes value. Range for MIN1:MIN0 is 00-59.

### 22.3.10 Alarm Hour Register (ALARMHOUR)

**NOTE:** Out of reset, the alarm hour register (ALARMHOUR) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The alarm hour register (ALARMHOUR) sets the hour value for the alarm interrupt. Hours are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMHOUR register is shown in [Figure 22-13](#) and described in [Table 22-12](#).

**Figure 22-13. Alarm Hour Register (ALARMHOUR)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-12. Alarm Hour Register (ALARMHOUR) Field Descriptions**

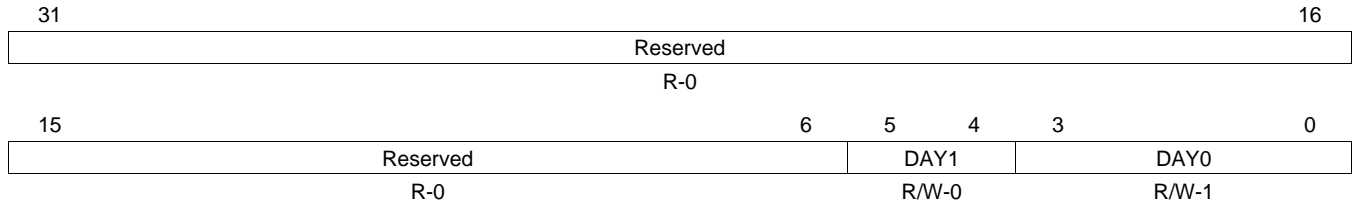
Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	MERIDIEM	0 1	Determines whether the hour is ante meridiem (AM) or post meridiem (PM) when 12-hour mode is enabled. Hour is AM Hour is PM
6	Reserved	0	Reserved.
5-4	HOUR1	0-2h	Most significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.
3-0	HOUR0	0-9h	Least significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.

### 22.3.11 Alarm Day of the Month Register (ALARMDAY)

**NOTE:** Out of reset, the alarm day of the month register (ALARMDAY) is write-protected. To disable write protection, correct keys must be written to the KICK<sub>n</sub>R registers (see [Section 22.2.6](#)).

The alarm day of the month register (ALARMDAY) sets the day of the month value for the alarm interrupt. Days are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMDAYS register is shown in [Figure 22-14](#) and described in [Table 22-13](#).

**Figure 22-14. Alarm Day Register (ALARMDAY)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-13. Alarm Day Register (ALARMDAY) Field Descriptions**

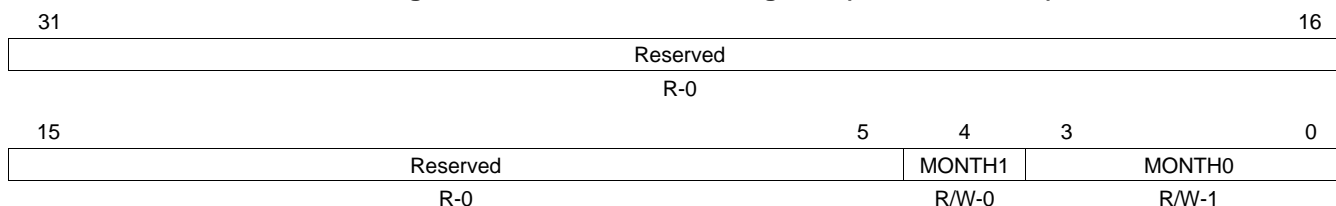
Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5-4	DAY1	0-3h	Most significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.
3-0	DAY0	0-9h	Least significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.

### 22.3.12 Alarm Month Register (ALARMMONTH)

**NOTE:** Out of reset, the alarm month register (ALARMMONTH) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The alarm month register (ALARMMONTH) sets the month in the year value for the alarm interrupt. The month is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMMONTH register is shown in [Figure 22-15](#) and described in [Table 22-14](#).

**Figure 22-15. Alarm Month Register (ALARMMONTH)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-14. Alarm Month Register (ALARMMONTH) Field Descriptions**

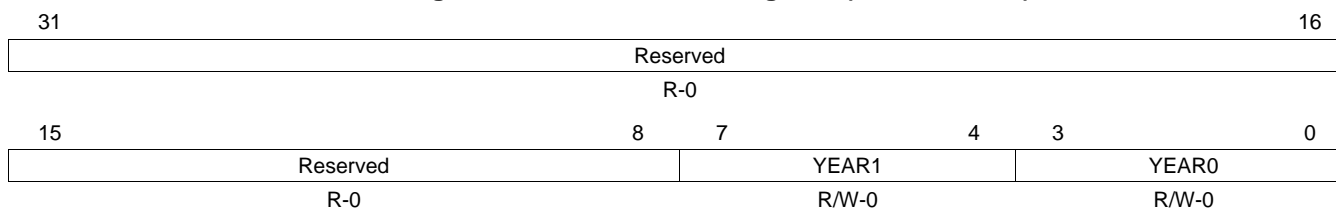
Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4	MONTH1	0-1h	Most significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.
3-0	MONTH0	0-9h	Least significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.

### 22.3.13 Alarm Year Register (ALARMYEAR)

**NOTE:** Out of reset, the alarm year register (ALARMYEAR) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The alarm year register (ALARMYEAR) sets the year for the alarm interrupt. The year is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMYEAR register is shown in [Figure 22-16](#) and described in [Table 22-15](#).

**Figure 22-16. Alarm Year Register (ALARMYEAR)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-15. Alarm Years Register (ALARMYEARS) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-4	YEAR1	0-9h	Most significant digit of years value. Range for YEAR1:YEAR0 is 00-99.
3-0	YEAR0	0-9h	Least significant digit of years value. Range for YEAR1:YEAR0 is 00-99.

### 22.3.14 Control Register (CTRL)

**NOTE:** Out of reset, the control register (CTRL) is write-protected. To disable write protection, correct keys must be written to the KICKnR registers (see [Section 22.2.6](#)).

The control register (CTRL) is shown in [Figure 22-17](#) and described in [Table 22-16](#).

**Figure 22-17. Control Register (CTRL)**

Reserved								8
R-0								
7	6	5	4	3	2	1	0	
SPLITPOWER	RTCDISABLE	SET32COUNTER	Reserved	HOURMODE	AUTOCOMP	ROUNDMIN	RUN	
W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

**Table 22-16. Control Register (CTRL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	SPLITPOWER	0 1	Enable leakage-isolation circuitry used for isolated power schemes. <b>Write-only bit. Read-modify-write updates to the control register may unintentionally clear the SPLITPOWER bit because the bit always reads back 0.</b> 0 Disable split power. 1 Enable split power.
6	RTCDISABLE	0 1	Disable RTC module and gate 32-kHz reference clock. RTC should only be disabled using this bit if the module will never be used and saving power is desired. 0 RTC is functional. 1 RTC is disabled and 32-kHz reference clock is gated.
5	SET32COUNTER	0 1	Set the 32-kHz counter with the value stored in the compensation registers when the SET32COUNTER bit is set. RTC does not run normally when the SET32COUNTER bit is high so this bit should be toggled low-high-low when used. 0 No action. 1 Set 32-kHz counter with compensation register value.
4	Reserved	0	Reserved.
3	HOURMODE	0 1	Enable 12-hour mode for HOURS and ALARMHOURS registers. 0 24 Hour Mode (Valid hours 00-24). 1 12 Hour Mode (Valid hours 00-12; MERIDIEM bit in HOURS and ALARMHOURS must be used to denote AM or PM).
2	AUTOCOMP	0 1	Enable oscillator compensation mode. Compensation takes place once every hour. 0 Auto compensation is disabled. 1 Auto compensation is enabled.
1	ROUNDMIN	0 1	Enable one-time rounding to nearest minute on next time register read. 0 Minute rounding disabled. 1 Rounding to nearest minute enabled.
0	RUN	0 1	Stop the RTC 32-kHz counter. RTC should be stopped using this bit for stopping and resuming the counter. 0 Stop RTC counter. 1 Run RTC counter.

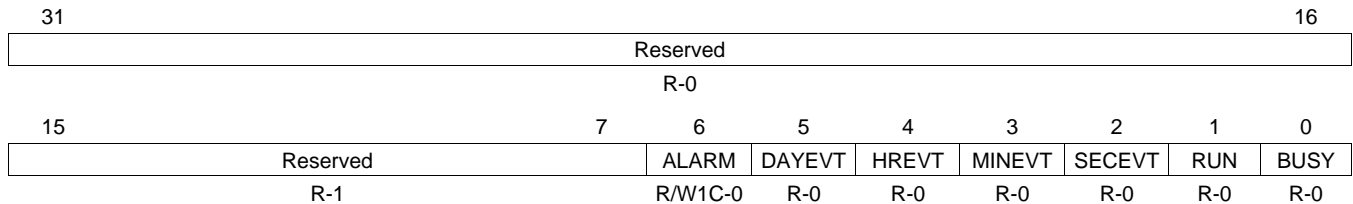


### 22.3.15 Status Register (STATUS)

The STATUS register is shown in [Figure 22-18](#) and described in [Table 22-17](#).

**NOTE:** Out of reset, the STATUS register is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

**Figure 22-18. Status Register (STATUS)**



LEGEND: R = Read only; W1C = Write 1 to clear bit; -n = value after reset

**Table 22-17. Status Register (STATUS) Field Descriptions**

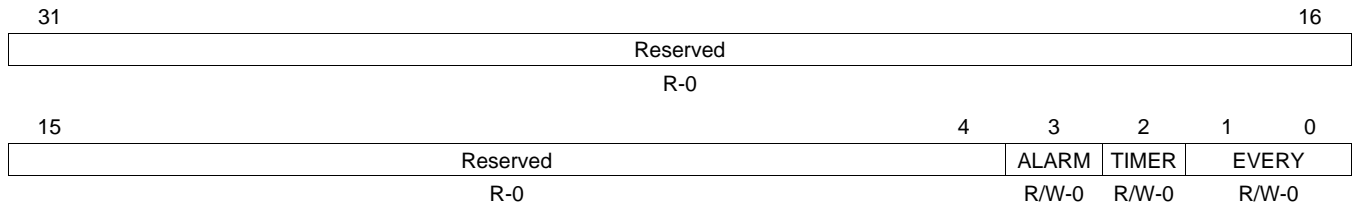
Bit	Field	Value	Description
31-7	Reserved	1	Reserved
6	ALARM	0	Indicates if an alarm interrupt has been generated. Write a 1 to clear the ALARM status. No new alarm interrupt was generated.
		1	New alarm interrupt was generated.
5	DAYEVT	0	When the (TIMER = 1) in the INTERRUPTS register, DAYEVT indicates if the DAYS register incremented during the most recent time update. DAYS register did not increment during the last time update.
		1	DAYS register incremented during the last time update.
4	HREVT	0	When the (TIMER = 1) in the INTERRUPTS register, HREVT indicates if the HOURS register incremented during the most recent time update. HOURS register did not increment during the last time update.
		1	HOURS register incremented during the last time update.
3	MINEVT	0	When the (TIMER = 1) in the INTERRUPTS register, MINEVT indicates if the MINUTES register incremented during the most recent time update. MINUTES register did not increment during the last time update.
		1	MINUTES register incremented during the last time update.
2	SECEVT	0	When the (TIMER = 1) in the INTERRUPTS register, SECEVT indicates if the SECONDS register incremented during the most recent time update. SECONDS register did not increment during the last time update.
		1	SECONDS register incremented during the last time update.
1	RUN	0	Indicates if RTC is running or stopped. RTC is stopped.
		1	RTC is running.
0	BUSY	0	Indicates if RTC is busy updating or is within 15 $\mu$ s of updating the time and calendar registers. RTC is free. The time, calendar, and control registers can be written to without contention.
		1	RTC is or will soon be busy updating the time and calendar registers.

### 22.3.16 Interrupt Register (INTERRUPT)

The INTERRUPT register is shown in [Figure 22-19](#) and described in [Table 22-18](#).

**NOTE:** Out of reset, the INTERRUPT register is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

**Figure 22-19. Interrupt Register (INTERRUPT)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-18. Interrupt Register (INTERRUPT) Field Descriptions**

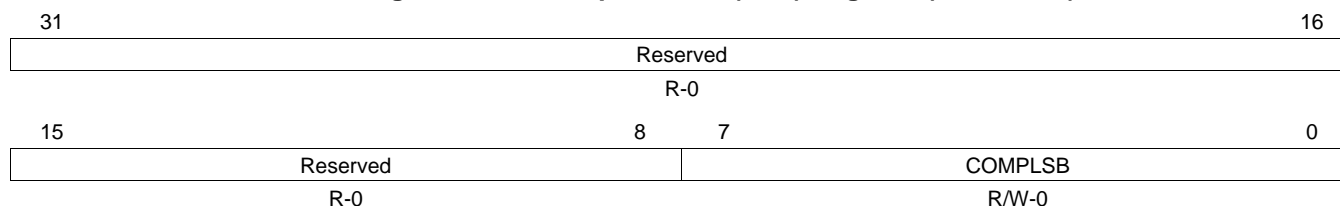
Bit	Field	Value	Description
31-4	Reserved	0	Reserved.
3	ALARM	0 1	Enable interrupt generation for when the alarm time and date match the current time and date Alarm interrupt is disabled. Alarm interrupt is enabled.
2	TIMER	0 1	Enable periodic timer interrupt generation. Period is determined by the EVERY field. Periodic timer interrupt is disabled. Periodic timer interrupt is enabled.
1-0	EVERY	0-3h 0 1h 2h 3h	Selects the time period desired when periodic timer interrupts are enabled by the TIMER bit. Second Minute Hour Day

### 22.3.17 Compensation (LSB) Register (COMPLSB)

**NOTE:** Out of reset, the compensation (LSB) register (COMPLSB) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The compensation (LSB) register (COMPLSB) works together with the COMPMSB register to set the hourly oscillator compensation value. The AUTOCOMP bit in the control register (CTRL) must be enabled for compensation to take place. The COMPLSB register is shown in [Figure 22-20](#) and described in [Table 22-19](#).

**Figure 22-20. Compensation (LSB) Register (COMPLSB)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-19. Compensations Register (COMPLSB) Field Descriptions**

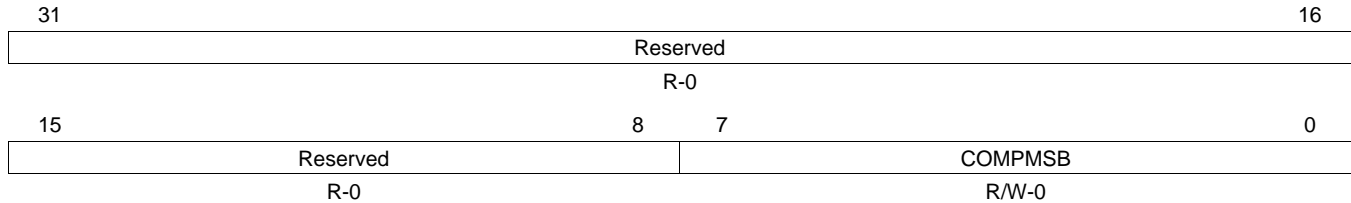
Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-0	COMPLSB	0-FFh	Lower bits of the 16-bit compensation value. The COMPMSB:COMPLSB register value is subtracted from the 32-kHz period. Compensation values are two's complement. The COMPMSB:COMPLSB value of 7F:FFh is not allowed.

### 22.3.18 Compensation (MSB) Register (COMPMSB)

**NOTE:** Out of reset, the compensation (MSB) register (COMPMSB) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The compensation (MSB) register (COMPMSB) works together with the COMPLSB register to set the hourly oscillator compensation value. The AUTOCOMP bit in the control register (CTRL) must be enabled for compensation to take place. The COMPMSB register is shown in [Figure 22-21](#) and described in [Table 22-20](#).

**Figure 22-21. Compensation (MSB) Register (COMPMSB)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 22-20. Compensations Register (COMPMSB) Field Descriptions**

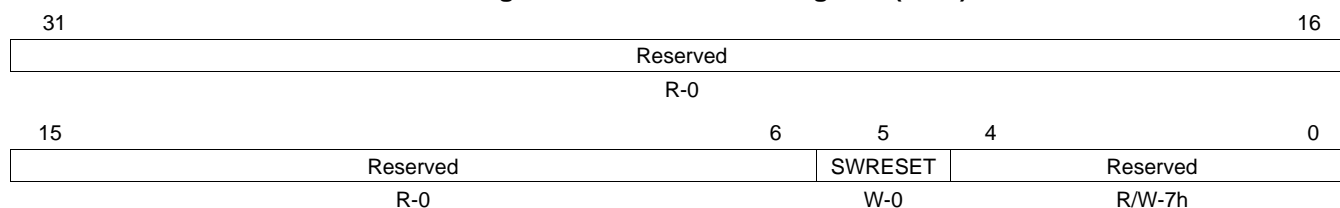
Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-0	COMPMSB	0-FFh	Lower bits of the 16-bit compensation value. The COMPMSB:COMPLSB register value is subtracted from the 32-kHz period. Compensation values are two's complement. The COMPMSB:COMPLSB value of 7F:FFh is not allowed.

### 22.3.19 Oscillator Register (OSC)

**NOTE:** Out of reset, the oscillator register (OSC) is write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The oscillator register (OSC) is shown in [Figure 22-22](#) and described in [Table 22-21](#).

**Figure 22-22. Oscillator Register (OSC)**



LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

**Table 22-21. Oscillator Register (OSC) Field Descriptions**

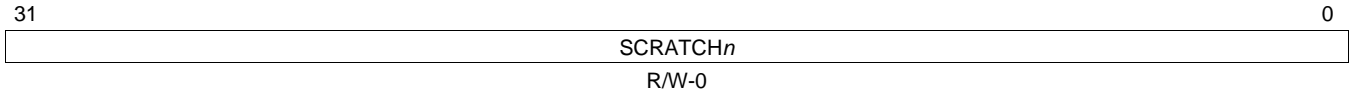
Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5	SWRESET	0	Software reset. Always reads back 0.
		0	No action.
		1	Reset RTC module and registers (except for OSC and KICK $n$ R registers). Registers must not be accessed for three 32-kHz reference periods after reset is asserted.
4-0	Reserved	7h	Reserved. This field is writeable, but should only be programmed to the value of 7h.

### 22.3.20 Scratch Registers (SCRATCH0-SCRATCH2)

**NOTE:** Out of reset, the scratch registers (SCRATCH $n$ ) are write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)).

The scratch registers (SCRATCH $n$ ) are 32-bit general-purpose registers that have no effect on RTC functionality. They can be used by the user arbitrarily. The SCRATCH $n$  register is shown in [Figure 22-23](#) and described in [Table 22-22](#).

**Figure 22-23. Scratch Registers (SCRATCH $n$ )**



LEGEND: R/W = Read/Write; - $n$  = value after reset

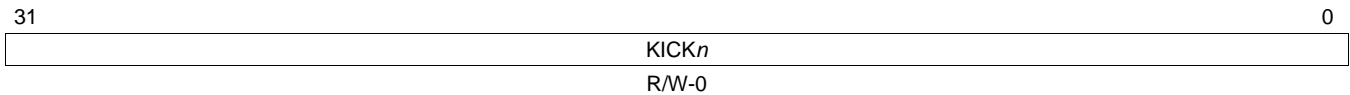
**Table 22-22. Scratch Registers (SCRATCH $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-0	SCRATCH $n$	0-FFFF FFFFh	General-purpose 32-bit registers that have no effect on RTC functionality.

### 22.3.21 Kick Registers (KICK0R, KICK1R)

The kick registers (KICK $n$ R) are used to enable and disable write protection on the RTC registers. Out of reset, the RTC registers are write-protected. To disable write protection, correct keys must be written to the KICK $n$ R registers (see [Section 22.2.6](#)). The KICK $n$ R register is shown in [Figure 22-24](#) and described in [Table 22-23](#).

**Figure 22-24. Kick Registers (KICK $n$ R)**



LEGEND: R/W = Read/Write; - $n$  = value after reset

**Table 22-23. Kick Registers (KICK $n$ R) Field Descriptions**

Bit	Field	Value	Description
31-0	KICK $n$	0-FFFF FFFFh	To disable RTC register write protection, the value of 83E7 0B13h must be written to KICK0R, followed by the value of 95A4 F1E0h written to KICK1R. RTC register write protection is enabled when any value is written to KICK0R.

## Serial Peripheral Interface (SPI)

---

---

This chapter describes the serial peripheral interface (SPI) module. See your device-specific data manual to determine how many SPIs are available on your device.

Topic	Page
<b>23.1 Introduction</b> .....	<b>989</b>
<b>23.2 Architecture</b> .....	<b>991</b>
<b>23.3 Registers</b> .....	<b>1017</b>

## 23.1 Introduction

### 23.1.1 Purpose of the Peripheral

The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (2 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the device and external peripherals. Typical applications include interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMS and analog-to-digital converters.

### 23.1.2 Features

The SPI has the following features:

- 16-bit shift register
- 16-bit Receive buffer register (SPIBUF) and 16-bit Receive buffer emulation 'alias' register (SPIEMU)
- 16-bit Transmit data register (SPIDAT0) and 16-bit Transmit data and format selection register (SPIDAT1)
- 8-bit baud clock generator
- Serial clock (SPIx\_CLK) I/O pin
- Slave in, master out (SPIx\_SIMO) I/O pin
- Slave out, master in (SPIx\_SOMI) I/O pin
- SPI enable ( $\overline{\text{SPIx\_ENA}}$ ) I/O pin (4-pin or 5-pin mode only)
- Multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) I/O pins (4-pin or 5-pin mode only)
- Programmable SPI clock frequency range
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)
- Interrupt capability
- DMA support (read/write synchronization events)

The SPI allows software to program the following options:

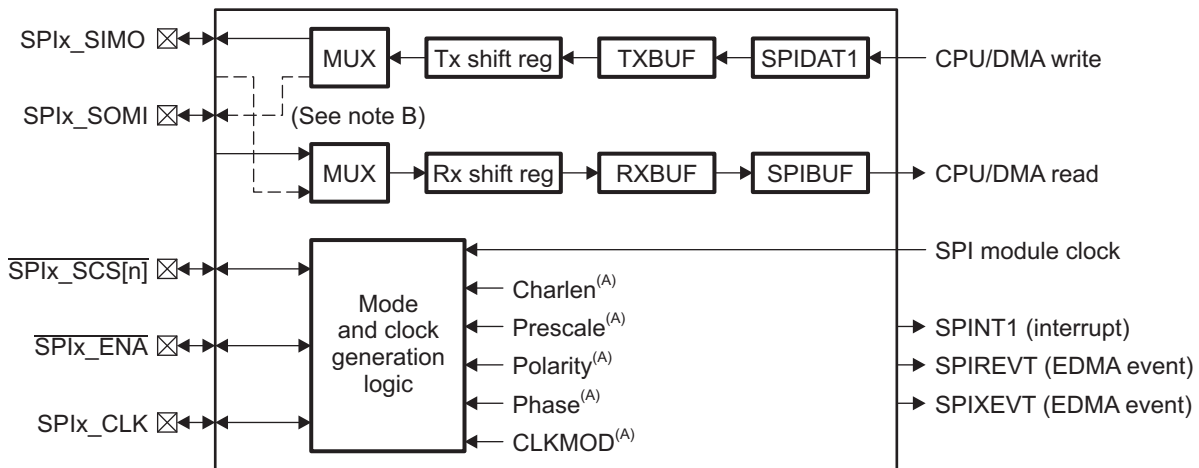
- SPI pins as functional or digital I/O pins
- SPI Master or Slave mode
- SPIx\_CLK frequency (SPI module clock/3 through SPI module clock/256)
- 3-pin, 4-pin, and 5-pin options
- Character length (2 to 16 bits) and shift direction (MSB/LSB first)
- Clock phase (delay or no delay) and polarity (high or low)
- Delay between transmissions in master mode.
- Chip select setup and hold times in master mode
- Chip select hold in master mode



### 23.1.3 Functional Block Diagram

The Figure 23-1 shows the SPI block diagram.

**Figure 23-1. SPI Block Diagram**



NOTE: The value *x* indicates the applicable SPI; that is, SPI0, SPI1, etc. See your device-specific data manual to determine how many SPIs are available on your device. The value *n* indicates the SPI pins available. See your device-specific data manual to determine how many SPI pins are available on your device.

A Indicates the log controlled by SPI register bits.

B Solid line represents data flow for SPI master mode. Dashed line represents data flow for SPI slave mode.

### 23.1.4 Industry Standard(s) Compliance Statement

The programmable configuration capability of the SPI allows it to gluelessly interface to a variety of SPI format devices. The SPI does not conform to a specific industry standard.

## 23.2 Architecture

This section describes the SPI operation modes. It gives an overview of SPI operation and then provides details on the 3-pin, 4-pin, and 5-pin options, as well as more specific details on the supported data formats.

### 23.2.1 Clock

The SPI clock (SPIx\_CLK) is derived from the SPI module clock. The maximum clock bit rate supported is SPI module clock/3, as determined by the PRESCALE field in the SPI data format register  $n$  (SPIFMT $n$ ). The SPIx\_CLK frequency is calculated as:

$$\text{SPIx\_CLK frequency} = [\text{SPI module clock}] / [\text{SPIFMT}n.\text{PRESCALE} + 1]$$

### 23.2.2 Signal Descriptions

Table 23-1 shows the SPI pins used to interface to external devices.

**Table 23-1. SPI Pins**

Pin <sup>(1)</sup>	Type	Function
SPIx_SIMO	Input/Output	Serial data input in slave mode, serial data output in master mode
SPIx_SOMI	Input/Output	Serial data output in slave mode, serial data input in master mode
SPIx_CLK	Input/Output	Serial clock input in slave mode, serial clock output in master mode
SPIx_SCS[n] <sup>(2)</sup>	Input/Output	Slave chip select output in master mode, input in slave mode
SPIx_ENA	Input/Output	Input in master mode, output in slave mode indicates slave is ready

<sup>(1)</sup> The value  $x$  indicates the applicable SPI; that is, SPI0, SPI1, etc. See your device-specific data manual to determine how many SPIs are available on your device.

<sup>(2)</sup> The value  $n$  indicates the SPI pins available; that is, SPIx\_SCS[0], SPIx\_SCS[1], etc. See your device-specific data manual to determine how many SPI pins are available on your device.

### 23.2.3 Operation Modes

The SPI operates in master or slave mode. The SPI bus master is the device that drives the SPIx\_CLK, SPIx\_SIMO, and optionally the SPIx\_SCS[n] signals, and therefore initiates SPI bus transfers. The CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) select between master and slave mode. In both master and slave mode, the SPI supports four options:

- 3-pin option
- 4-pin with chip select option
- 4-pin with enable option
- 5-pin with enable and chip select option

The 3-pin option is the basic clock, data in, and data out SPI interface and uses the SPIx\_CLK, SPIx\_SIMO, and SPIx\_SOMI pins. The 4-pin with chip select option adds the SPIx\_SCS[n] pin that is used to support multiple SPI slave devices on a single SPI bus. The 4-pin with enable option adds the SPIx\_ENA pin that is used to increase the overall throughput by adding hardware handshaking. The 5-pin option uses all the SPI pins and is a superset of the different options.

### 23.2.4 Programmable Registers

A general representation of the SPI programmable registers is shown in [Table 23-2](#). For details on registers, see [Section 23.3](#).

**Table 23-2. SPI Registers**

Offset Address <sup>(1)</sup>	Acronym	Name	Description	Section
0h	SPIGCR0	Global control register 0	Contains the software reset bit for the module	<a href="#">Section 23.3.1</a>
4h	SPIGCR1	Global control register 1	Controls basic configurations of the module	<a href="#">Section 23.3.2</a>
8h	SPIINT0	Interrupt register	Enable bits for interrupts, error, DMA and other functionality.	<a href="#">Section 23.3.3</a>
Ch	SPIVLV	Level register	SPI interrupt levels are set in this register.	<a href="#">Section 23.3.4</a>
10h	SPIFLG	Flag register	Shows the status of several events during the operation.	<a href="#">Section 23.3.5</a>
14h	SPIPC0	Pin control register 0	Determines if pins operate as general I/O or SPI functional pin	<a href="#">Section 23.3.6</a>
18h	SPIPC1	Pin control register 1	Controls the direction of data on the I/O pins	<a href="#">Section 23.3.7</a>
1Ch	SPIPC2	Pin control register 2	Reflects the values on the I/O pins	<a href="#">Section 23.3.8</a>
20h	SPIPC3	Pin control register 3	Controls the values sent to the I/O pins	<a href="#">Section 23.3.9</a>
24h	SPIPC4	Pin control register 4	Sets data values in the SPIPC3 register	<a href="#">Section 23.3.10</a>
28h	SPIPC5	Pin control register 5	Clears values in the SPIPC3 register	<a href="#">Section 23.3.11</a>
38h	SPIDAT0	Transmit data register 0	Transmit data register	<a href="#">Section 23.3.12</a>
3Ch	SPIDAT1	Transmit data register 1	Transmit data with format selection register	<a href="#">Section 23.3.13</a>
40h	SPIBUF	Receive buffer register	Holds received word	<a href="#">Section 23.3.14</a>
44h	SPIEMU	Receive buffer emulation register	Mirror of SPIBUF. Read does not clear flags	<a href="#">Section 23.3.15</a>
48h	SPIDELAY	Delay register	Sets $\overline{\text{SPIx\_SCS}}[n]$ mode, $\overline{\text{SPIx\_SCS}}[n]$ pre-/post-transfer delay time and $\overline{\text{SPIx\_ENA}}$ time-out	<a href="#">Section 23.3.16</a>
4Ch	SPIDEF	Chip select default register	In $\overline{\text{SPIx\_SCS}}[n]$ decoded mode only: sets high low/active $\overline{\text{SPIx\_SCS}}[n]$ signal	<a href="#">Section 23.3.17</a>
50h	SPIFMT0	Format 0 register	Configuration of data word format 0	<a href="#">Section 23.3.18</a>
54h	SPIFMT1	Format 1 register	Configuration of data word format 1	<a href="#">Section 23.3.18</a>
58h	SPIFMT2	Format 2 register	Configuration of data word format 2	<a href="#">Section 23.3.18</a>
5Ch	SPIFMT3	Format 3 register	Configuration of data word format 3	<a href="#">Section 23.3.18</a>
64h	INTVEC1	Interrupt vector register 1	Interrupt vector for line INT1	<a href="#">Section 23.3.19</a>

<sup>(1)</sup> The actual address of these registers is device specific and CPU specific. See your device-specific data manual to verify the SPI register addresses.

### 23.2.5 Master Mode Settings

The four master mode options are defined by the configuration bit settings listed in [Table 23-3](#). Other configuration bits may take any value in the range listed in [Table 23-4](#). The values listed in [Table 23-3](#) and [Table 23-4](#) should not be changed while the ENABLE bit in the SPI global control register 1 (SPIGCR1) is set to 1. Note that in certain cases the allowed values may still be ignored. For example, [Table 23-4](#) indicates that SPIDELAY may take a range of values in Master 3-pin mode; however, SPIDELAY has no effect in Master 3-pin mode. For complete details on each mode, see the following sections that explain the SPI operation for each of the master modes.

**Table 23-3. SPI Register Settings Defining Master Modes**

Register	Bit(s)	Master 3-pin	Master 4-pin Chip Select	Master 4-pin Enable	Master 5-pin
SPIGCR0	RESET	1	1	1	1
SPIGCR1	ENABLE	1	1	1	1
SPIGCR1	LOOPBACK	0	0	0	0
SPIGCR1	CLKMOD	1	1	1	1
SPIGCR1	MASTER	1	1	1	1
SPIPC0	SOMIFUN	1	1	1	1
SPIPC0	SIMOFUN	1	1	1	1
SPIPC0	CLKFUN	1	1	1	1
SPIPC0	ENAFUN	0	0	1	1
SPIPC0	SCS0FUN	0	1	0	1

**Table 23-4. Allowed SPI Register Settings in Master Modes**

Register	Bit(s)	Master 3-pin	Master 4-pin Chip Select	Master 4-pin Enable	Master 5-pin
SPIINT0	ENABLEHIGHZ	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	WDELAY	0 to 3Fh	0 to 3Fh	0 to 3Fh	0 to 3Fh
SPIFMT <sub>n</sub>	PARPOL	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	PARENA	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	WAITENA	0	0	1	1
SPIFMT <sub>n</sub>	SHIFTDIR	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	DISCSTIMERS	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	POLARITY	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	PHASE	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub>	PRESCALE	2 to FFh	2 to FFh	2 to FFh	2 to FFh
SPIFMT <sub>n</sub>	CHARLEN	2 to 10h	2 to 10h	2 to 10h	2 to 10h
SPIDELAY	C2TDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2CDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	C2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh

#### 23.2.5.1 Master Mode Timing Options

The SPI in master mode supports several options to modify the timing of its generation of the chip select signal ( $\overline{\text{SPiX\_SCS}}[n]$ ). This allows the SPI to support the timing requirements of various slave devices without adding additional overhead to the CPU by generating the appropriate delays automatically.

### 23.2.5.1.1 Chip Select Setup Time

The master can be configured to provide a (slow) slave device a certain chip select setup time to the first edge on SPIx\_CLK. This delay is controlled by the C2TDELAY field in the SPI delay register (SPIDELAY) and can be configured between 3 and 257 SPI module clock cycles. The C2TDELAY is applicable only in 4-pin with chip select and 5-pin SPI master modes. The C2TDELAY begins when the SPI master asserts SPIx\_SCS[n]. The C2T delay period is specified by:

*Maximum duration of C2TDELAY period = SPIDELAY.C2TDELAY + 2 (SPI module clock cycles)*

Note that if SPIDELAY.C2TDELAY = 0, then the C2TDELAY period = 0.

The previous value of the CSHOLD bit in the SPI transmit data register (SPIDAT1) must be cleared to 0 for the C2T delay to be enabled.

---

**NOTE:** If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

---

### 23.2.5.1.2 Chip Select Hold Time

The master can be configured to provide a (slow) slave device a certain chip select hold time after the last edge on SPIx\_CLK. This delay is controlled by the T2CDELAY bit in the SPI delay register (SPIDELAY) and can be configured between 2 and 256 SPI module clock cycles. The T2CDELAY is applicable only in 4-pin with chip select and 5-pin SPI master modes. The T2CDELAY begins after the data shifting period ends. The T2C delay period is specified by:

*Maximum duration of T2CDELAY period = SPIDELAY.T2CDELAY + 1 (SPI module clock cycle)*

Note that if SPIDELAY.T2CDELAY = 0, then the T2CDELAY period = 0. If the PHASE bit in the SPI data format register *n* (SPIFMT*n*) is 0, then the T2CDELAY period lasts for an additional 1/2 SPIx\_CLK time over that specified by the above equation.

The current value of the CSHOLD bit in the SPI transmit data register (SPIDAT1) must be cleared to 0 for T2C delay to be enabled.

---

**NOTE:** If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

---

### 23.2.5.1.3 Automatic Delay Between Transfers

The SPI master can automatically insert a delay of between 2 and 65 SPI module clock cycles between transmissions. This delay is controlled by the WDELAY field in the SPI data format register *n* (SPIFMT*n*) and is enabled by setting the WDEL bit in the SPI transmit data register (SPIDAT1) to 1. The WDELAY period begins when the T2EDELAY period terminates (if T2E delay period is enabled) or when the T2CDELAY period terminates (if T2E delay period was disabled and T2C delay period was enabled) or when the master deasserts SPIx\_SCS[n] (if T2E and T2C delay periods are disabled). If a transfer is initiated by writing a 32-bit value to SPIDAT1, then the new values of SPIDAT1.WDEL and SPIFMT*n*.WDELAY are used; otherwise, the old values of SPIDAT1.WDEL and SPIFMT*n*.WDELAY are used. The WDELAY delay period is specified by:

*Maximum duration of WDELAY period = SPIFMTn.WDELAY + 2 (SPI module clock cycles)*

### 23.2.5.1.4 Chip Select Hold Option

There are slave devices available that require the chip select signal to be held continuously active during several consecutive data word transfers. Other slave devices require the chip select signal to be deactivated between consecutive data word transfers. The SPI can support both types of slave devices. The CSHOLD bit in the SPI transmit data register (SPIDAT1) selects between the two options.

If the chip select hold option is enabled, the chip select will not toggle between two consecutive accesses; therefore, the SPIDELAY.T2CDELAY of the first transfer and the SPIDELAY.C2TDELAY of the second transfer will not be applied. However, the wait delay could still be applied between the two transactions, if the WDEL bit in SPIDAT1 is set to 1.

The current and previous values of the CSHOLD bit are retained. Though the current value of the CSHOLD bit is initialized to 0 when the RESET bit in the SPI global control register 0 (SPIGCR0) is cleared to 0, the previous value of the CSHOLD bit is not initialized. The previous value of the CSHOLD bit must be explicitly initialized by writing twice to the CSHOLD bit.

### 23.2.6 Slave Mode Settings

The four slave mode options are defined by the configuration bit settings listed in Table 23-5. Other configuration bits may take any value in the range listed in Table 23-6. The values listed in Table 23-5 and Table 23-6 should not be changed while the ENABLE bit in the SPI global control register 1 (SPIGCR1) is set to 1. Note that in certain cases the allowed values may still be ignored. For complete details on each mode, see the following sections that explain the SPI operation for each of the slave modes.

**Table 23-5. SPI Register Settings Defining Slave Modes**

Register	Bit(s)	Slave 3-pin	Slave 4-pin Chip Select	Slave 4-pin Enable	Slave 5-pin
SPIGCR0	RESET	1	1	1	1
SPIGCR1	ENABLE	1	1	1	1
SPIGCR1	LOOPBACK	0	0	0	0
SPIGCR1	CLKMOD	0	0	0	0
SPIGCR1	MASTER	0	0	0	0
SPIPC0	SOMIFUN	1	1	1	1
SPIPC0	SIMOFUN	1	1	1	1
SPIPC0	CLKFUN	1	1	1	1
SPIPC0	ENAFUN	0	0	1	1
SPIPC0	SCS0FUN	0	1	0	1

**Table 23-6. Allowed SPI Register Settings in Slave Modes**

Register	Bit(s)	Slave 3-pin	Slave 4-pin Chip Select	Slave 4-pin Enable	Slave 5-pin
SPIINT0	ENABLEHIGHZ	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	WDELAY	0 to 3Fh	0 to 3Fh	0 to 3Fh	0 to 3Fh
SPIFMT <sub>n</sub> <sup>(1)</sup>	PARPOL	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	PARENA	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	WAITENA	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	SHIFTDIR	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	DISCSTIMERS	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	POLARITY	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	PHASE	0,1	0,1	0,1	0,1
SPIFMT <sub>n</sub> <sup>(1)</sup>	PRESCALE	2 to FFh	2 to FFh	2 to FFh	2 to FFh
SPIFMT <sub>n</sub> <sup>(1)</sup>	CHARLEN	2 to 10h	2 to 10h	2 to 10h	2 to 10h
SPIDELAY	C2TDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2CDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	C2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh

<sup>(1)</sup> In slave mode, only SPIFMT0 is used. When SPIDAT1 is written, the DFSEL field in SPIDAT1 is cleared to 0 to select SPIFMT0.

### 23.2.7 SPI Operation: 3-Pin Mode

**NOTE:** If only unidirectional communication is required, the SPIx\_CLK pin and the two data pins (SPIx\_SOMI and SPIx\_SIMO) must all be configured as functional pins. A 2-pin unidirectional mode is not supported.

The SPI 3-pin mode uses only the clock (SPIx\_CLK) and data (SPIx\_SOMI and SPIx\_SIMO) pins for bidirectional communication between master and slave devices. Figure 23-2 shows the basic 3-pin SPI option.

To select the 3-pin SPI option, the SPIx\_CLK, SPIx\_SOMI, and SPIx\_SIMO pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0). The SPIx\_SCS[n] and SPIx\_ENA pins can be used as general-purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers.

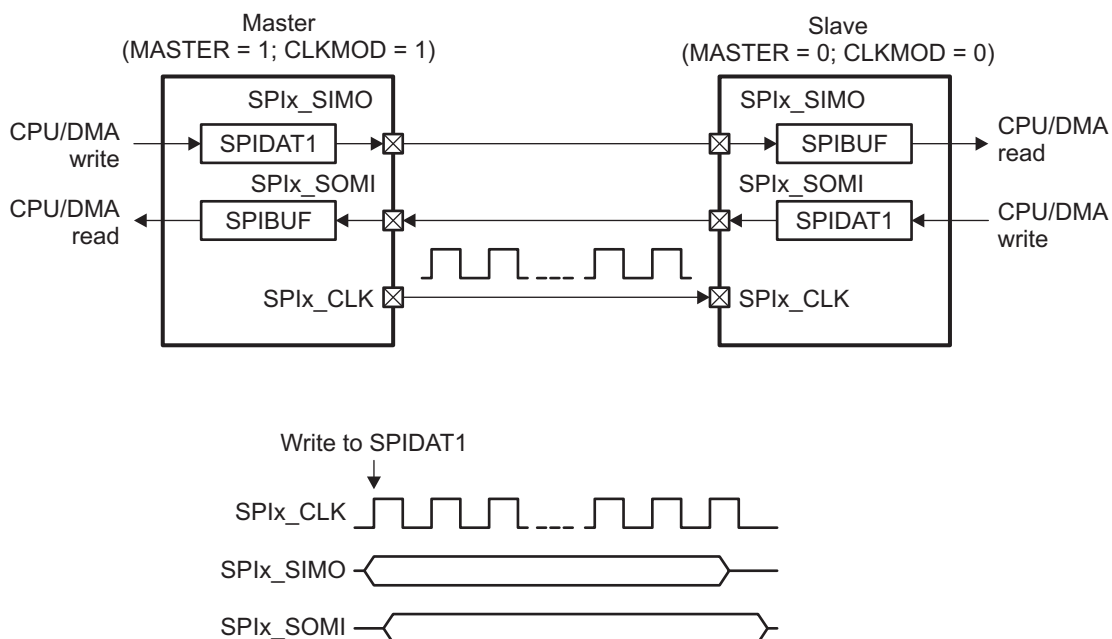
The SPI operates in either master or slave mode. The CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) select between master and slave mode; both must be programmed to 1 to configure the SPI for master mode or to 0 to configure the SPI for slave mode. The SPI bus master is the device that drives the SPIx\_CLK signal and initiates SPI bus transfers. In SPI master mode, the SPIx\_SOMI pin output buffer is in a high-impedance state and the SPIx\_CLK and the SPIx\_SIMO pin output buffer is enabled. In SPI slave mode, the SPIx\_SIMO and SPIx\_CLK pin output buffer is in a high-impedance state and the SPIx\_SOMI pin output buffer is enabled.

In master mode with the 3-pin option, the DSP writes transmit data to the SPI transmit data registers (SPIDAT0[15:0] or SPIDAT1[15:0]). This initiates a transfer. A series of clock pulses will be driven out on the SPIx\_CLK pin to complete the transfer. Each clock pulse on the SPIx\_CLK pin causes the simultaneous transfer (in both directions) of one bit by both the master and slave SPI devices. CPU writes to the configuration bits in SPIDAT1 (not writing to SPIDAT1[15:0]) do not result in a new transfer. When the selected number of bits has been transmitted, the received data is transferred to the SPI receive buffer register (SPIBUF) for the CPU to read. Data is stored right-justified in SPIBUF.

In slave mode with 3-pin option, CPU writes to SPIDAT0[15:0] or SPIDAT1[15:0] makes the slave ready to transmit. CPU writes to the configuration bits in SPIDAT1 (not writing to SPIDAT1[15:0]) do not make the slave ready to transmit.

**NOTE:** Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

**Figure 23-2. SPI 3-Pin Option**



### 23.2.8 SPI Operation: 4-Pin with Chip Select Mode

The 4-pin with chip select option is a superset of the 3-pin option and uses the chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pin in addition to the clock ( $\text{SPIx\_CLK}$ ) and data ( $\text{SPIx\_SOMI}$  and  $\text{SPIx\_SIMO}$ ) pins. Figure 23-3 shows the SPI 4-pin chip select option.

To select the 4-pin with chip select option, the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SOMI}$ ,  $\text{SPIx\_SIMO}$ , and  $\overline{\text{SPIx\_SCS}}[n]$  pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0). The  $\overline{\text{SPIx\_ENA}}$  pin can be used as a general-purpose I/O pin by configuring the SPIPC1 through SPIPC5 registers.

In SPI master mode, the  $\text{SPIx\_SOMI}$  pin output buffer is in a high-impedance state and the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SIMO}$ , and  $\overline{\text{SPIx\_SCS}}[n]$  pin output buffer is enabled. In SPI slave mode, the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SIMO}$ , and  $\overline{\text{SPIx\_SCS}}[n]$  pin output buffer is in a high-impedance state, and the  $\text{SPIx\_SOMI}$  pin output buffer is enabled when  $\overline{\text{SPIx\_SCS}}[n]$  is asserted and in a high-impedance state when  $\overline{\text{SPIx\_SCS}}[n]$  is deasserted.

In slave mode with the chip select option enabled, the SPI ignores all transactions on the bus unless  $\overline{\text{SPIx\_SCS}}[n]$  is asserted by the bus master. It also 3-states its output pin when  $\overline{\text{SPIx\_SCS}}[n]$  is deasserted by the master to avoid conflicting with the active slave device on the bus.

In master mode, the  $\overline{\text{SPIx\_SCS}}[n]$  pin functions as an output, and toggles when a specific slave device is selected. However, this is most useful on devices that support multiple  $\overline{\text{SPIx\_SCS}}[n]$  pins.

However, one reason to use the  $\overline{\text{SPIx\_SCS}}[n]$  pin as a functional pin for the SPI master is to take advantage of the timing parameters that can be set using the SPI delay register (SPIDELAY). The SPIDELAY allows delays to be added automatically so that the slave timing requirements between clock and chip select may be more easily met. Another reason would be to make use of the error detection built into the SPI.

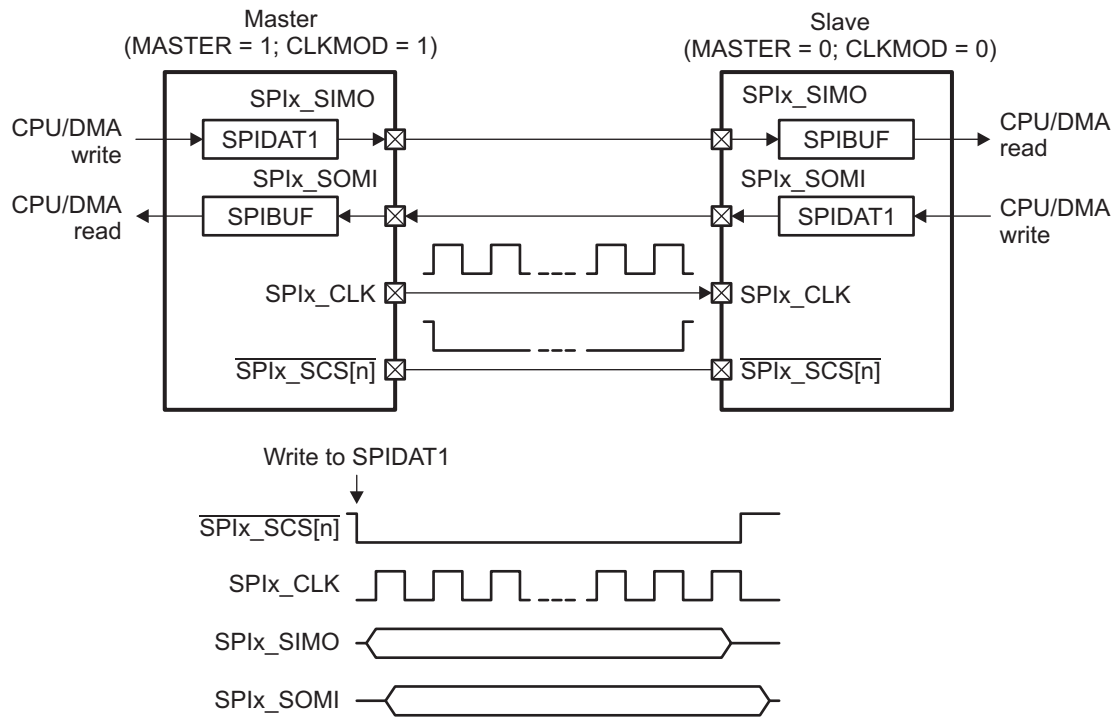
---

**NOTE:** Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

---



**Figure 23-3. SPI 4-Pin Option with SPIx\_SCS[n]**



**NOTE:** During an SPI transfer, if the slave mode SPI detects a deassertion of its chip select even before its internal character length counter overflows, then it 3-states its SPIx\_SOMI pin. Once this condition has occurred, if a SPIx\_CLK edge is detected while the chip select is deasserted, the SPI stops the transfer and sets an error flag DLENERR (data length) and generates an interrupt if enabled.

### 23.2.9 SPI Operation: 4-Pin with Enable Mode

The 4-pin with enable option is a superset of the 3-pin option and uses the enable ( $\overline{\text{SPIx\_ENA}}$ ) pin in addition to the clock ( $\text{SPIx\_CLK}$ ) and data ( $\text{SPIx\_SOMI}$  and  $\text{SPIx\_SIMO}$ ) pins. Figure 23-4 shows the SPI 4-pin enable option.

To select the 4-pin with enable option, the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SOMI}$ ,  $\text{SPIx\_SIMO}$ , and  $\overline{\text{SPIx\_ENA}}$  pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0). The  $\overline{\text{SPIx\_SCS[n]}}$  pins can be used as general-purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers.

In SPI master mode, the  $\text{SPIx\_SOMI}$  and  $\overline{\text{SPIx\_ENA}}$  pin output buffer is in a high-impedance state and the  $\text{SPIx\_CLK}$  and  $\text{SPIx\_SIMO}$  pin output buffer is enabled. In SPI slave mode, the  $\text{SPIx\_CLK}$  and  $\text{SPIx\_SIMO}$  pin output buffer is in a high-impedance state, and the  $\text{SPIx\_SOMI}$  pin output buffer is enabled. In SPI slave mode, the  $\overline{\text{SPIx\_ENA}}$  pin output buffer enable depends upon the status of the transmit buffer and the configuration of the ENABLEHIGHZ bit in the SPI interrupt register (SPIINT0).

The handshake operation works this way:

- After a transfer completes, both the master and slave SPI modules need to be serviced.
- The slave SPI deasserts  $\overline{\text{SPIx\_ENA}}$  after the transfer, indicating it requires servicing and is not ready.
- The slave should begin servicing its SPI by first reading receive data from the SPI receive buffer register (SPIBUF).
- Next, the slave device should write transmit data to the SPI transmit data registers (SPIDAT0 or SPIDAT1). This causes the slave SPI to assert  $\overline{\text{SPIx\_ENA}}$  indicating it is ready for the next transmission.
- In parallel, the master device can service its SPI at any time. It does not need to insert a delay before writing to its SPIDAT0 or SPIDAT1 in order to avoid overrunning the slave device. Instead, the master SPI module will automatically delay the next transfer until the slave has asserted  $\overline{\text{SPIx\_ENA}}$  again to indicate it is ready for the transmission.

This handshake allows the two SPIs to communicate at the maximum rate possible. Without the handshake pin, the master must insert a delay between each transfer long enough to support the worst case response time of the slave servicing its SPI or risk an overrun condition. With the handshake, the throughput is determined by the average response time of the two devices servicing their SPI ports.

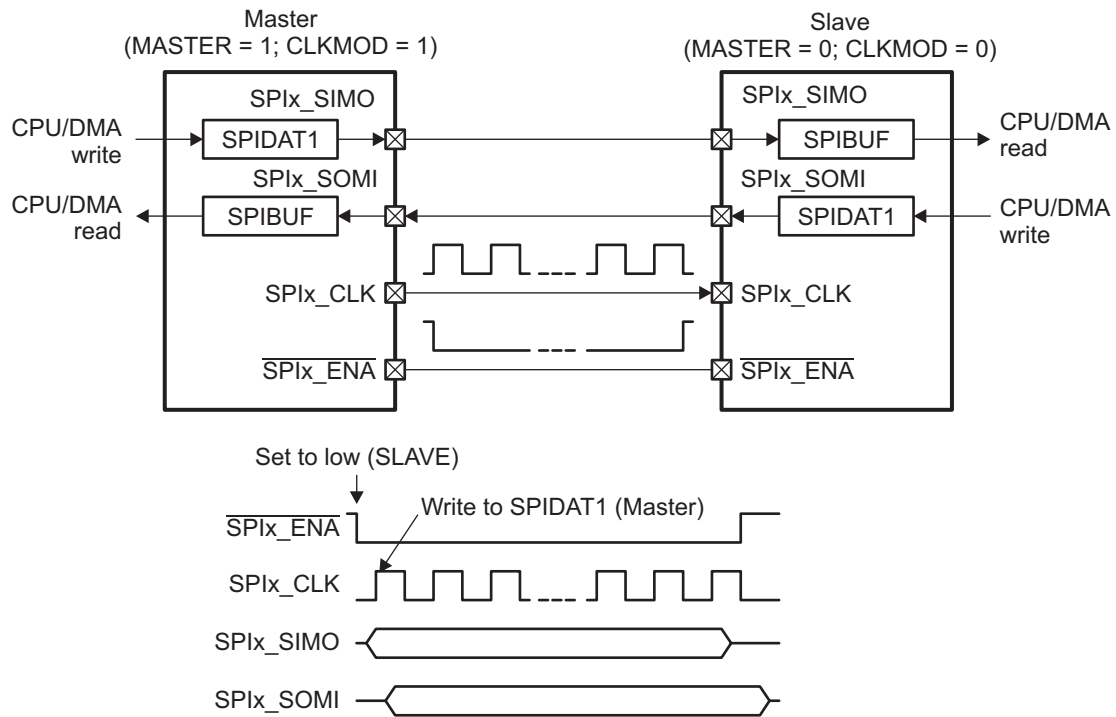
The  $\overline{\text{SPIx\_ENA}}$  pin can be driven in a push-pull or open-drain mode, depending upon the setting of the ENABLEHIGHZ bit.

---

**NOTE:** Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

---

**Figure 23-4. SPI 4-Pin Option with  $\overline{\text{SPIx\_ENA}}$**



### 23.2.10 SPI Operation: 5-Pin Mode

The 5-pin mode is a superset of both 4-pin modes. To use the 5-pin mode, both the  $\overline{\text{SPIx\_ENA}}$  and the  $\overline{\text{SPIx\_SCS[n]}}$  pins must be configured as functional pins, in addition to the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SIMO}$ , and  $\text{SPIx\_SOMI}$  pins by configuring the SPI pin control register 0 (SPIPC0). [Figure 23-5](#) shows the SPI 5-pin option.

In SPI master mode, the  $\text{SPIx\_SOMI}$  and  $\overline{\text{SPIx\_ENA}}$  pin output buffer is in a high-impedance state and the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SIMO}$ , and  $\overline{\text{SPIx\_SCS[n]}}$  pin output buffer is enabled. In SPI slave mode, the  $\text{SPIx\_CLK}$ ,  $\text{SPIx\_SIMO}$ , and  $\overline{\text{SPIx\_SCS[n]}}$  pin output buffer is in a high-impedance state, and the  $\text{SPIx\_SOMI}$  pin output buffer is enabled and disabled asynchronously by the  $\overline{\text{SPIx\_SCS[n]}}$  input and the  $\overline{\text{SPIx\_ENA}}$  pin output buffer enable depends upon the status of the transmit buffer and the state of the  $\overline{\text{SPIx\_SCS[n]}}$  input. In SPI slave mode, the assertion of the  $\overline{\text{SPIx\_ENA}}$  pin by the slave is delayed until the master asserts  $\overline{\text{SPIx\_SCS[n]}}$ , thereby, allowing multiple SPI slaves on a single SPI bus, each slave with its own enable pin.

If the  $\overline{\text{SPIx\_ENA}}$  pin is in high-impedance mode ( $\text{ENABLEHIGHZ} = 1$  in the SPI interrupt register (SPIINT0)), the slave SPI will put this signal into the high-impedance state by default. The slave SPI will drive the  $\overline{\text{SPIx\_ENA}}$  signal low when new data is written to the slave transmit shift register and the slave has been selected by the master ( $\overline{\text{SPIx\_SCS[n]}}$  is low).

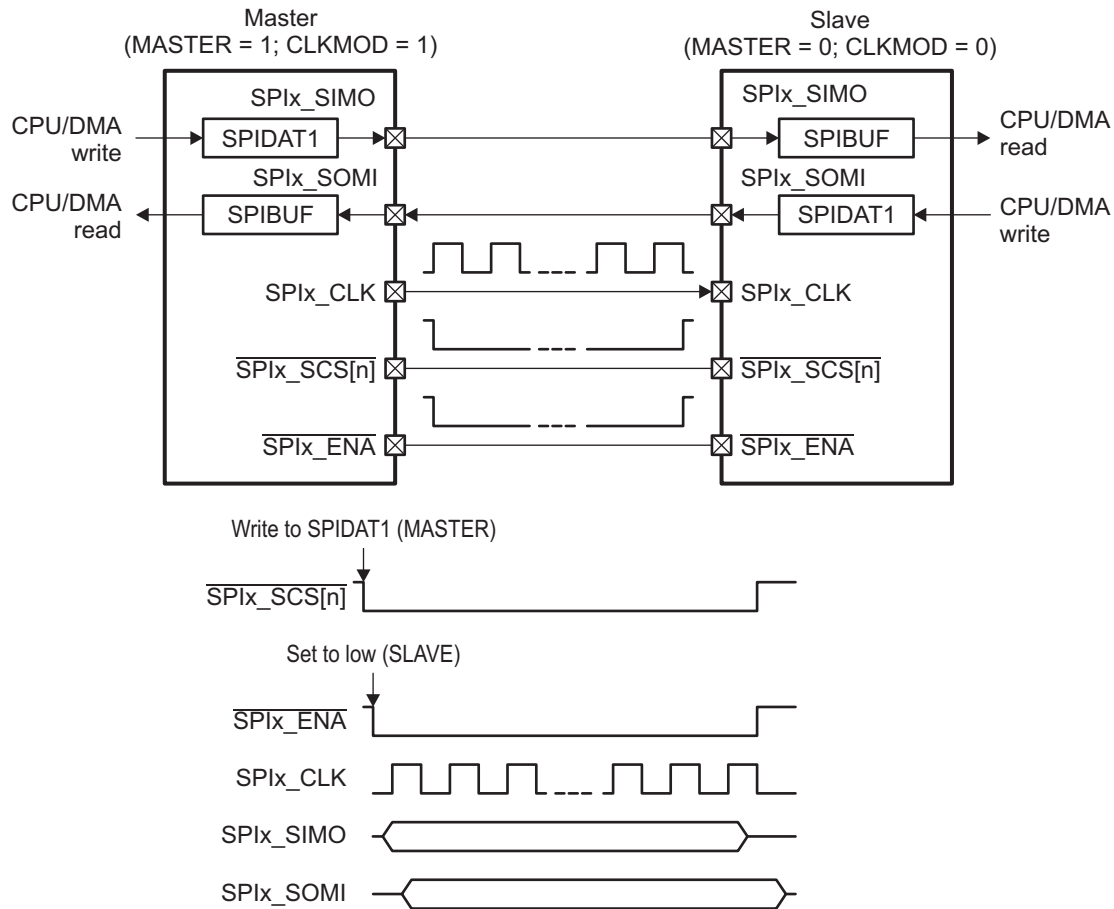
If the  $\overline{\text{SPIx\_ENA}}$  pin is in push-pull mode ( $\text{ENABLEHIGHZ} = 0$ ), the slave SPI will drive this pin high by default when it is in functional mode. The slave SPI will drive the  $\overline{\text{SPIx\_ENA}}$  signal low when new data is written to the slave transmit shift register and the slave is selected by the master ( $\overline{\text{SPIx\_SCS[n]}}$  is low). If the slave is deselected by the master ( $\overline{\text{SPIx\_SCS[n]}}$  goes high), the slave  $\overline{\text{SPIx\_ENA}}$  signal is driven high automatically.

---

**NOTE:** Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

---

**Figure 23-5. SPI 5-Pin Option with  $\overline{\text{SPIx\_ENA}}$  and  $\overline{\text{SPIx\_SCS[n]}}$**



**NOTE:** Push-Pull mode of the  $\overline{\text{SPIx\_ENA}}$  pin can be used only when there is a single slave in the system. When there are multiple SPI slave devices connected to the common  $\overline{\text{SPIx\_ENA}}$  pin, all the slaves should configure their  $\overline{\text{SPIx\_ENA}}$  pins in high-impedance mode.

During an SPI transfer, if slave mode SPI detects a deassertion of its chip select even before its internal character length counter overflows, then it 3-states its SPIx\_SOMI and  $\overline{\text{SPIx\_ENA}}$  (if SPIINT0.ENABLEHIGHZ bit is set to 1) pins. Once this condition has occurred, if a SPIx\_CLK edge is detected while the chip select is deasserted, then the SPI stops that transfer and sets an error flag DLENERR (data length) and generates an interrupt if enabled.

### 23.2.11 Data Formats

The SPI provides the capability to configure four independent data formats. These formats are configured by programming the corresponding SPI data format registers (SPIFMT $n$ ). In each data format, the following characteristics of the SPI operation are selected:

- Character length from 2 to 16 bits: The character length is configured by the SPIFMT $n$ .CHARLEN field.
- Shift direction (MSB first or LSB first): The shift out direction is configured by the SPIFMT $n$ .SHIFTDIR bit.
- Clock polarity: The clock polarity is configured by the SPIFMT $n$ .POLARITY bit.
- Clock phase: The clock phase is configured by the SPIFMT $n$ .PHASE bit.

The data format is chosen on each transaction. Transmit data is written to the SPI transmit data register 1 (SPIDAT1) and in the same write the data word format select (DFSEL) bit in SPIDAT1 indicates which data format is to be used for the next transaction. Alternatively, the data format can be configured once and applies to all transactions that follow until the data format is changed.

#### 23.2.11.1 Character Length

The character length is configured by the SPIFMT $n$ .CHARLEN bit. Legal values are 2 bits (2h) to 16 bits (10h). The character length is independently configured for each of the four data formats; and it must be programmed in both master mode and slave mode.

Transmit data is written to SPIDAT1. The transmit data must be written right-justified irrespective of the character length. The SPI automatically sends out the data correctly based on the chosen data format.

Figure 23-6 shows how a 12-bit word (EC9h) needs to be written to the transmit buffer in order to be transmitted correctly.

**Figure 23-6. Format for Transmitting 12-Bit Word**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	1	1	1	0	1	1	0	0	1	0	0	1

The data received in SPIBUF is right-justified irrespective of the character length and is padded with 0s when character length is less than 16.

Figure 23-7 shows how a 10-bit word (3A2h) is stored in the buffer once it is received.

**Figure 23-7. Format for 10-Bit Received Word**

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0

#### 23.2.11.2 Shift Direction

The shift out direction is configured as most-significant bit (MSB) first or least significant bit (LSB) first. The shift out direction is selected by the SPIFMT $n$ .SHIFTDIR bit. The shift out direction is independently configured for each of the four data formats.

- When SPIFMT $n$ .SHIFTDIR is 0, the transmit data is shifted out MSB first.
- When SPIFMT $n$ .SHIFTDIR is 1, the transmit data is shifted out LSB first.

### 23.2.11.3 Clock Phase and Polarity

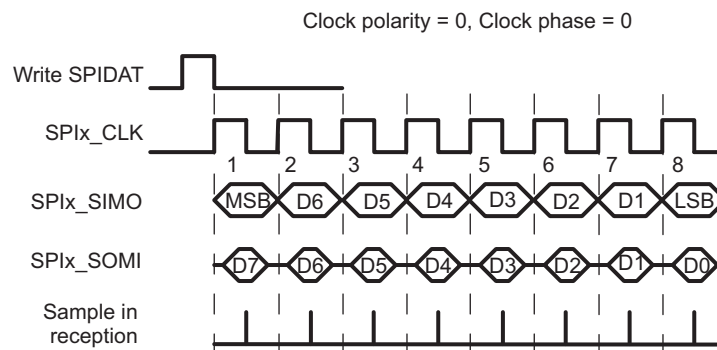
The SPI provides the flexibility to program four different clock mode combinations that SPIx\_CLK may operate, enabling a choice of the clock phase (delay or no delay) and the clock polarity (rising edge or falling edge). When operating with PHASE active, the SPI makes the first bit of data available after SPIDAT1 is written and before the first edge of SPIx\_CLK. The data input and output edges depend on the values of both the POLARITY and PHASE bits as shown in [Table 23-7](#).

**Table 23-7. Clocking Modes**

POLARITY	PHASE	Action
0	0	Data is output on the rising edge of SPIx_CLK. Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPIx_CLK and on subsequent falling edges. Input data is latched on the rising edge of SPIx_CLK.
1	0	Data is output on the falling edge of SPIx_CLK. Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPIx_CLK and on subsequent rising edges. Input data is latched on the falling edge of SPIx_CLK.

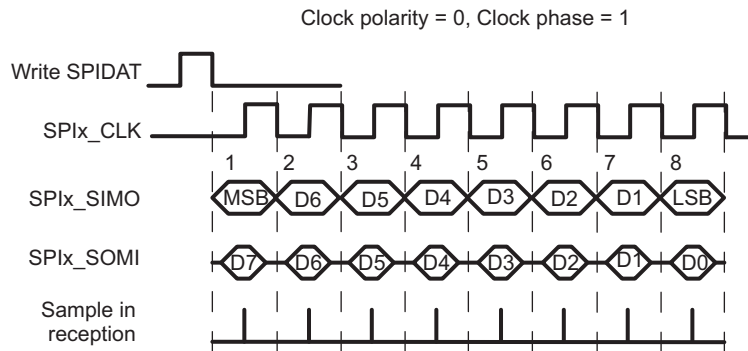
[Figure 23-8](#) to [Figure 23-11](#) illustrate the four possible signals of SPIx\_CLK corresponding to each mode. Having four signal options allows the SPI to interface with different types of serial devices. Also shown are the SPIx\_CLK control bit polarity and phase values corresponding to each signal.

**Figure 23-8. Clock Mode with POLARITY = 0 and PHASE = 0**



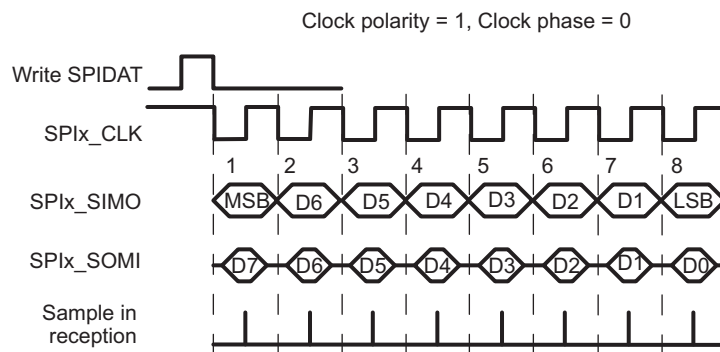
Clock phase = 0 (SPIx\_CLK without delay)  
 - Data is output on the rising edge of SPIx\_CLK  
 - Input data is latched on the falling edge of SPIx\_CLK  
 - A write to the SPIDAT register starts SPIx\_CLK

**Figure 23-9. Clock Mode with POLARITY = 0 and PHASE = 1**



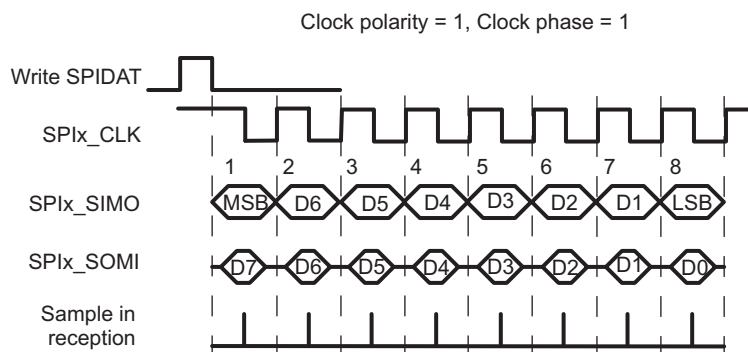
Clock phase = 1 (SPIx\_CLK with delay)  
 - Data is output one-half cycle before the first rising of SPIx\_CLK and on subsequent falling edges of SPIx\_CLK  
 - Input data is latched on the rising edge of SPIx\_CLK

**Figure 23-10. Clock Mode with POLARITY = 1 and PHASE = 0**



Clock phase = 0 (SPIx\_CLK without delay)  
 - Data is output on the falling edge of SPIx\_CLK  
 - Input data is latched on the rising edge of SPIx\_CLK  
 - A write to the SPIDAT register starts SPIx\_CLK

**Figure 23-11. Clock Mode with POLARITY = 1 and PHASE = 1**



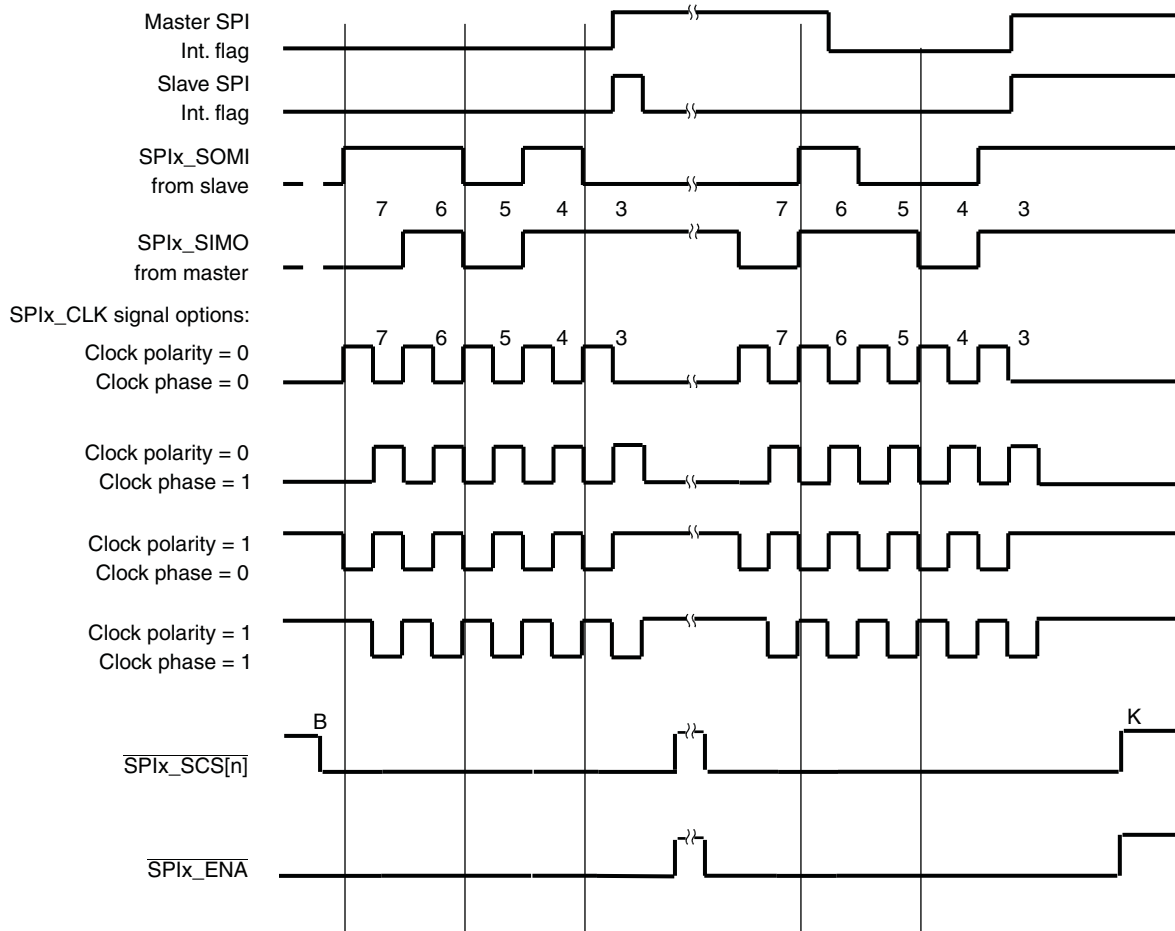
Clock phase = 1 (SPIx\_CLK with delay)  
 - Data is output one-half cycle before the first falling edge of SPIx\_CLK and on the subsequent rising edges of SPIx\_CLK  
 - Input data is latched on the falling edge of SPIx\_CLK



### 23.2.11.4 SPI Data Transfer Example

Figure 23-12 illustrates an SPI data transfer between two devices using a character length of five bits.

**Figure 23-12. Five Bits per Character (5-Pin Option)**



### 23.2.12 Interrupt Support

The SPI interrupt system is controlled by three registers:

- The SPI interrupt level register (SPILVL) controls the interrupt level. The interrupt level must be set to select the level one interrupt (INT1).
- The SPI interrupt register (SPIINT) contains bits to selectively enable/disable each interrupt event.
- The SPI flag register (SPIFLG) contains flags indicating the interrupt conditions that have occurred.

To identify the interrupt source in the SPI peripheral, the CPU reads the SPI flag status register (SPIFLG) or the INTVECT1 code in the SPI interrupt vector register 1 (INTVEC1).

Check your device-specific data manual for details on the exact CPU interrupt numbers assigned to the SPI interrupts.

### 23.2.13 DMA Events Support

If handling the SPI message traffic on a character-by-character basis requires too much CPU overhead, then the CPU can configure the system DMA to handle the SPI data transfer.

The SPI module has two DMA synchronization event outputs for receive (REVT) and transmit (XEVT), allowing DMA transfers to be triggered by SPI read receive and write transmit events. The SPI module enables DMA requests by enabling the DMA request enable (DMAREQEN) bit in the SPI interrupt register (SPIINT0).

When a character is to be transmitted the SPI module signals the DMA via the XEVT signal. The DMA controller then transfers the data from the source buffer into the SPI transmit data register (SPIDAT1). When a character is received, the SPI module signals the DMA via the REVT signal. The DMA controller then reads the data from the SPI receive buffer register (SPIBUF) and transfers it to a destination buffer for ready access.

In most cases, if the DMA is being used to service received data from the SPI, the receive interrupt enable (RXINTEN) bit in SPIINT0 should be cleared to 0. This prevents the CPU from responding to the received data in addition to the DMA. For specific SPI synchronization event number assignments and detailed DMA features, see your device-specific data manual.

### 23.2.14 Robustness Features

The SPI module includes many features to make the SPI communication link robust. An internal loopback test mode can be used to facilitate a power-on self test routine. Additionally, the SPI master continually monitors the bus for faults on its data line. The handshaking between master and slave can be monitored as well, and appropriate actions can be taken (interrupt, timeout) when the handshake breaks down. The following sections describe these robustness features in more detail.

#### 23.2.14.1 SPI Internal Loopback Test Mode (Master Only)

#### CAUTION

The internal loop-back self-test mode should not be entered during a normal data transaction or unpredictable operation may occur.

To select the loopback mode, the SPIx\_CLK, SPIx\_SOMI, SPIx\_SIMO pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0) and by setting the LOOPBACK bit in the SPI global control register 1 (SPIGCR1). The SPIx\_ENA and SPIx\_SCS[n] pins can be used as general-purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers. The internal loop-back self-test mode can be utilized to test the SPI transmit path and receive path including the transmit and receive buffers. In this mode, the transmit signal is internally fed back to the receiver and the SPIx\_SIMO, SPIx\_SOMI, and SPIx\_CLK pins are in a high-impedance state. This mode allows the CPU to write into the transmit buffer, and check that the receive buffer contains the correct transmit data. If an error occurs the corresponding error is set within the status field.

#### 23.2.14.2 SPI Transmission Continuous Self-Test

During a data transfer, the SPI inputs the value from its data output pin on the appropriate SPIx\_CLK edge. This value is compared against the expected value and any difference indicates a fault on the SPI bus. If a fault is detected, then the BITERR bit in the SPI receive buffer register (SPIBUF) and the BITERRFLG bit in the SPI flag register (SPIFLG) are set and an error interrupt is generated if enabled. The SPI continuous self-test mode is not available in SPI loopback mode.

### 23.2.14.3 SPI Detection of Slave Desynchronization

In the 4-pin with enable and 5-pin modes, the SPI master can monitor the slave  $\overline{\text{SPIx\_ENA}}$  activity to detect a desynchronization event.

Some conditions that may cause a desynchronization event are:

- Master or slave device being reset during a transmission.
- Asserting a software reset of the SPI module during transmission.
- Having an incorrect SPI pin configuration, causing the  $\overline{\text{SPIx\_ENA}}$  pin to behave incorrectly.
- Signal integrity problem causing additional clocks to be recognized by the slave.

The master can detect two desynchronization error conditions on the  $\overline{\text{SPIx\_ENA}}$  pin:

1. Slave deasserts  $\overline{\text{SPIx\_ENA}}$  after a transmission has begun, but before it completes.
2. Slave fails to deassert  $\overline{\text{SPIx\_ENA}}$  within a certain time period after the completion of the last bit of the transmission.

The first error condition is straightforward to detect. To detect the second error condition, the SPI module includes an eight-bit counter with a timeout count that can be configured through the T2EDELAY field in the SPI delay register (SPIDELAY).

When a desynchronization event is detected, the DESYNC bit in the SPI receive buffer register (SPIBUF) and the DESYNCFLG bit in the SPI flag register (SPIFLG) are set and a desynchronization error interrupt is asserted if enabled.

---

**NOTE:** Remember that even though the desynchronization is detected by the master device, the problem causing the desynchronization event can be on either the master or the slave device.

---

The T2EDELAY period begins once the T2CDELAY period terminates or after the data shifting period in case the T2CDELAY is disabled. It defines the maximum time for the slave to deassert the  $\overline{\text{SPIx\_ENA}}$  signal. If the slave device does not deassert the  $\overline{\text{SPIx\_ENA}}$  signal before the T2EDELAY timeout value expires, the SPIFLG.DESYNC flag is set and a desynchronization interrupt is asserted if enabled. The T2E delay period does not always complete, sometimes it is skipped or terminated early. The T2E delay period terminates immediately after the  $\overline{\text{SPIx\_ENA}}$  input is sampled (using the SPI module clock at intervals of  $\text{SPIFMTn.PRESCALE} + 2$ ) as deasserted. However, assuming the T2E period completes its duration is specified by:

*Maximum duration of T2EDELAY period = SPIDELAY.T2EDELAY + SPIFMTn.PRESCALE + 2 (SPI module clock cycles)*

The T2EDELAY period is enabled only when the  $\overline{\text{SPIx\_ENA}}$  is asserted at the beginning of the T2E delay period, the SPIDELAY.T2EDELAY field has a non-zero value, and SPIFMTn.WAITENA bit is set to 1.

### 23.2.14.4 $\overline{\text{SPIx\_ENA}}$ Signal Time-Out

In 5-pin mode, in addition to the slave desynchronization detection, the master can also detect whether the slave fails to respond to the  $\text{SPIx\_SCS}[n]$  signal by asserting  $\overline{\text{SPIx\_ENA}}$  in a timely manner.

This condition could be the result of a serious error, or it could simply be the result of the slave device taking too long to service its SPI.

To detect this condition, the C2EDELAY field in the SPI delay register (SPIDELAY) is used. The C2EDELAY period begins once the C2TDELAY period terminates or when the master asserts  $\text{SPIx\_SCS}[n]$  (if C2TDELAY is disabled). It defines the maximum time for the addressed slave to respond by activating the  $\overline{\text{SPIx\_ENA}}$  signal. If the slave does not respond with the  $\overline{\text{SPIx\_ENA}}$  signal before the timeout value expires, then the TIMEOUT bit in the SPI receive buffer register (SPIBUF) and the TIMEOUTFLG bit in the SPI flag register (SPIFLG) are set, an interrupt is asserted if enabled, and the current transfer is terminated. The C2E delay period does not always complete, sometimes it is skipped or terminated early. The C2E delay period terminates immediately after the  $\overline{\text{SPIx\_ENA}}$  input is sampled (using the SPI module clock at intervals of  $\text{SPIFMTn.PRESCALE} + 2$ ) as asserted. However, assuming the C2E period completes its duration is specified by:

*Maximum duration of C2EDELAY period = SPIDELAY.C2EDELAY + SPIFMTn.PRESCALE + 2 (SPI module clock cycles)*

The C2EDELAY period is enabled only when the  $\overline{\text{SPIx\_ENA}}$  is deasserted at the beginning of the C2E delay period and SPIFMTn.WAITENA bit is set to 1. If SPIFMTn.WAITENA bit is set to 1 and C2EDELAY is cleared to 0, then the master waits indefinitely for the slave to assert  $\text{SPIx\_ENA}$ .

### 23.2.14.5 SPI Data Length Error

An SPI can generate an error flag by detecting any mismatch in length of received/transmitted data with the programmed character length under certain conditions.

**Master Mode:** During a data transfer, if the SPI detects a deassertion of the  $\overline{\text{SPIx\_ENA}}$  pin (by the slave) while the character counter is not overflowed, then an error flag is set indicating the data length error. This can be caused by a slave receiving extra clocks (because of noise on the SPIx\_CLK line).

---

**NOTE:** In SPI master mode, the data length error will be generated only if the  $\overline{\text{SPIx\_ENA}}$  pin is used as a functional pin.

---

**Slave Mode:** During a transfer, if the SPI detects a deassertion of the  $\overline{\text{SPIx\_SCS[n]}}$  pin before its character length counter overflows, then an error flag is set indicating the data length error. If the slave SPI misses one or more SPIx\_CLK pulses from the master, this situation can occur. This error in slave mode would mean that both the transmitted and received data were not complete.

---

**NOTE:** In SPI slave mode, the data length error flag will be generated only if the  $\overline{\text{SPIx\_SCS[n]}}$  pin is configured as a functional pin.

---

## 23.2.15 Reset Considerations

This section describes the software and hardware reset considerations.

### 23.2.15.1 Software Reset Considerations

The SPI module contains a software reset (RESET) bit in the SPI global control register 0 (SPIGCR0) that is used to reset the SPI module. As a result of a reset, the SPI module register values go to their reset state. The RESET bit must be set before any operation on the SPI is done.

### 23.2.15.2 Hardware Reset Considerations

In the event of a hardware reset, the SPI module register values go to their reset state and the application software needs to reprogram the registers to the desired values.

## 23.2.16 Power Management

The SPI module can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SPI. During global low-power mode, all clocks to the SPI are turned off so the module is completely inactive.

The SPI local low-power mode is asserted by setting the POWERDOWN bit in the SPI global control register 1 (SPIGCR1). Setting this bit stops the clocks to the SPI internal logic and the SPI registers. Setting the POWERDOWN bit causes the SPI to enter local low-power mode and clearing the POWERDOWN bit causes SPI to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SPI for that particular access alone.

Since entering a low-power mode has the effect of suspending all state machine activities, care must be taken when entering such modes to ensure that a valid state is entered when low-power mode is active. As a result, application software must ensure that a low-power mode is not entered during a transmission or reception of data.

### 23.2.17 General-Purpose I/O Pin

Each of the SPI pins may be programmed via the SPI pin control registers (SPIPC0 to SPIPC5) to be a general-purpose I/O (GPIO) pin.

When the SPI pins are not used as functional pins, they may be programmed to be either general input or general output pins by configuring SPIPC0. For example, in 3-pin mode, SPIx\_SOMI, SPIx\_SIMO, and SPIx\_CLK must be configured as SPI pins, while the SPIx\_SCS[n] and SPIx\_ENA pins should be configured as GPIO pins. The direction is controlled by configuring SPIPC1.

If configured as a general-purpose output, then SPIPC3 controls the output value. There is also a write 1 to set (SPIPC4) and a write 1 to clear (SPIPC5) for the data out value. These registers allow different tasks running on the CPU to manipulate the SPI I/O pins without read-modify-write hazards.

SPIPC2 reflects the current value on the pin when the particular pin is configured as a functional or general-purpose input pin. When the pin is configured as a functional or general-purpose output pin, SPIPC2 indicates the value that is attempted to be driven on the pin.

### 23.2.18 Emulation Considerations

#### CAUTION

Viewing or otherwise reading the following SPI registers: SPIBUF, SPIFLG, and INTVEC1 through the JTAG debugger causes their contents to change, possibly invalidating the results of the debug session. Be sure to set up the debugger to avoid reading these registers.

The SPI module does not support soft or hard stop during emulation breakpoints. The SPI module will continue to run if an emulation breakpoint is encountered.

In addition, any status registers that are cleared after reading will be affected if viewed in a memory or watch window of the debugger; since the emulator will read these registers to update the value displayed in the window.

### 23.2.19 Initialization

Perform the following procedure for initializing the SPI:

1. Reset the SPI by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Take the SPI out of reset by setting SPIGCR0.RESET to 1.
3. Configure the SPI for master or slave mode by configuring the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1).
4. Configure the SPI for 3-pin, 4-pin with chip select, 4-pin with enable, or 5-pin mode by configuring the SPI pin control register 0 (SPIPC0).
5. Choose the SPI data format register  $n$  (SPIFMT $n$ ) to be used by configuring the DFSEL bit in the SPI transmit data register (SPIDAT1). In slave mode, only SPIFMT0 is supported.
6. Configure the SPI data rate, character length, shift direction, phase, polarity and other format options using SPIFMT $n$  selected in step 5.
7. If SPI master, then configure the master delay options using the SPI delay register (SPIDELAY). In slave mode, SPIDELAY is not relevant.
8. Select the error interrupt notifications by configuring the SPI interrupt register (SPIINT0) and the SPI interrupt level register (SPILVL).
9. Enable the SPI communication by setting the SPIGCR1.ENABLE to 1.
10. Setup and enable the DMA for SPI data handling and then enable the DMA servicing for the SPI data requests by setting the SPIINT0.DMAREQEN to 1.
11. Handle SPI data transfer requests using DMA and service any SPI error conditions using the interrupt service routine.

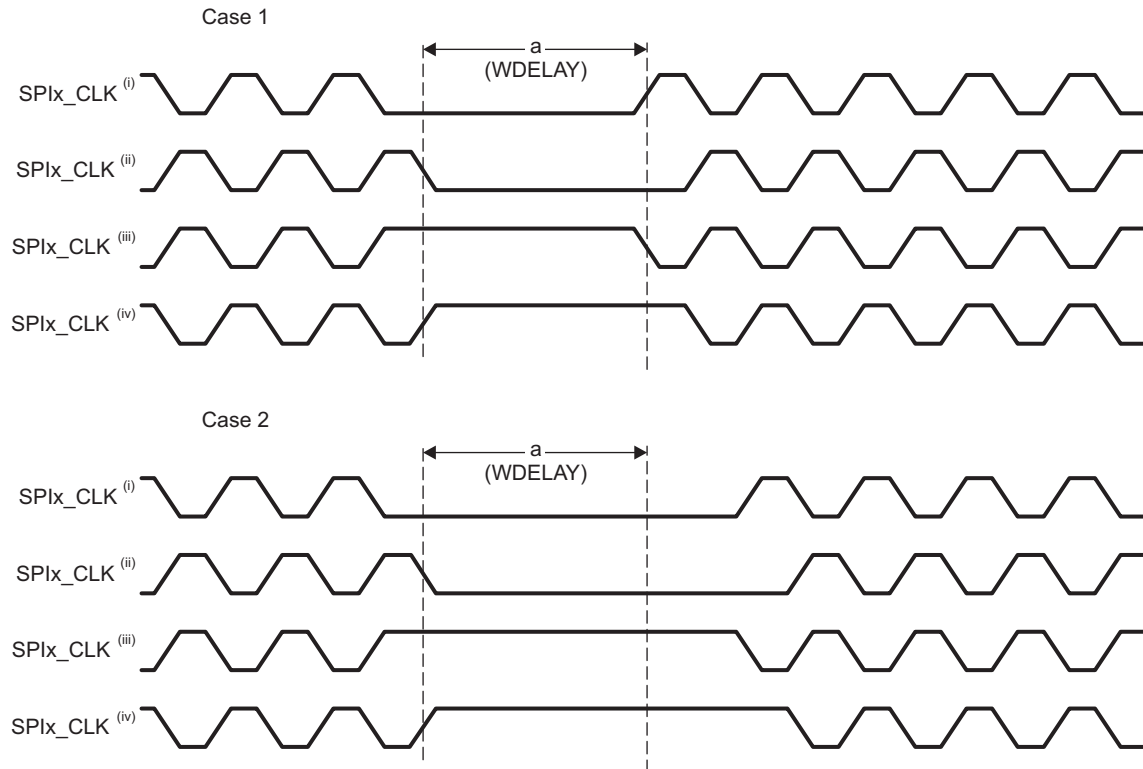
### 23.2.20 Timing Diagrams

This section contains timing diagrams illustrating the C2TDELAY, C2EDELAY, T2CDELAY, T2EDELAY, and WDELAY delays and their interaction with the SPIx\_SCS[n] and SPIx\_ENA pins for all SPI modes.

#### 23.2.20.1 SPI 3-Pin Mode

Figure 23-13 illustrates the WDELAY option in SPI 3-pin master mode. This is the only delay available in this mode. In CASE1, a new transfer is initiated during the WDELAY period and the transfer begins immediately after the WDELAY period ends. In CASE2, while WDELAY has completed, a new transfer will not begin until SPIDAT0/SPIDAT1 have been written with new data.

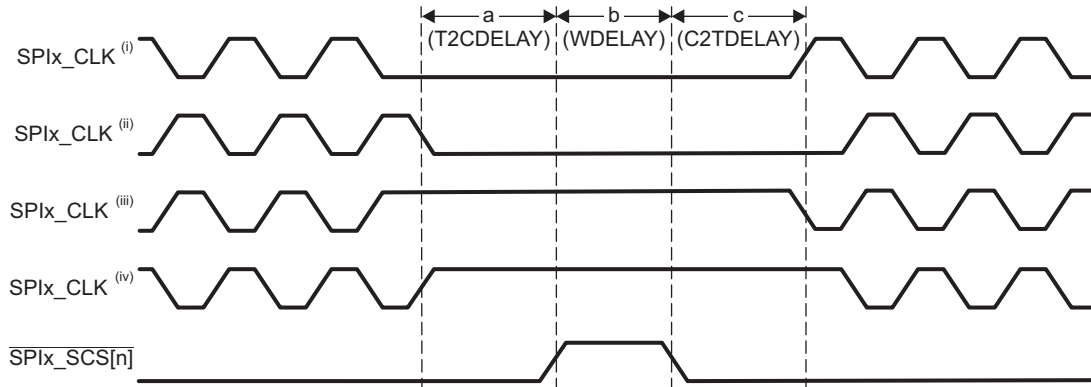
**Figure 23-13. SPI 3-Pin Master Mode with WDELAY**



### 23.2.20.2 SPI 4-Pin with $\overline{\text{SPIx\_SCS[n]}}$ Mode

Figure 23-14 illustrates the T2CDELAY, WDELAY and C2TDELAY delays in SPI 4-pin with  $\overline{\text{SPIx\_SCS[n]}}$  master mode. C2EDELAY and T2EDELAY are not available in this mode. All the three delay periods T2CDELAY, WDELAY, and C2TDELAY proceed to completion when enabled.

**Figure 23-14. SPI 4-Pin with  $\overline{\text{SPIx\_SCS[n]}}$  Mode with T2CDELAY, WDELAY, and C2TDELAY**



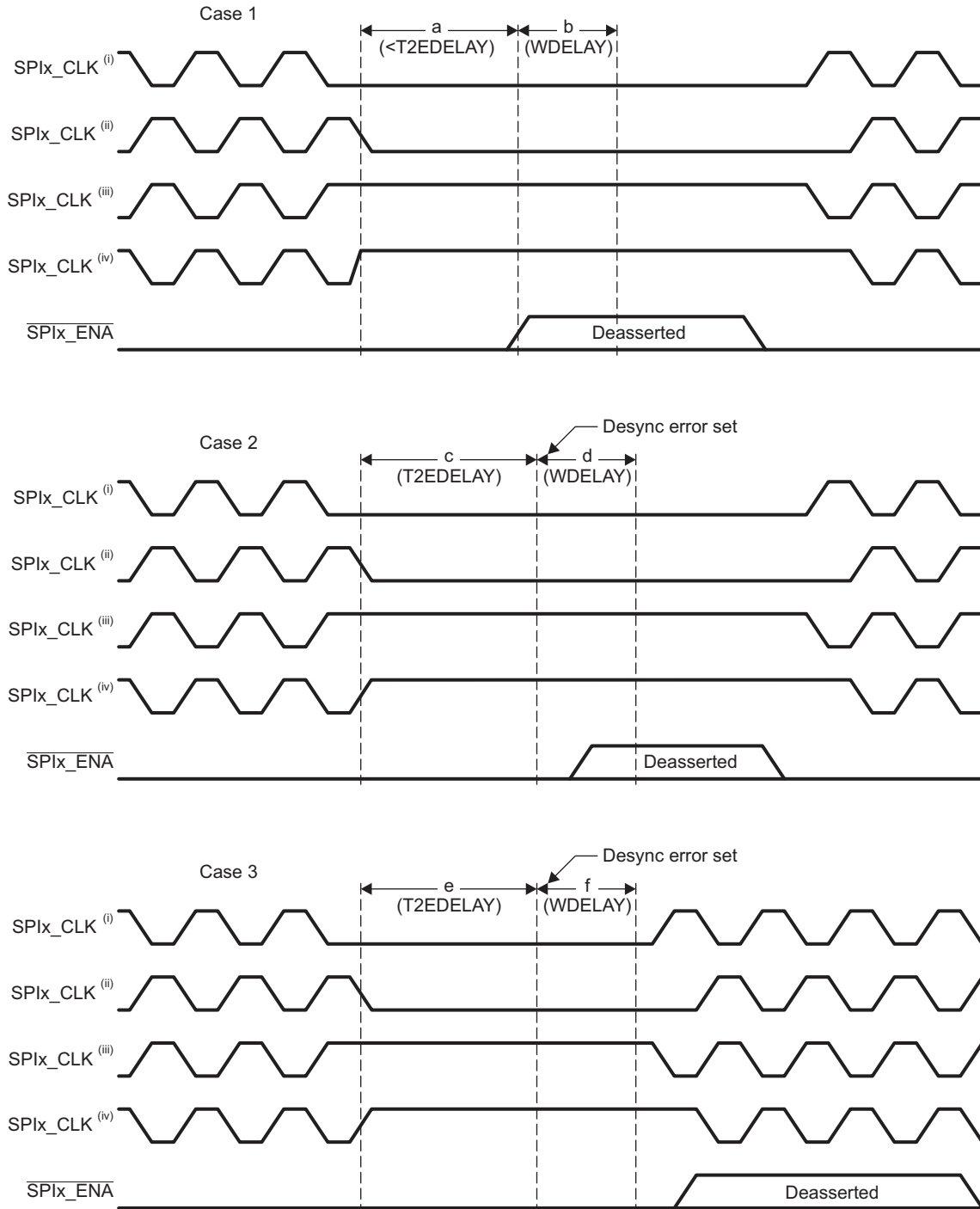
### 23.2.20.3 SPI 4-Pin with $\overline{\text{SPIx\_ENA}}$ Mode

Figure 23-15 shows the T2EDELAY and WDELAY delays in SPI 4-pin with  $\overline{\text{SPIx\_ENA}}$  master mode. T2CDELAY, C2TDELAY, and C2EDELAY are not available in this mode.

- In CASE1, the  $\overline{\text{SPIx\_ENA}}$  is deasserted during the T2EDELAY period. Consequently the T2EDELAY period is terminated early (a) and the WDELAY period begins immediately (b) if enabled. The next transfer is initiated as soon as the slave asserts  $\overline{\text{SPIx\_ENA}}$  again.
- In CASE2, the T2EDELAY period (c) completes before the  $\overline{\text{SPIx\_ENA}}$  is deasserted. As a result the DESYNC error is set. However since the  $\overline{\text{SPIx\_ENA}}$  is deasserted during the WDELAY period (d), the master delays the next transfer until the  $\overline{\text{SPIx\_ENA}}$  is asserted again.
- In CASE3, the T2EDELAY (e) and WDELAY (f) period (if enabled) both expire before the  $\overline{\text{SPIx\_ENA}}$  input is deasserted. The DESYNC error is set at the end of the T2EDELAY period (e). However in this case the master begins the next transfer immediately after it is initiated and ignores the  $\overline{\text{SPIx\_ENA}}$  during the transfer even if it is subsequently deasserted.

If the T2EDELAY delay period is disabled then the DESYNC error is not set. The SPI master behavior in this case depends on whether the  $\overline{\text{SPIx\_ENA}}$  gets deasserted during the WDELAY period (CASE2) or  $\overline{\text{SPIx\_ENA}}$  gets deasserted after the WDELAY period completes (CASE3).

**Figure 23-15. SPI 4-Pin with  $\overline{\text{SPIx\_ENA}}$  Mode Demonstrating T2EDELAY and WDELAY**





### 23.2.20.4 SPI 5-Pin Mode

Figure 23-16 shows the T2CDELAY, T2EDELAY, and WDELAY delays in SPI 5-pin master mode.

- In CASE1, the  $\overline{\text{SPIx\_EN\bar{A}}}$  is deasserted during the T2CDELAY period. However the T2CDELAY period proceeds to completion(a), the T2EDELAY period is skipped (if enabled) and the WDELAY period begins immediately (b) (if enabled). The next transfer is initiated as soon as the slave asserts  $\overline{\text{SPIx\_EN\bar{A}}}$  again.
- In CASE2, the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal is deasserted by the slave during the T2EDELAY period (d) which begins upon the completion of the T2CDELAY period (c). The deassertion of the  $\overline{\text{SPIx\_EN\bar{A}}}$  causes the T2EDELAY period to terminate early and the WDELAY period (e) begins immediately (if enabled) after the T2EDELAY period terminates. The next transfer is initiated as soon as the slave asserts  $\overline{\text{SPIx\_EN\bar{A}}}$  again.
- In CASE3, the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal is deasserted by the slave during the WDELAY period (h) which begins upon the completion of the T2CDELAY period (f) and T2EDELAY period (g). As a result the DESYNC error is set at the end of the T2EDELAY period (g). However since the  $\overline{\text{SPIx\_EN\bar{A}}}$  is deasserted during the WDELAY period (h), the master delays the next transfer until the  $\overline{\text{SPIx\_EN\bar{A}}}$  is asserted again.
- In CASE4, the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal is not deasserted until after the completion of the T2CDELAY (j), T2EDELAY (k) and WDELAY (m) (if enabled) periods. The DESYNC error is set at the end of the T2EDELAY period (k). However in this case the master begins the next transfer immediately after it is initiated and ignores the  $\overline{\text{SPIx\_EN\bar{A}}}$  during the transfer even if it is subsequently deasserted.

If the T2EDELAY delay period is disabled then the DESYNC error is not set. The SPI master behavior in this case depends on whether the  $\overline{\text{SPIx\_EN\bar{A}}}$  gets deasserted during the T2CDELAY period (CASE1), WDELAY period (CASE3) or after the WDELAY period completes (CASE4).

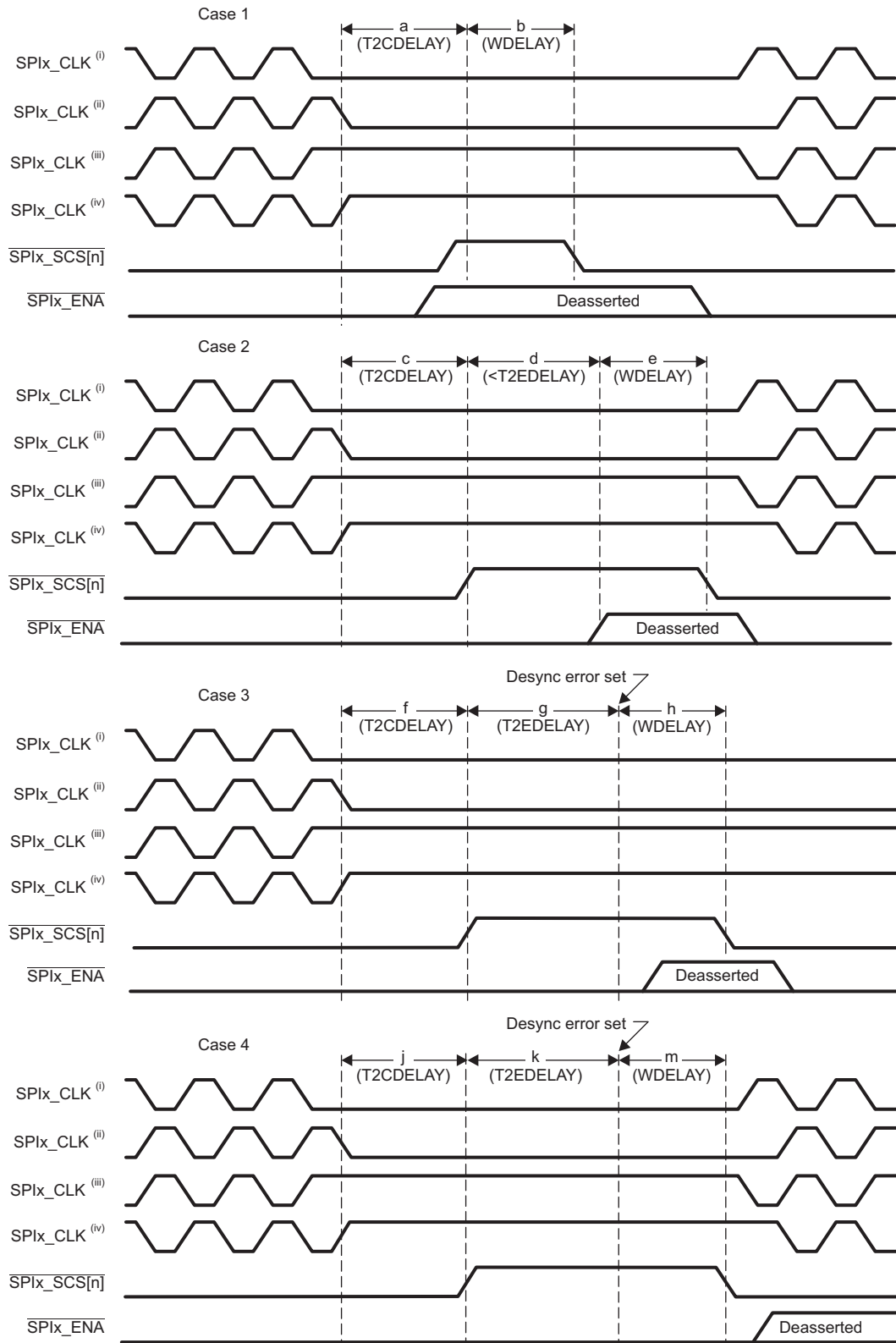
If the slave deasserts the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal before the completion of the configured master delays (T2CDELAY, T2EDELAY, WDELAY) then the master delays the next transfer until the slave asserts the  $\overline{\text{SPIx\_EN\bar{A}}}$  again. However if the slave delays the  $\overline{\text{SPIx\_EN\bar{A}}}$  deassertion until after the completion of the configured master delays then the master begins the next transfer immediately after it is initiated and ignores the  $\overline{\text{SPIx\_EN\bar{A}}}$  during the transfer even if it is subsequently deasserted.

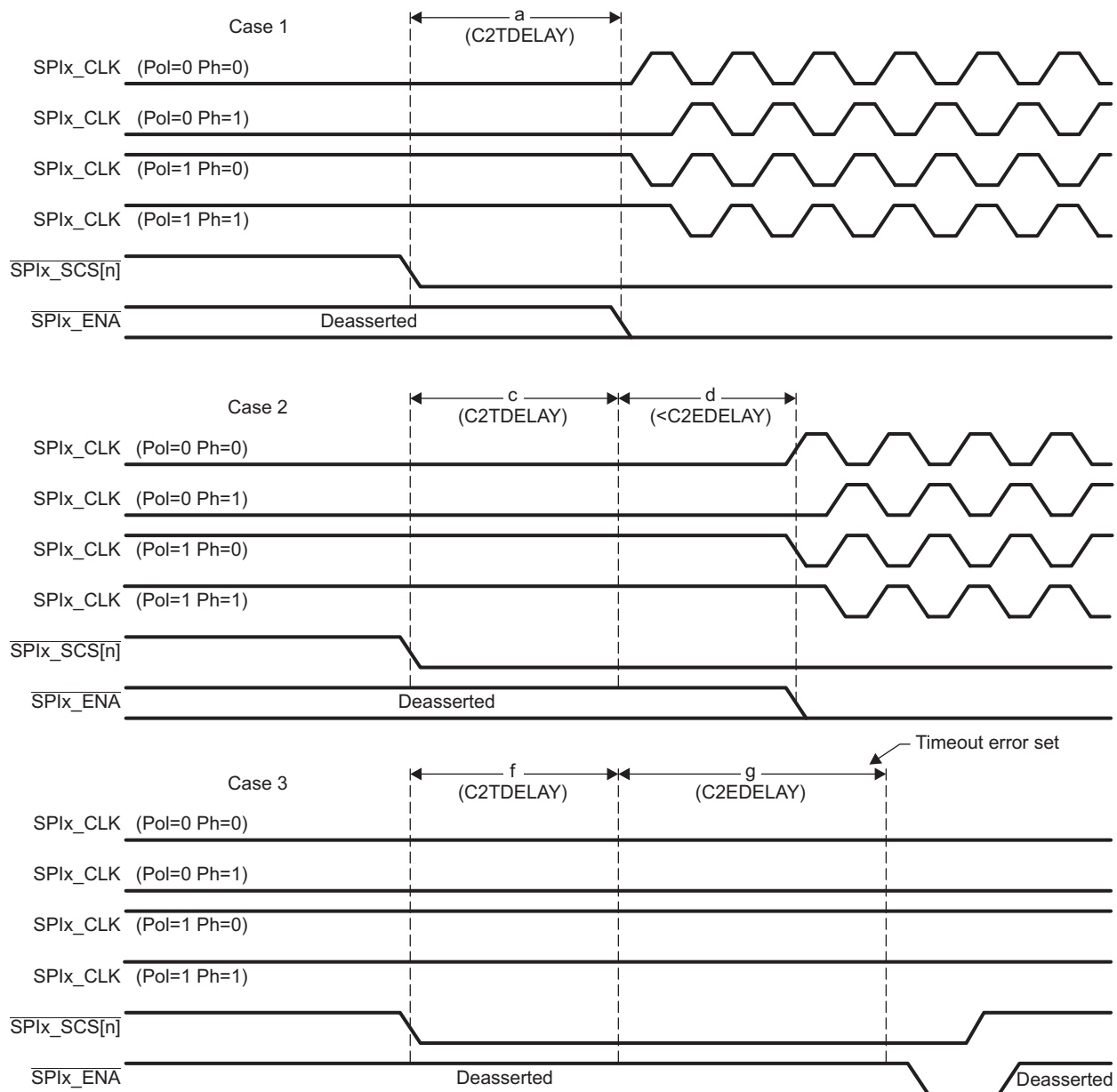
Figure 23-17 shows the C2TDELAY and C2EDELAY in SPI 5-pin master mode.

- In CASE1, the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal is asserted during the C2TDELAY period (a). However the C2TDELAY period proceeds to completion(a), the C2EDELAY period is skipped (if enabled) and the master begins generating the SPI clock for transmission.
- In CASE2, the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal is asserted during the C2EDELAY period (d) which begins upon the completion of C2TDELAY period (c). The assertion of the  $\overline{\text{SPIx\_EN\bar{A}}}$  causes the C2EDELAY period to terminate early and the master begins generating the SPI clock for transmission.
- In CASE3, the  $\overline{\text{SPIx\_EN\bar{A}}}$  signal is not asserted until after the completion of the C2TDELAY (f) and C2EDELAY (g) periods. The TIMEOUT error is set at the end of the C2EDELAY period (g). The master deasserts the  $\overline{\text{SPIx\_SCS[n]}}$  signal immediately and clears the current transmit request.

If the C2EDELAY delay period is disabled then the SPI master behavior depends on whether the  $\overline{\text{SPIx\_EN\bar{A}}}$  gets asserted during the C2TDELAY period (CASE1) or after the C2TDELAY period completes (CASE2). In latter case there is no limit on how long the master will wait for the slave to respond with  $\overline{\text{SPIx\_EN\bar{A}}}$  asserted and hence there is no limit on period 'd' shown in CASE2. Thus when C2EDELAY period is disabled the TIMEOUT error is not set.

**Figure 23-16. SPI 5-Pin Mode Demonstrating T2CDELAY, T2EDELAY, and WDELAY**



**Figure 23-17. SPI 5-Pin Mode Demonstrating C2TDELAY and C2EDELAY**


## 23.3 Registers

This section describes the SPI control, data, and pin registers. The offset is relative to the associated base address of the module. See your device-specific data manual for the memory address of these registers.

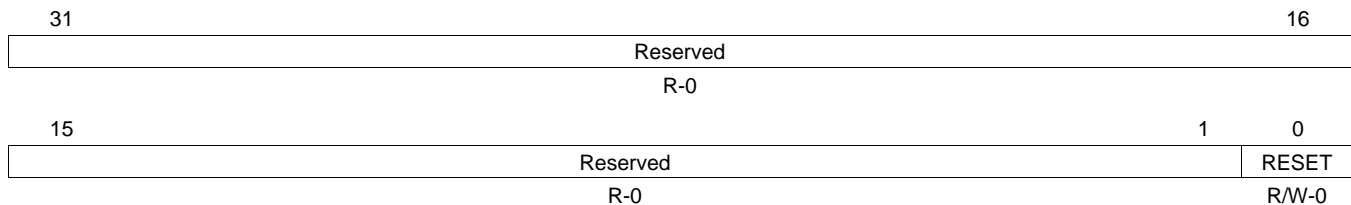
**Table 23-8. SPI Registers**

Offset Address	Acronym	Register Description	Section
0h	SPIGCR0	SPI Global Control Register 0	<a href="#">Section 23.3.1</a>
4h	SPIGCR1	SPI Global Control Register 1	<a href="#">Section 23.3.2</a>
8h	SPIINT0	SPI Interrupt Register	<a href="#">Section 23.3.3</a>
Ch	SPIILVL	SPI Interrupt Level Register	<a href="#">Section 23.3.4</a>
10h	SPIFLG	SPI Flag Register	<a href="#">Section 23.3.5</a>
14h	SPIPC0	SPI Pin Control Register 0 (Function)	<a href="#">Section 23.3.6</a>
18h	SPIPC1	SPI Pin Control Register 1 (Direction)	<a href="#">Section 23.3.7</a>
1Ch	SPIPC2	SPI Pin Control Register 2 (Input)	<a href="#">Section 23.3.8</a>
20h	SPIPC3	SPI Pin Control Register 3 (Output)	<a href="#">Section 23.3.9</a>
24h	SPIPC4	SPI Pin Control Register 4 (Set SPIPC3)	<a href="#">Section 23.3.10</a>
28h	SPIPC5	SPI Pin Control Register 5 (Clear SPIPC3)	<a href="#">Section 23.3.11</a>
38h	SPIDAT0	SPI Data Transmit Register 0	<a href="#">Section 23.3.12</a>
3Ch	SPIDAT1	SPI Data Transmit Register 1 (Data Transmit and Format Select)	<a href="#">Section 23.3.13</a>
40h	SPIBUF	SPI Receive Buffer Register	<a href="#">Section 23.3.14</a>
44h	SPIEMU	SPI Receive Emulation Register	<a href="#">Section 23.3.15</a>
48h	SPIDELAY	SPI Delay Register	<a href="#">Section 23.3.16</a>
4Ch	SPIDEF	SPI Default Chip Select Register	<a href="#">Section 23.3.17</a>
50h	SPIFMT0	SPI Data Format Register 0	<a href="#">Section 23.3.18</a>
54h	SPIFMT1	SPI Data Format Register 1	<a href="#">Section 23.3.18</a>
58h	SPIFMT2	SPI Data Format Register 2	<a href="#">Section 23.3.18</a>
5Ch	SPIFMT3	SPI Data Format Register 3	<a href="#">Section 23.3.18</a>
64h	INTVEC1	SPI Interrupt Vector Register 1	<a href="#">Section 23.3.19</a>

### 23.3.1 SPI Global Control Register 0 (SPIGCR0)

The SPI global control register 0 (SPIGCR0) is shown in [Figure 23-18](#) and described in [Table 23-9](#).

**Figure 23-18. SPI Global Control Register 0 (SPIGCR0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-9. SPI Global Control Register 0 (SPIGCR0) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zero and writes have no effect.
0	RESET	0	Reset bit for the module. This bit needs to be set to 1 before any operation on SPI can be done.
		1	SPI is in reset state.
		1	SPI is out of reset state.

### 23.3.2 SPI Global Control Register 1 (SPIGCR1)

The SPI global control register 1 (SPIGCR1) is shown in [Figure 23-19](#) and described in [Table 23-10](#).

**Figure 23-19. SPI Global Control Register 1 (SPIGCR1)**

31	25	24
Reserved R-0		ENABLE R/W-0
23	17	16
Reserved R-0		LOOPBACK R/W-0
15	9	8
Reserved R-0		POWERDOWN R/W-0
7	2	1
Reserved R-0		CLKMOD R/W-0
		0
		MASTER R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-10. SPI Global Control Register 1 (SPIGCR1) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zero and writes have no effect.
24	ENABLE	0 1	<p>SPI enable. This bit enables the SPI transfers. The other SPI configuration registers except SPIINT0.DMAREQEN should be configured before writing a 1 to this bit. This will prevent the SPI from responding to bus operations erroneously while it is in the process of being configured. The SPIINT0.DMAREQEN should be enabled after setting ENABLE. If SPIINT0.DMAREQEN is enabled before setting ENABLE then the first DMA request that occurs before the SPI is ready for data transfer may get dropped.</p> <p>When ENABLE bit is cleared to 0, the following SPI registers get forced to their default states (to 0s except for RXEMPTY bit in SPIBUF):</p> <ul style="list-style-type: none"> <li>• Both TX and RX shift registers</li> <li>• The TXDATA fields of SPIDAT0 and SPIDAT1 registers</li> <li>• All the fields of the SPIFLG register</li> <li>• Contents of SPIBUF and the internal RXBUF registers</li> </ul> <p>0 SPI is not activated for transfers.</p> <p>1 Activates SPI.</p>
23-17	Reserved	0	Reads return zero and writes have no effect.
16	LOOPBACK	0 1	<p>Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPIx_SIMO and SPIx_SOMI pins are configured with SPI functionality, then the SPIx_SIMO pin is internally connected to the SPIx_SOMI pin. The transmit data is looped back as receive data and is stored in the receive field of the concerned buffer.</p> <p>Externally, during loop-back operation, the SPIx_CLK pin outputs an inactive value, SPIx_SIMO and SPIx_SOMI pins remain in high-impedance state. The SPI has to be initialized in master mode before the loop-back can be selected. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result.</p> <p>0 Internal loop-back test mode disabled.</p> <p>1 Internal loop-back test mode enabled.</p>
15-9	Reserved	0	Reads return zero and writes have no effect.
8	POWERDOWN	0 1	<p>When active, the SPI state machine enters a power-down state.</p> <p>0 The SPI is in active mode.</p> <p>1 The SPI is in power-down mode.</p>
7-2	Reserved	0	Reads return zero and writes have no effect.

**Table 23-10. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)**

Bit	Field	Value	Description
1-0	CLKMOD,MASTER	0-3h	These two bits (CLKMOD,MASTER) determine whether the SPI operates in master or slave mode.
		0	SLAVE MODE. SPIx_CLK is an input from the master who initiates the transfers. Data is transmitted on the SPIx_SOMI pin and received on the SPIx_SIMO pin. The SPIx_SCS[n] pin is an input pin if configured as SPI slave chip select. The SPIx_ENA pin is an output pin if configured as the SPI enable pin.
		1h-2h	Reserved
		3h	MASTER MODE. SPIx_CLK is an output and the SPI initiates transfers. Data is transmitted on the SPIx_SIMO pin and received on the SPIx_SOMI pin. The SPIx_SCS[n] pin is an output pin if configured as SPI slave chip select. The SPIx_ENA pin is an input pin if configured as the SPI enable pin.

### 23.3.3 SPI Interrupt Register (SPIINT0)

The SPI interrupt register (SPIINT0) is shown in [Figure 23-20](#) and described in [Table 23-11](#).

**Figure 23-20. SPI Interrupt Register (SPIINT0)**

31										25					24								
Reserved										R-0					ENABLEHIGHZ								
23										17					16								
Reserved										R-0					DMAREQEN								
15										10					9		8						
Reserved										R-0					TXINTENA		RXINTENA						
7										6		5		4		3		2		1		0	
Reserved		OVRNINTENA		Reserved		BITERRENA		DESYNCENA		PARERRENA		TIMEOUTENA		DLNERRENA									
R-0		R/W-0		R-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0									

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-11. SPI Interrupt Register (SPIINT0) Field Descriptions**

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zero and writes have no effect.
24	ENABLEHIGHZ	0 1	<p>SPIx_ENA pin high-impedance enable. If ENABLEHIGHZ is enabled, the SPIx_ENA pin (when it is configured as a WAIT functional output signal in a slave SPI) is forced to place it is output in high-impedance when not driving a low signal. If ENABLEHIGHZ is disabled, then the pin will output both a high and a low signal.</p> <p>0 SPIx_ENA pin is pulled high when not active.</p> <p>1 SPIx_ENA pin remains in high-impedance when not active.</p>
23-17	Reserved	0	Reads return zero and writes have no effect.
16	DMAREQEN	0 1	<p>DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. Set DMAREQEN only after setting the SPIGCR1.ENABLE bit to 1.</p> <p>0 DMA is not used.</p> <p>1 DMA requests will be generated.</p> <p><b>Note:</b> A transmit DMA request will be generated each time a transmit data is copied to the shift register either from TXBUF or directly from SPIDAT0/SPIDAT1.</p> <p><b>Note:</b> A receive DMA request will be generated each time a received data is copied to SPIBUF register either from RXBUF or directly from the shift register.</p>
15-10	Reserved	0	Reads return zero and writes have no effect.
9	TXINTENA	0 1	<p>An interrupt is to be generated every time data is written to the shift register, so that a new data can be written to TXBUF. Setting this bit will generate an interrupt if the SPIFLG.TXINTFLG bit is set to 1.</p> <p>0 No interrupt will be generated upon SPIFLG.TXINTFLG being set to 1.</p> <p>1 Interrupt will be generated upon SPIFLG.TXINTFLG being set to 1.</p>
8	RXINTENA	0 1	<p>Receive interrupt enable. An interrupt is to be generated when the SPIFLG.RXINTFLAG bit is set.</p> <p>0 Interrupt will not be generated.</p> <p>1 Interrupt will be generated.</p>
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTENA	0 1	<p>Overrun interrupt enable. An interrupt is to be generated when the SPIFLG.OVRNINTFLG bit is set. The overrun interrupt is not useful if receive data is serviced with CPU interrupts because the overrun and receive events share a common level interrupt signal.</p> <p>0 Overrun interrupt will not be generated.</p> <p>1 Overrun interrupt will be generated.</p>
5	Reserved	0	Reads return zero and writes have no effect.

**Table 23-11. SPI Interrupt Register (SPIINT0) Field Descriptions (continued)**

Bit	Field	Value	Description
4	BITERRENA	0 1	Enables interrupt on bit error. An interrupt is to be generated when the SPIFLG.BITERRFLG is set. No interrupt asserted upon bit error. Enables an interrupt on a bit error.
3	DESYNCENA	0 1	Enables interrupt on desynchronized slave. DESYNCENA is used in master mode only. The desynchronization monitor is active in master mode for the 4-pin with enable and 5-pin options. An interrupt is to be generated when the SPIFLG.DESYNCF LG is set. No interrupt asserted upon desynchronization error. Enables an interrupt on desynchronization of the slave.
2	PARERRENA	0 1	Enables interrupt on parity error. An interrupt is to be generated when the SPIFLG.PARERRFLG is set. No interrupt asserted upon parity error. Enables an interrupt on a parity error.
1	TIMEOUTENA	0 1	Enables interrupt on $\overline{\text{SPIx\_EN A}}$ signal time-out. An interrupt is to be generated when SPIFLG.TIMEOUTFLG is set. No interrupt asserted upon $\overline{\text{SPIx\_EN A}}$ signal time-out. Enables an interrupt on a time-out of the $\overline{\text{SPIx\_EN A}}$ signal.
0	DLENERRENA	0 1	Data length error interrupt enable. A data length error occurs under the following conditions. Master: In a 4-pin with $\overline{\text{SPIx\_EN A}}$ mode or 5-pin mode, if the $\overline{\text{SPIx\_EN A}}$ pin from the slave is deasserted before the master has completed its transfer, the data length error is set. That is, if the character length counter has not overflowed while $\overline{\text{SPIx\_EN A}}$ deassertion is detected, then it means that the slave has neither received full data from the master nor has it transmitted complete data. Slave: In a 4-pin with chip select mode or 5-pin mode, if the incoming valid $\overline{\text{SPIx\_SCS[n]}}$ pin is deactivated before the character length counter overflows, then data length error is set. No interrupt is generated upon data length error. Enables an interrupt when data length error occurs.



### 23.3.4 SPI Interrupt Level Register (SPILVL)

The SPI interrupt level register (SPILVL) is shown in [Figure 23-21](#) and described in [Table 23-12](#).

**Figure 23-21. SPI Interrupt Level Register (SPILVL)**

	Reserved	
	R-0	
15	Reserved	8
	R-0	R/W-0
7	OVRNINTLVL	0
	R/W-0	R/W-0
6	Reserved	1
	R-0	R/W-0
5	BITERRLVL	2
	R/W-0	R/W-0
4	DESYNCLVL	3
	R/W-0	R/W-0
3	PARERRLVL	4
	R/W-0	R/W-0
2	TIMEOUTLVL	5
	R/W-0	R/W-0
1	DLENERRLVL	6
	R/W-0	R/W-0
0	Reserved	7
	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-12. SPI Interrupt Level Register (SPILVL) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	0	Reads return zero and writes have no effect.
9	TXINTLVL	0	Reserved
		1	Transmit interrupt is mapped to interrupt line INT1.
8	RXINTLVL	0	Reserved
		1	Receive interrupt is mapped to interrupt line INT1.
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTLVL	0	Reserved
		1	Receive overrun interrupt is mapped to interrupt line INT1.
5	Reserved	0	Reads return zero and writes have no effect.
4	BITERRLVL	0	Reserved
		1	Bit error interrupt is mapped to interrupt line INT1.
3	DESYNCLVL	0	Reserved
		1	An interrupt due to desynchronization of the slave is mapped to interrupt line INT1.
2	PARERRLVL	0	Reserved
		1	A parity error interrupt is mapped to interrupt line INT1.
1	TIMEOUTLVL	0	Reserved
		1	An interrupt on a time-out of the $\overline{\text{SPIx\_ENA}}$ signal is mapped to interrupt line INT1.
0	DLENERRLVL	0	Reserved
		1	An interrupt on data length error is mapped to interrupt line INT1.

### 23.3.5 SPI Flag Register (SPIFLG)

The SPI flag register (SPIFLG) is shown in [Figure 23-22](#) and described in [Table 23-13](#).

**Figure 23-22. SPI Flag Register (SPIFLG)**

Reserved							
R-100h							
Reserved				TXINTFLG		RXINTFLG	
R-0				R-0		R/WC-0	
7	6	5	4	3	2	1	0
Reserved	OVRNINTFLG	Reserved	BITERRFLG	DESYNCFLG	PARERRFLG	TIMEOUTFLG	DLENERRFLG
R-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear bit; -n = value after reset

**Table 23-13. SPI Flag Register (SPIFLG) Field Descriptions**

Bit	Field	Value	Description
31-10	Reserved	4000h	Reads return default value and writes have no effect.
9	TXINTFLG	0 1	<p>Transmitter empty interrupt flag. Serves as an interrupt flag indicating that the transmit buffer (TXBUF) is empty and a new data can be written to it. This flag is set when a data is copied to the shift register either directly or from the TXBUF register. This bit is cleared by one of following ways:</p> <ul style="list-style-type: none"> <li>Writing a new data to either SPIDAT0 or SPIDAT1</li> <li>Writing a 0 to SPIGCR1.ENABLE</li> </ul> <p>0 Transmit buffer is now full. No interrupt pending for transmitter empty.</p> <p>1 Transmit buffer is empty. An interrupt is pending to fill the transmitter.</p>
8	RXINTFLG	0 1	<p>Receiver full interrupt flag. This flag is set when a word is received and copied into the buffer register (SPIBUF). This bit is cleared under the following ways:</p> <ul style="list-style-type: none"> <li>Reading the SPIBUF register. During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit.</li> <li>Reading INTVEC1 register when there is a receive buffer full interrupt</li> <li>Writing a 1 to this bit</li> <li>Writing a 0 to SPIGCR1.ENABLE</li> <li>System reset</li> </ul> <p>0 No new received data pending. Receive buffer is empty.</p> <p>1 A newly received data is ready to be read. Receive buffer is full.</p> <p><b>Note:</b> Clearing RXINTFLG bit by writing a 1 before reading the SPIBUF sets the RXEMPTY bit of the SPIBUF register too. This way, one can ignore a received data. However, if the internal RXBUF is already full, the data from RXBUF will be copied to SPIBUF and the RXEMPTY bit will be cleared again. The SPIBUF contents should be read first if this situation needs to be avoided.</p>
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTFLG	0 1	<p>Receiver overrun flag. The bit is set when a receive operation completes before the previous character has been read from the receive buffer. The bit indicates that the last received character has been overwritten and therefore lost. This bit is cleared under the following conditions:</p> <ul style="list-style-type: none"> <li>Reading INTVEC1 register when there is a receive buffer overrun interrupt</li> <li>Writing a 1 to this bit</li> </ul> <p>0 Overrun condition did not occur.</p> <p>1 Overrun condition has occurred.</p> <p><b>Note:</b> Reading SPIBUF register does not clear the OVRNINTFLG bit. If an overrun interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF.</p> <p><b>Note:</b> A special condition under which OVRNINTFLG flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR and DLENERR occur, then OVRNINTFLG will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.</p>

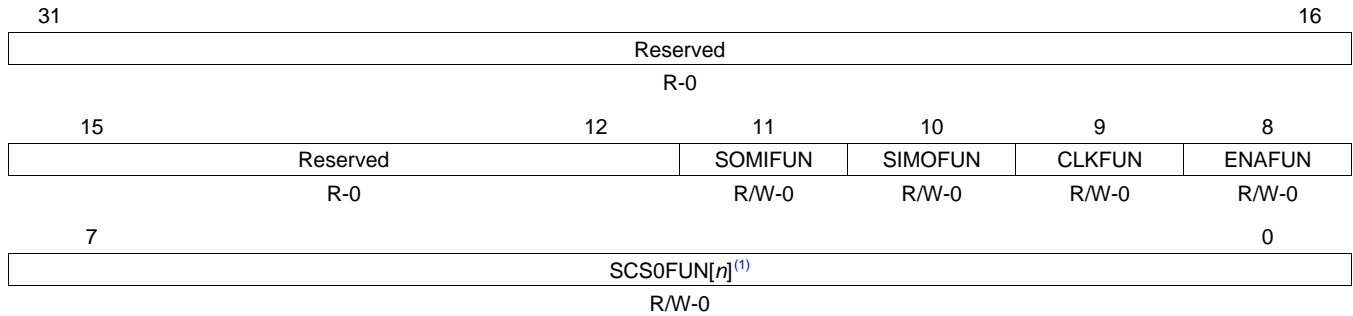
**Table 23-13. SPI Flag Register (SPIFLG) Field Descriptions (continued)**

Bit	Field	Value	Description
5	Reserved	0	Reads return zero and writes have no effect.
4	BITERRFLG	0	This bit is set when a mismatch of internal transmit data and transmitted data is detected. The SPI samples the signal of the transmit pin (master: SPIx_SIMO, slave: SPIx_SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value a bit error is detected and the flag is set. A possible reason for a bit error can be a too high bit rate/capacitive load or another master/slave trying to transmit at the same time. This flag can be cleared by one of the following ways: <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIGCR1.ENABLE bit to 0.</li> </ul>
		0	No bit error occurred.
		1	A bit error occurred.
3	DESYNCFLG	0	Desynchronization of slave device. Desynchronization monitor is active in master mode only. The master monitors the $\overline{\text{SPIx\_EN\bar{A}}}$ signal coming from the slave device and sets the DESYNCFLG bit if the $\overline{\text{SPIx\_EN\bar{A}}}$ signal is not deasserted after the last bit is transmitted plus $t_{\text{TZDELAY}}$ . Desynchronization can occur if a slave device misses a clock edge coming from the master. This flag can be cleared by one of the following ways: <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIGCR1.ENABLE bit to 0.</li> </ul>
		0	No slave desynchronization detected.
		1	Slave is desynchronized <b>Note:</b> Inconsistency of DESYNCFLG in SPI. Due to the nature of this error, under some circumstances it is possible for a desynchronized error detected for the previous buffer to be visible in the current buffer. This is due to the fact that receive completion flag/interrupt will be generated when the buffer transfer is completed. But deince will be detected after the buffer transfer is completed. So, if CPU/DMA reads the received data quickly when an receive interrupt is detected, then the status flag may not reflect the correct deince condition.
2	PARERRFLG	0	Calculated parity differs from received parity bit. If the parity generator is enabled an even or odd parity bit is added at the end of a data word. During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARERRFLG flag is set. This flag can be cleared by one of the following ways: <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIGCR1.ENABLE bit to 0.</li> </ul>
		0	No parity error detected.
		1	A parity error occurred.
1	TIMEOUTFLG	0	Time-out due to non-activation of $\overline{\text{SPIx\_EN\bar{A}}}$ signal. This flag is applicable only for the master mode. The SPI generates a time-out because the slave hasn't responded in time by activating the $\overline{\text{SPIx\_EN\bar{A}}}$ signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUTFLG flag is set. This flag can be cleared by one of the following ways: <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIGCR1.ENABLE bit to 0.</li> </ul>
		0	No $\overline{\text{SPIx\_EN\bar{A}}}$ signal time-out occurred.
		1	An $\overline{\text{SPIx\_EN\bar{A}}}$ signal time-out occurred.
0	DLENERRFLG	0	Data length error flag. This flag can be cleared by one of the following ways: <ul style="list-style-type: none"> <li>• Write a 1 to this bit.</li> <li>• Set SPIGCR1.ENABLE bit to 0.</li> </ul>
		0	No data length error has occurred.
		1	A data length error has occurred.

### 23.3.6 SPI Pin Control Register 0 (SPIPC0)

The SPI pin control register 0 (SPIPC0) is shown in [Figure 23-23](#) and described in [Table 23-14](#).

**Figure 23-23. SPI Pin Control Register 0 (SPIPC0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

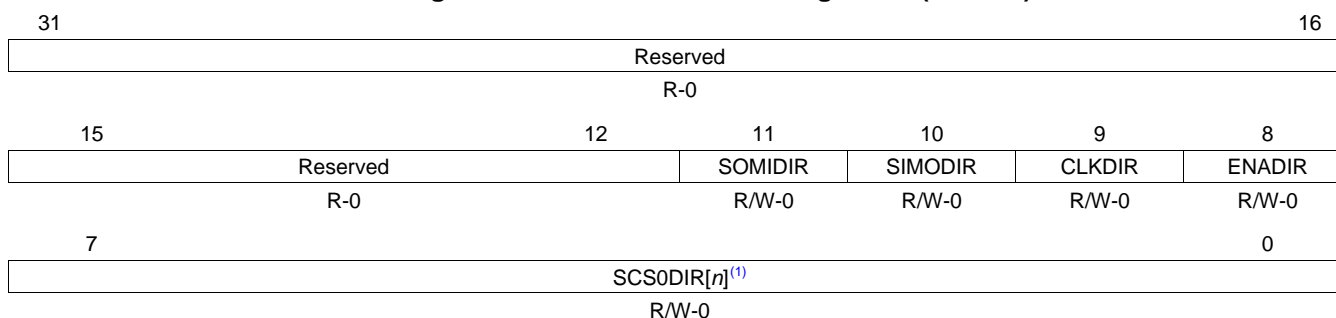
**Table 23-14. SPI Pin Control Register 0 (SPIPC0) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIFUN	0 1	Slave out, master in pin function. This bit determines whether the $\overline{\text{SPIx\_SOMI}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin. 0 $\overline{\text{SPIx\_SOMI}}$ pin is a GPIO pin. 1 $\overline{\text{SPIx\_SOMI}}$ pin is a SPI functional pin.
10	SIMOFUN	0 1	Slave in, master out pin function. This bit determines whether the $\overline{\text{SPIx\_SIMO}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin. 0 $\overline{\text{SPIx\_SIMO}}$ pin is a GPIO pin. 1 $\overline{\text{SPIx\_SIMO}}$ pin is a SPI functional pin.
9	CLKFUN	0 1	SPI clock pin function. This bit determines whether the $\overline{\text{SPIx\_CLK}}$ pin is to be used as a general-purpose I/O pin, or as a SPI functional pin. 0 $\overline{\text{SPIx\_CLK}}$ pin is a GPIO pin. 1 $\overline{\text{SPIx\_CLK}}$ pin is a SPI functional pin.
8	ENAFUN	0 1	SPI enable pin function. This bit determines whether the $\overline{\text{SPIx\_ENA}}$ pin is to be used as a general-purpose I/O pin, or as a SPI functional pin. 0 $\overline{\text{SPIx\_ENA}}$ pin is a GPIO pin. 1 $\overline{\text{SPIx\_ENA}}$ pin is a SPI functional pin.
7-0	SCS0FUN[n]	0 1	SPI chip select pin <i>n</i> function. This bit determines whether the $\overline{\text{SPIx\_SCS}}[n]$ pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.  Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0. 0 $\overline{\text{SPIx\_SCS}}[n]$ pin is a GPIO pin. 1 $\overline{\text{SPIx\_SCS}}[n]$ pin is a SPI functional pin.

### 23.3.7 SPI Pin Control Register 1 (SPIPC1)

The SPI pin control register 1 (SPIPC1) is shown in [Figure 23-24](#) and described in [Table 23-15](#).

**Figure 23-24. SPI Pin Control Register 1 (SPIPC1)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

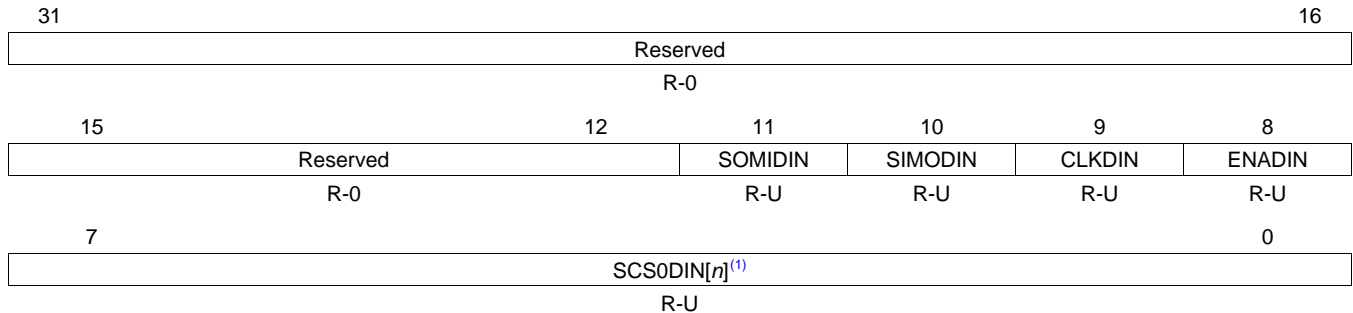
**Table 23-15. SPI Pin Control Register 1 (SPIPC1) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDIR	0 1	<p><math>\overline{\text{SPIx\_SOMI}}</math> pin direction. Controls the direction of the <math>\overline{\text{SPIx\_SOMI}}</math> pin when it is used as a general-purpose I/O pin. If the <math>\overline{\text{SPIx\_SOMI}}</math> pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as master or slave.</p> <p>0 <math>\overline{\text{SPIx\_SOMI}}</math> pin is an input. 1 <math>\overline{\text{SPIx\_SOMI}}</math> pin is an output.</p>
10	SIMODIR	0 1	<p><math>\overline{\text{SPIx\_SIMO}}</math> pin direction. Controls the direction of the <math>\overline{\text{SPIx\_SIMO}}</math> pin when it is used as a general-purpose I/O pin. If the <math>\overline{\text{SPIx\_SIMO}}</math> pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as master or slave.</p> <p>0 <math>\overline{\text{SPIx\_SIMO}}</math> pin is an input. 1 <math>\overline{\text{SPIx\_SIMO}}</math> pin is an output.</p>
9	CLKDIR	0 1	<p><math>\overline{\text{SPIx\_CLK}}</math> pin direction. Controls the direction of the <math>\overline{\text{SPIx\_CLK}}</math> pin when it is used as a general-purpose I/O pin. If the <math>\overline{\text{SPIx\_CLK}}</math> pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as master or slave.</p> <p>0 <math>\overline{\text{SPIx\_CLK}}</math> pin is an input. 1 <math>\overline{\text{SPIx\_CLK}}</math> pin is an output.</p>
8	ENADIR	0 1	<p><math>\overline{\text{SPIx\_ENA}}</math> pin direction. Controls the direction of the <math>\overline{\text{SPIx\_ENA}}</math> pin when it is used as a general-purpose I/O pin. If the <math>\overline{\text{SPIx\_ENA}}</math> pin is used as a SPI functional pin, then the I/O direction is determined by whether the SPI is configured as master or slave.</p> <p>0 <math>\overline{\text{SPIx\_ENA}}</math> pin is an input. 1 <math>\overline{\text{SPIx\_ENA}}</math> pin is an output.</p>
7-0	SCS0DIR[n]	0 1	<p><math>\overline{\text{SPIx\_SCS}}[n]</math> pin direction. Controls the direction of the <math>\overline{\text{SPIx\_SCS}}[n]</math> pin when it is used as a general-purpose I/O pin. If the <math>\overline{\text{SPIx\_SCS}}[n]</math> pin is used as a SPI functional pin, then the I/O direction is determined by whether the SPI is configured as master or slave.</p> <p>Not all devices support multiple slave chip select (<math>\overline{\text{SPIx\_SCS}}[n]</math>) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.</p> <p>0 <math>\overline{\text{SPIx\_SCS}}[n]</math> pin is an input. 1 <math>\overline{\text{SPIx\_SCS}}[n]</math> pin is an output.</p>

### 23.3.8 SPI Pin Control Register 2 (SPIPC2)

The SPI pin control register 2 (SPIPC2) is shown in [Figure 23-25](#) and described in [Table 23-16](#).

**Figure 23-25. SPI Pin Control Register 2 (SPIPC2)**



LEGEND: R = Read only; U = Undefined; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

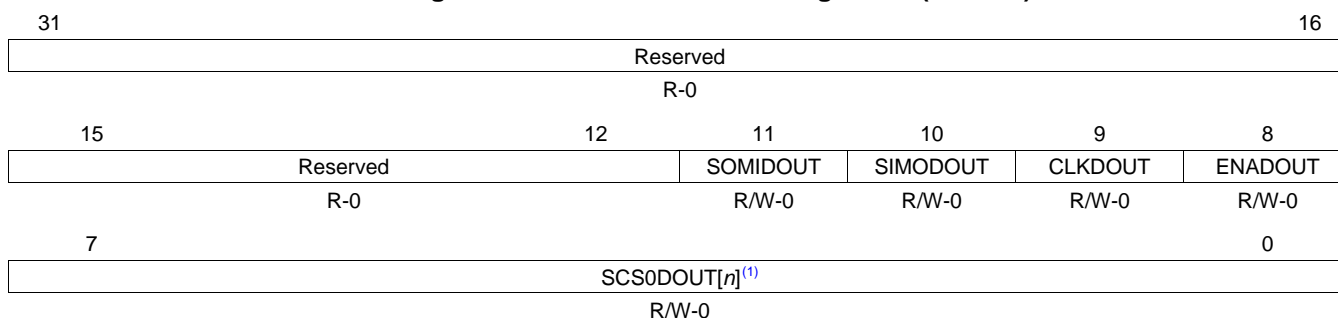
**Table 23-16. SPI Pin Control Register 2 (SPIPC2) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDIN		SPIx_SOMI data in. This bit reflects the value of the SPIx_SOMI pin.
		0	Current value of SPIx_SOMI pin is logic 0.
		1	Current value of SPIx_SOMI pin is logic 1.
10	SIMODIN		SPIx_SIMO data in. This bit reflects the value of the SPIx_SIMO pin.
		0	Current value of SPIx_SIMO pin is logic 0.
		1	Current value of SPIx_SIMO pin is logic 1.
9	CLKDIN		Clock data in. This bit reflects the value of the SPIx_CLK pin.
		0	Current value of SPIx_CLK pin is logic 0.
		1	Current value of SPIx_CLK pin is logic 1.
8	ENADIN		$\overline{\text{SPIx\_ENA}}$ data in. This bit reflects the value of the $\overline{\text{SPIx\_ENA}}$ pin.
		0	Current value of $\overline{\text{SPIx\_ENA}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPIx\_ENA}}$ pin is logic 1.
7-0	SCS0DIN[n]		$\overline{\text{SPIx\_SCS}}[n]$ data in. This bit reflects the value of the $\overline{\text{SPIx\_SCS}}[n]$ pin. Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		0	Current value of $\overline{\text{SPIx\_SCS}}[n]$ pin is logic 0.
		1	Current value of $\overline{\text{SPIx\_SCS}}[n]$ pin is logic 1.

### 23.3.9 SPI Pin Control Register 3 (SPIPC3)

The SPI pin control register 3 (SPIPC3) is shown in [Figure 23-26](#) and described in [Table 23-17](#).

**Figure 23-26. SPI Pin Control Register 3 (SPIPC3)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

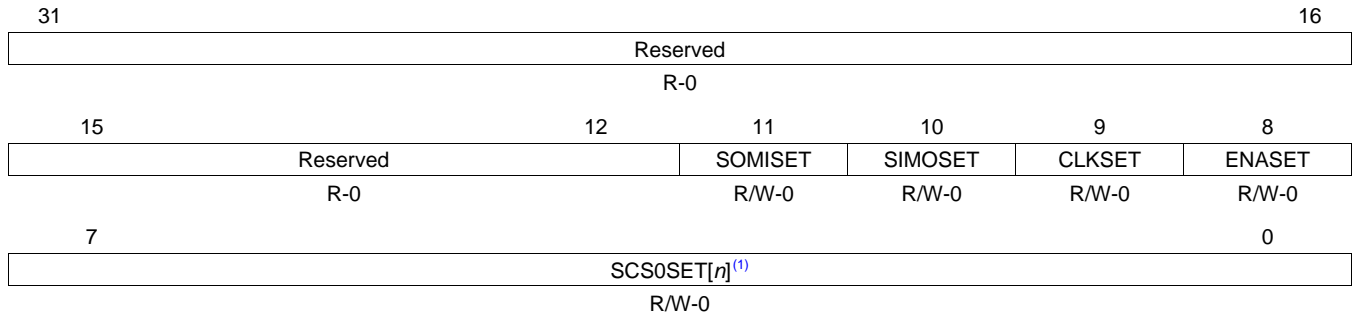
**Table 23-17. SPI Pin Control Register 3 (SPIPC3) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDOUT	0 1	<p><math>\overline{\text{SPIx\_SOMI}}</math> data out write. This bit is only active when the <math>\overline{\text{SPIx\_SOMI}}</math> pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.</p> <p>0 Current value of <math>\overline{\text{SPIx\_SOMI}}</math> pin is logic 0. 1 Current value of <math>\overline{\text{SPIx\_SOMI}}</math> pin is logic 1.</p>
10	SIMODOUT	0 1	<p><math>\overline{\text{SPIx\_SIMO}}</math> data out write. This bit is only active when the <math>\overline{\text{SPIx\_SIMO}}</math> pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.</p> <p>0 Current value of <math>\overline{\text{SPIx\_SIMO}}</math> pin is logic 0. 1 Current value of <math>\overline{\text{SPIx\_SIMO}}</math> pin is logic 1.</p>
9	CLKDOUT	0 1	<p><math>\overline{\text{SPIx\_CLK}}</math> data out write. This bit is only active when the <math>\overline{\text{SPIx\_CLK}}</math> pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.</p> <p>0 Current value of <math>\overline{\text{SPIx\_CLK}}</math> pin is logic 0. 1 Current value of <math>\overline{\text{SPIx\_CLK}}</math> pin is logic 1.</p>
8	ENADOUT	0 1	<p><math>\overline{\text{SPIx\_EN\overline{A}}}</math> data out write. Only active when the <math>\overline{\text{SPIx\_EN\overline{A}}}</math> pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.</p> <p>0 Current value of <math>\overline{\text{SPIx\_EN\overline{A}}}</math> pin is logic 0. 1 Current value of <math>\overline{\text{SPIx\_EN\overline{A}}}</math> pin is logic 1.</p>
7-0	SCS0DOUT[n]	0 1	<p><math>\overline{\text{SPIx\_SCS}}[n]</math> data out write. Only active when the <math>\overline{\text{SPIx\_SCS}}[n]</math> pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin <i>n</i>.</p> <p>Not all devices support multiple slave chip select (<math>\overline{\text{SPIx\_SCS}}[n]</math>) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.</p> <p>0 Current value of <math>\overline{\text{SPIx\_SCS}}[n]</math> pin is logic 0. 1 Current value of <math>\overline{\text{SPIx\_SCS}}[n]</math> pin is logic 1.</p>

### 23.3.10 SPI Pin Control Register 4 (SPIPC4)

The SPI pin control register 4 (SPIPC4) is shown in [Figure 23-27](#) and described in [Table 23-18](#).

**Figure 23-27. SPI Pin Control Register 4 (SPIPC4)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

**Table 23-18. SPI Pin Control Register 4 (SPIPC4) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMISET	Write 0 Write 1	SPIx_SOMI data out set. This bit is only active when the SPIx_SOMI pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SOMI pin. No effect SPIPC3.SOMIDOUT is set to 1.
10	SIMOSET	Write 0 Write 1	SPIx_SIMO data out set. This bit is only active when the SPIx_SIMO pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SIMO pin. No effect SPIPC3.SIMODOUT is set to 1.
9	CLKSET	Write 0 Write 1	SPIx_CLK data out set. This bit is only active when the SPIx_CLK pin is configured as a general-purpose output pin. Reads return the value of the SPIx_CLK pin. No effect SPIPC3.CLKDOUT is set to 1.
8	ENASET	Write 0 Write 1	$\overline{\text{SPIx\_EN}}[n]$ data out set. This bit is only active when the $\overline{\text{SPIx\_EN}}[n]$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx\_EN}}[n]$ pin. No effect. SPIPC3.ENADOUT is set to 1.
7-0	SCS0SET[n]	Write 0 Write 1	$\overline{\text{SPIx\_SCS}}[n]$ data out set. This bit is only active when the $\overline{\text{SPIx\_SCS}}[n]$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx\_SCS}}[n]$ pin. Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0. No effect SPIPC3.SCS0DOUT[n] is set to 1.



### 23.3.11 SPI Pin Control Register 5 (SPIPC5)

The SPI pin control register 5 (SPIPC5) is shown in [Figure 23-28](#) and described in [Table 23-19](#).

**Figure 23-28. SPI Pin Control Register 5 (SPIPC5)**

31	Reserved				16
R-0					
15	12	11	10	9	8
Reserved	SOMICLR	SIMOCLR	CLKCLR	ENACLR	
R-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	SCS0CLR[[n] <sup>(1)</sup>				0
R/W-0					

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

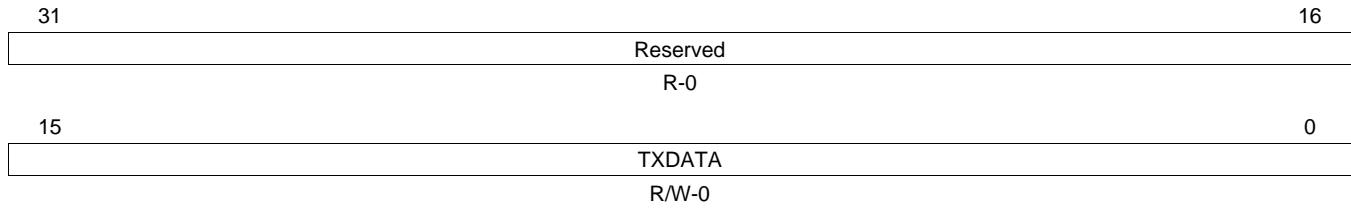
**Table 23-19. SPI Pin Control Register 5 (SPIPC5) Field Descriptions**

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMICLR	Write 0 Write 1	SPIx_SOMI data out clear. This bit is only active when the SPIx_SOMI pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SOMI pin. No effect. SPIPC3.SOMIDOUT is cleared to 0.
10	SIMOCLR	Write 0 Write 1	SPIx_SIMO data out clear. This bit is only active when the SPIx_SIMO pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SIMO pin. No effect. SPIPC3.SIMODOUT is cleared to 0.
9	CLKCLR	Write 0 Write 1	SPIx_CLK data out clear. This bit is only active when the SPIx_CLK pin is configured as a general-purpose output pin. Reads return the value of the SPIx_CLK pin. No effect. SPIPC3.CLKDOUT is cleared to 0.
8	ENACLR	Write 0 Write 1	$\overline{\text{SPIx\_ENA}}$ data out clear. This bit is only active when the $\overline{\text{SPIx\_ENA}}$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx\_ENA}}$ pin. No effect. SPIPC3.ENADOUT is cleared to 0.
7-0	SCS0CLR[n]	Write 0 Write 1	$\overline{\text{SPIx\_SCS}}[n]$ data out clear. This bit is only active when the $\overline{\text{SPIx\_SCS}}[n]$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx\_SCS}}[n]$ pin. Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0. No effect. SPIPC3.SCS0DOUT[n] is cleared to 0.

### 23.3.12 SPI Transmit Data Register 0 (SPIDAT0)

The SPI transmit data register 0 (SPIDAT0) is shown in [Figure 23-29](#) and described in [Table 23-20](#).

**Figure 23-29. SPI Data Register 0 (SPIDAT0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-20. SPI Data Register 0 (SPIDAT0) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zero and writes have no effect.
15-0	TXDATA	0-FFFFh	<p>SPI transmit data. When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values. SPIGCR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPIGCR1.ENABLE forces the TXDATA field to 0.</p> <p><b>Note:</b> Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT0 register.</p> <p><b>Note:</b> The default data format control register for SPIDAT0 is SPIFMT0. However, it is possible to reprogram the DFSEL field of SPIDAT1 before using SPIDAT0, to select a different SPIFMT<sub>n</sub> register.</p>

### 23.3.13 SPI Transmit Data Register 1 (SPIDAT1)

The SPI transmit data register (SPIDAT1) is shown in [Figure 23-30](#) and described in [Table 23-21](#).

**Figure 23-30. SPI Data Register 1 (SPIDAT1)**

31	29	28	27	26	25	24
Reserved		CSHOLD	Reserved	WDEL	DFSEL	
R-0		R/W-0	R-0	R/W-0	R/W-0	
23						16
CSNR[n] <sup>(1)</sup>						
R/W-0						
15						0
TXDATA						
R/W-0						

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins.

**Table 23-21. SPI Data Register 1 (SPIDAT1) Field Descriptions**

Bit	Field	Value	Description
31-29	Reserved	0	Reads return zero and writes have no effect.
28	CSHOLD	0 1	Chip select hold mode. The CSHOLD bit is supported in master mode only. In slave mode, this bit is ignored. CSHOLD defines the behavior of the chip select line at the end of a data transfer. 0 The chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. 1 The chip select signal is held active at the end of a transfer until a control field with new data and control information is loaded into SPIDAT1. If the new chip select hold information equals the previous one, the active chip select signal is extended until the end of transfer with CSHOLD cleared.
27	Reserved	0	Reads return zero and writes have no effect.
26	WDEL	0 1	Enable the delay counter at the end of the current transaction. The WDEL bit is supported in master mode only. In slave mode, this bit is ignored. 0 No delay will be inserted. However, $\overline{\text{SPIx\_SCS}}[n]$ pin will still be deactivated for at least 2 SPI module clock cycles if CSHOLD = 0. 1 After a transaction, SPIFMTn.WDELAY of the selected data format will be loaded into the delay counter. No transaction will be performed until the SPIFMTn.WDELAY counter overflows. The $\overline{\text{SPIx\_SCS}}[n]$ pin will be deactivated for at least (WDELAY + 2) × SPI module clock period.
25-24	DFSEL	0-3h 0 1h 2h 3h	Data word format select 0 Data word format 0 is selected 1h Data word format 1 is selected 2h Data word format 2 is selected 3h Data word format 3 is selected  <b>Note: Preselecting a Format Register.</b> Writing to just the control field (using byte writes) does not initiate any SPI transfer in master mode. This feature can be used to set up SPIx_CLK phase or polarity before actually starting the transfer by just updating the DFSEL fields in the control field to select the required phase/polarity combination.
23-16	CSNR[n]	0 1	Chip select number. The CSNR field defines the state of the $\overline{\text{SPIx\_SCS}}[n]$ pins during a master data transfer. The value of the CSNR field is driven directly on the $\overline{\text{SPIx\_SCS}}[n]$ pins. Each bit in the CSNR field corresponds to an $\overline{\text{SPIx\_SCS}}[n]$ pin, for example, CSNR[0] corresponds to $\overline{\text{SPIx\_SCS}}[0]$ (see your device-specific data manual to determine how many SPI pins are available on your device).  The state of the chip select pins when no transmissions are active is specified through the CSDEF field in the SPI default chip select register (SPIDEF). The chip select pins can remain in their active state by setting the CSHOLD bit to 1. When the SPI is configured in slave mode, this field must be written as 00h. 0 $\overline{\text{SPIx\_SCS}}[n]$ pin is driven low. 1 $\overline{\text{SPIx\_SCS}}[n]$ pin is driven high.

**Table 23-21. SPI Data Register 1 (SPIDAT1) Field Descriptions (continued)**

Bit	Field	Value	Description
15-0	TXDATA	0-FFFFh	<p>Transfer data. When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values.</p> <p>SPIGCR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPIGCR1.ENABLE forces the lower 16 bits of the SPIDAT1 to 0.</p> <p><b>Note:</b> Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT1.</p>

### 23.3.14 SPI Receive Buffer Register (SPIBUF)

The SPI receive buffer register (SPIBUF) is shown in [Figure 23-31](#) and described in [Table 23-22](#).

**Figure 23-31. SPI Buffer Register (SPIBUF)**

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARERR	TIMEOUT	DLENERR
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0
23	Reserved						16
R-0							
15	RXDATA						0
R-0							

LEGEND: R/W = Read/Write; R = Read only; C = Clear; S = Set; -n = value after reset

**Table 23-22. SPI Buffer Register (SPIBUF) Field Descriptions**

Bit	Field	Value	Description
31	RXEMPTY	0 1	<p>Receive data buffer empty. When host reads the RXDATA field or the entire SPIBUF register this automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, the RXEMPTY flag is cleared. This flag gets set to 1 under following conditions:</p> <ul style="list-style-type: none"> <li>Reading the RXDATA field of the SPIBUF register.</li> <li>Writing 1 to clear the RXINTFLG bit in the SPIFLG register.</li> </ul> <p>New data has been received and copied into the SPIBUF register.</p> <p>No data received since last reading of the SPIBUF register.</p> <p>Write-Clearing the SPIFLG.RXINTFLG bit before reading the SPIBUF register indicates the received data is being ignored. Conversely, SPIFLG.RXINTFLG can be cleared by reading the RXDATA field of the SPIBUF register or the entire SPIBUF register.</p>
30	RXOVR	0 1	<p>Receive data buffer overrun. When a data transfer is completed and the received data is copied into the RXBUF while it is already full, RXOVR is set. An overrun always occurs to the RXBUF, and SPIBUF contents never get overwritten until after it is read by the CPU/DMA.</p> <p>Reading SPIBUF register does not clear the RXOVR bit. If an overrun interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF.</p> <p><b>Note:</b> A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR and DLENERR occur, then RXOVR will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.</p> <p>No receive data overrun condition occurred since last time reading the data field.</p> <p>A receive data overrun condition occurred since last time reading the data field.</p>

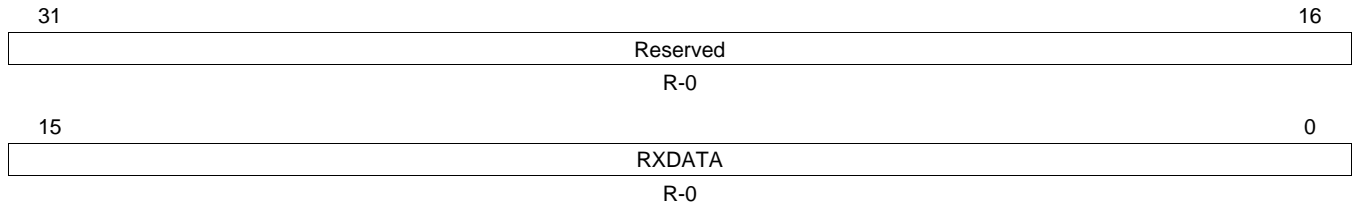
**Table 23-22. SPI Buffer Register (SPIBUF) Field Descriptions (continued)**

Bit	Field	Value	Description
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the data is copied to the shift register, the TXFULL flag will be cleared. Writing to the SPIDAT0/SPIDAT1 register when both TXBUF and the TX shift register are empty does not set the TXFULL flag.
		0	The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.
		1	The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data. The SPI samples the signal of the transmit pin (master: SIMO, slave: SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the flag BITERR is set. A possible reason for a bit error can be noise, a too-high bit rate/capacitive load, or another master/slave trying to transmit at the same time.
		0	No bit error occurred.
		1	A bit error occurred.
27	DESYNC		Desynchronization of slave device. This bit is active in master mode only. The master monitors the $\overline{\text{SPIx\_EN\bar{A}}}$ signal coming from the slave device and sets the DESYNC flag if $\overline{\text{SPIx\_EN\bar{A}}}$ is deactivated before the last reception point or after the last bit is transmitted plus $t_{\text{TZED\bar{E}LAY}}$ . If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
		0	No slave de-synchronization detected.
		1	A slave device is desynchronized.
26	PARERR		Parity error. The calculated parity differs from received parity bit. If the parity generator is enabled an even or odd parity bit is added at the end of a data word. During reception of the data word, the parity generator calculates the reference parity and compares it to the received parity bit. If a mismatch is detected, the PARERR flag is set.
		0	No parity error detected.
		1	A parity error occurred.
25	TIMEOUT		Time-out because of non-activation of $\overline{\text{SPIx\_EN\bar{A}}}$ pin. This bit is valid in master mode only. The SPI generates a time-out because the slave hasn't responded in time by activating the $\overline{\text{SPIx\_EN\bar{A}}}$ signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set.
		0	No $\overline{\text{SPIx\_EN\bar{A}}}$ pin time-out occurred.
		1	An $\overline{\text{SPIx\_EN\bar{A}}}$ signal time-out occurred.
24	DLENERR		Data length error flag.
		0	No data length error has occurred.
		1	A data length error has occurred.
23-16	Reserved	0	Reads return zero and writes have no effect.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received data, transferred from the receive shift-register at the end of a transfer completion. Irrespective of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

### 23.3.15 SPI Emulation Register (SPIEMU)

The SPI emulation register (SPIEMU) is shown in [Figure 23-32](#) and described in [Table 23-23](#).

**Figure 23-32. SPI Emulation Register (SPIEMU)**



LEGEND: R = Read only; -n = value after reset

**Table 23-23. SPI Emulation Register (SPIEMU) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zero and writes have no effect.
15-0	RXDATA	0-FFFFh	SPI receive data. SPI emulation is a mirror of the SPIBUF register. The only difference between SPIEMU and SPIBUF is that a read from SPIEMU does not clear any of the status flags.

### 23.3.16 SPI Delay Register (SPIDELAY)

The SPI delay register (SPIDELAY) is shown in [Figure 23-33](#) and described in [Table 23-24](#).

**Figure 23-33. SPI Delay Register (SPIDELAY)**

31	24	23	16
C2TDELAY		T2CDELAY	
R/W-0		R/W-0	
15	8	7	0
T2EDELAY		C2EDELAY	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; -n = value after reset

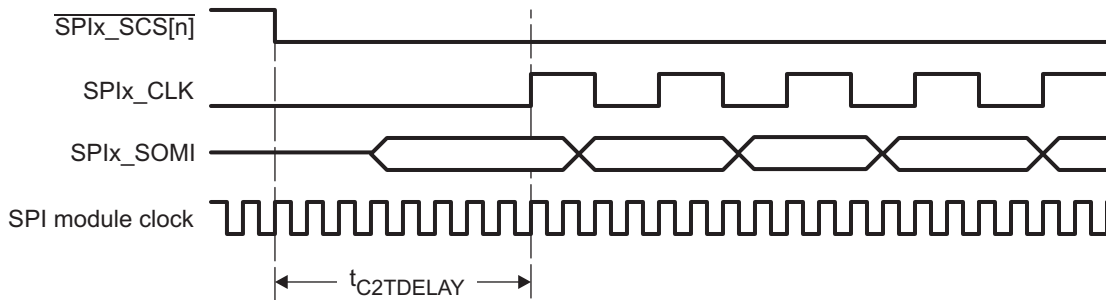
**Table 23-24. SPI Delay Register (SPIDELAY) Field Descriptions**

Bit	Field	Value	Description
31-24	C2TDELAY	0-FFh	<p>Chip-select-active-to-transmit-start-delay. C2TDELAY is used in master mode only. It defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SPI module clock cycles. C2TDELAY can be configured between 3 and 257 SPI module clock cycles. See <a href="#">Figure 23-34</a>.</p> <p>The setup time value is calculated as follows:  <math>t_{C2TDELAY} = (C2TDELAY + 2) \times \text{SPI module clock period}</math></p> <p><b>Note:</b> If C2TDELAY = 0, then <math>t_{C2TDELAY} = 0</math>.</p> <p>Example: SPI module clock = 25 MHz -&gt; SPI module clock period = 40 ns; C2TDELAY = 06h;                      &gt; <math>t_{C2TDELAY} = 320 \text{ ns}</math>;</p> <p>When the chip select signal becomes active, the slave has to prepare for data transfer within 320 ns.</p> <p><b>Note:</b> If phase = 1, the delay between <math>\overline{\text{SPIx\_SCS[n]}}</math> falling edge to the first edge of SPIx_CLK will have an additional 0.5 SPIx_CLK period delay. This delay is as per the SPI protocol.</p>
23-16	T2CDELAY	0-FFh	<p>Transmit-end-to-chip-select-inactive-delay. T2CDELAY is used in master mode only. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of SPI module clock cycles after the last bit is transferred. T2CDELAY can be configured between 2 and 256 SPI module clock cycles. See <a href="#">Figure 23-35</a>.</p> <p>The hold time value is calculated as follows:  <math>t_{T2CDELAY} = (T2CDELAY + 1) \times \text{SPI module clock period}</math></p> <p><b>Note:</b> If T2CDELAY = 0, then <math>t_{T2CDELAY} = 0</math></p> <p>Example: VBUSPCLK = 25 MHz -&gt; VBUSPCLK period = 40 ns; T2CDELAY = 03h;                      &gt; <math>t_{T2CDELAY} = 160 \text{ ns}</math>;</p> <p>After the last data bit (or parity bit) is being transferred the chip select signal is held active for 160 ns.</p> <p><b>Note:</b> If phase = 0, then between the last edge of SPIx_CLK and rise-edge of <math>\overline{\text{SPIx\_SCS[n]}}</math> there will be an additional delay of 0.5 SPIx_CLK period. This is as per the SPI protocol.</p> <p>Both C2TDELAY and T2CDELAY counters will not have any dependency on the <math>\overline{\text{SPIx\_ENA}}</math> pin value. Even if the <math>\overline{\text{SPIx\_ENA}}</math> pin is asserted by the slave, the master will continue to delay the start of SPIx_CLK until the C2TDELAY counter overflows.</p> <p>Similarly, even if the <math>\overline{\text{SPIx\_ENA}}</math> pin is deasserted by the slave, the master will continue to hold the <math>\overline{\text{SPIx\_SCS[n]}}</math> pins active until the T2CDELAY counter overflows. This way, it is assured that the setup/hold times of the <math>\overline{\text{SPIx\_SCS[n]}}</math> pins are determined by the delay timers alone. To achieve better throughput, it should be ensured that these two timers are kept at the minimum possible values.</p>

**Table 23-24. SPI Delay Register (SPIDELAY) Field Descriptions (continued)**

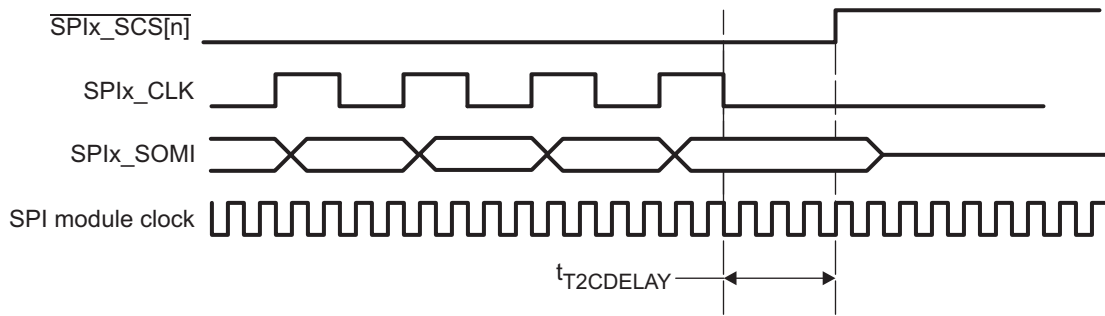
Bit	Field	Value	Description
15-8	T2EDELAY	0-FFh	<p>Transmit-data-finished-to-<math>\overline{\text{SPIx\_ENA}}</math>-pin-inactive-time-out. T2EDELAY is used in master mode only. It defines a time-out value as a multiple of SPI clock before the <math>\overline{\text{SPIx\_ENA}}</math> signal has to become inactive and after the CS becomes inactive. The SPI clock depends on which data format is selected. If the slave device is missing one or more clock edges, it is becoming desynchronized. Although the master has finished the data transfer the slave is still waiting for the missed clock pulses and the <math>\overline{\text{SPIx\_ENA}}</math> signal is not disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the <math>\overline{\text{SPIx\_ENA}}</math> signal is not deactivated in time. The DESYNC flag is set to indicate that the slave device did not deassert its <math>\overline{\text{SPIx\_ENA}}</math> pin in time to acknowledge that it has received all the bits of the sent character. The DESYNC flag is also set if the SPI detects a deassertion of the <math>\overline{\text{SPIx\_ENA}}</math> pin even before the end of the transmission. See <a href="#">Figure 23-36</a>.</p> <p>The time-out value is calculated as follows:  <math>t_{\text{T2EDELAY}} = \text{T2EDELAY}/\text{SPIClock}</math></p> <p>Example: SPIClock = 8 Mbit/s; T2EDELAY = 10h;  <math>&gt; t_{\text{T2EDELAY}} = 2 \mu\text{s}</math>;</p> <p>The slave device has to disable the <math>\overline{\text{SPIx\_ENA}}</math> signal within 2 <math>\mu\text{s}</math>; otherwise, the DESYNC flag in SPIFLG is set and an interrupt is asserted if enabled.</p>
7-0	C2EDELAY	0-FFh	<p>Chip-select-active-to-<math>\overline{\text{SPIx\_ENA}}</math>-signal-active-time-out. C2EDELAY is used only in master mode and it applies only if the addressed slave generates an <math>\overline{\text{SPIx\_ENA}}</math> signal as a hardware handshake response. C2EDELAY defines the maximum time between the SPI activates the chip select signal and the addressed slave has to respond by activating the <math>\overline{\text{SPIx\_ENA}}</math> signal. C2EDELAY defines a time-out value as a multiple of SPI clocks. See <a href="#">Figure 23-37</a>.</p> <p><b>Note:</b> If the slave device is not responding with the <math>\overline{\text{SPIx\_ENA}}</math> signal before the time-out value is reached, the TIMEOUT flag in SPIFLG is set and an interrupt is asserted if enabled.</p> <p>The timeout value is calculated as follows:  <math>t_{\text{C2EDELAY}} = \text{C2EDELAY}/\text{SPIClock}</math></p> <p>Example: SPIClock = 8 Mbit/s; C2EDELAY = 30h;  <math>&gt; t_{\text{C2EDELAY}} = 6 \mu\text{s}</math>;</p> <p>The slave device has to activate the <math>\overline{\text{SPIx\_ENA}}</math> signal within 6 <math>\mu\text{s}</math> after the SPI has activated the chip select signal (<math>\text{SPIx\_SCS}[n]</math>); otherwise, the TIMEOUT flag in SPIFLG is set and an interrupt is asserted if enabled.</p>

**Figure 23-34. Example:  $t_{\text{C2TDELAY}} = 8$  SPI Module Clock Cycles**

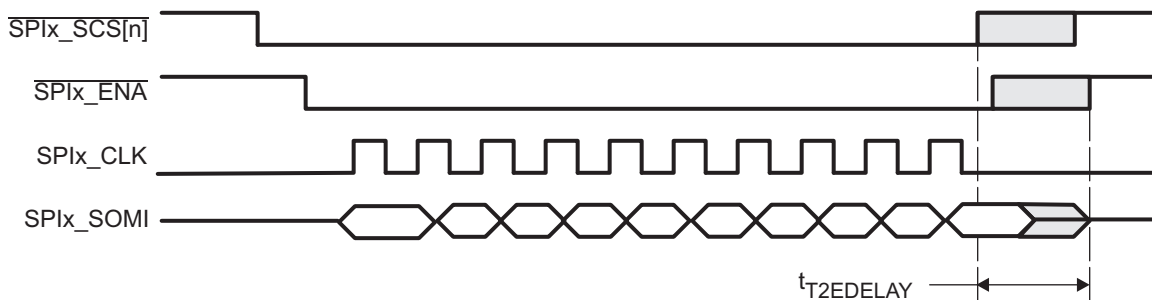




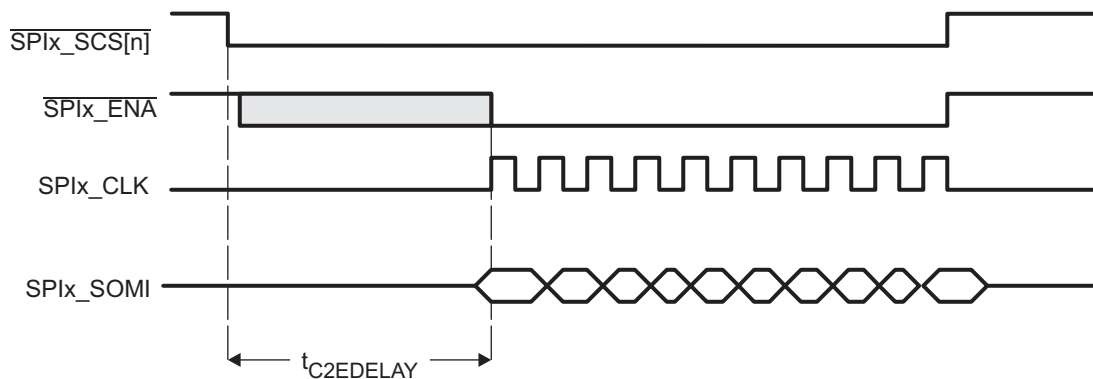
**Figure 23-35. Example:  $t_{T2CDELAY} = 4$  SPI Module Clock Cycles**



**Figure 23-36. Transmit-Data-Finished-to- $\overline{\text{SPIx\_ENA}}$ -Inactive-Timeout**



**Figure 23-37. Chip-Select-Active-to- $\overline{\text{SPIx\_ENA}}$ -Signal-Active-Timeout**



### 23.3.17 SPI Default Chip Select Register (SPIDEF)

The SPI default chip select register (SPIDEF) is shown in [Figure 23-38](#) and described in [Table 23-25](#).

**Figure 23-38. SPI Default Chip Select Register (SPIDEF)**

31	Reserved			16
	R-0			
15	8	7		0
	Reserved		CSDEF[n] <sup>(1)</sup>	
	R-0		R/W-FFh	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> Not all devices support multiple slave chip select ( $\overline{\text{SPIx\_SCS}}[n]$ ) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

**Table 23-25. SPI Default Chip Select Register (SPIDEF) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reads return zero and writes have no effect.
7-0	CSDEF[n]	0	Chip select default pattern. The CSDEF field defines the state of the the $\overline{\text{SPIx\_SCS}}[n]$ pins when no transmissions are performed. The value of the CSDEF field is driven directly on the $\overline{\text{SPIx\_SCS}}[n]$ pins. Each bit in the CSDEF field corresponds to an $\overline{\text{SPIx\_SCS}}[n]$ pin, for example, CSDEF[0] corresponds to $\overline{\text{SPIx\_SCS}}[0]$ (see your device-specific data manual to determine how many SPI pins are available on your device).  The state of the chip select pins during a transmission is specified through the CSNR field in the SPI transmit data register (SPIDAT1). The chip select pins can remain in their active state by setting the CSHOLD bit in SPIDAT1 to 1. In slave mode, the CSDEF field should be set to FFh.
		0	$\overline{\text{SPIx\_SCS}}[n]$ pin is driven low.
		1	$\overline{\text{SPIx\_SCS}}[n]$ pin is driven high.

### 23.3.18 SPI Data Format Registers (SPIFMT<sub>n</sub>)

The SPI data format registers (SPIFMT0, SPIFMT1, SPIFMT2, and SPIFMT3) are shown in [Figure 23-39](#) and described in [Table 23-26](#).

**Figure 23-39. SPI Data Format Register (SPIFMT<sub>n</sub>)**

31		30		29		24									
Reserved				WDELAY											
R-0				R/W-0											
23		22		21		20		19		18		17		16	
PARPOL		PARENA		WAITENA		SHIFTDIR		Reserved		DISCSTIMERS		POLARITY		PHASE	
R/W-0		R/W-0		R/W-0		R/W-0		R-0		R/W-0		R/W-0		R/W-0	
15		PRESCALE												8	
R/W-0															
7		5		4		0									
Reserved				CHARLEN											
R-0				R/W-0											

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-26. SPI Data Format Register (SPIFMT<sub>n</sub>) Field Descriptions**

Bit	Field	Value	Description
31-30	Reserved	0	Reads return zero and writes have no effect.
29-24	WDELAY	0-3Fh	Delay in between transmissions. Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. The delay to be applied is equal to:  $WDELAY \times P_{SPI\ module\ clock} + 2 \times P_{SPI\ module\ clock}$ P <sub>SPI module clock</sub> -> Period of SPI module clock
23	PARPOL	0 1	Parity polarity: even or odd. PARPOL can be modified in privilege mode only. 0 An even parity flag is added at the end of the transmit data stream. 1 An odd parity flag is added at the end of the transmit data stream.
22	PARENA	0 1	Parity enable. 0 No parity generation/ verification is performed. 1 A parity is transmitted at the end of each transmit data stream. At the end of a transfer the parity generator compares the received parity bit with the locally calculated parity flag. If the parity bits do not match the PARERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit.
21	WAITENA	0 1	The master waits for $\overline{SPIx\_ENA}$ signal from slave. WAITENA is considered in master mode only. In slave mode this bit has no meaning. WAITENA enables a flexible SPI network where slaves with $\overline{SPIx\_ENA}$ signal and slaves without $\overline{SPIx\_ENA}$ signal can be mixed. 0 The SPI does not wait for the $\overline{SPIx\_ENA}$ signal from the slave and directly starts the transfer. 1 Before the SPI starts the data transfer it waits for the $\overline{SPIx\_ENA}$ signal to become low. If the $\overline{SPIx\_ENA}$ signal is not pulled down by the addressed slave before the internal time-out counter (C2DELAY) overflows, then the master aborts the transfer and sets the TIMEOUT error flag.
20	SHIFTDIR	0 1	Shift direction. 0 Most significant bit is shifted out first. 1 Least significant bit is shifted out first.
19	Reserved	0	Reads return zero and writes have no effect.
18	DISCSTIMERS	0 1	Disable chip select timers for this format register. The C2TDELAY and T2CDELAY timers are by default enabled for all the data format registers. Using this bit, these timers can be disabled for a particular data format if not required. When a master is handling multiple slaves, with varied set-up hold requirement, the application can selectively choose to include or not include the chip select delay timers for any slaves. 0 Both C2TDELAY and T2CDELAY counts are inserted for the chip selects. 1 No C2TDELAY or T2CDELAY is inserted in the chip select timings.

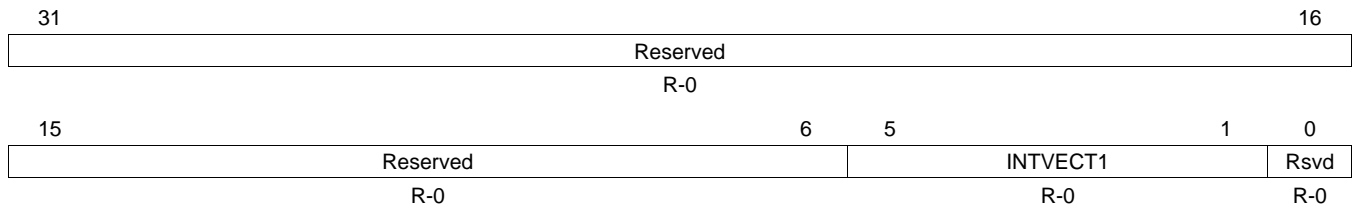
**Table 23-26. SPI Data Format Register (SPIFMT $n$ ) Field Descriptions (continued)**

Bit	Field	Value	Description
17	POLARITY	0 1	SPI clock polarity. 0 SPI clock signal is low-inactive (before and after data transfer the clock signal is low). 1 SPI clock signal is high-inactive (before and after data transfer the clock signal is high).
16	PHASE	0 1	SPI clock delay. 0 SPI clock signal is not delayed versus the transmit/receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge. 1 SPI clock signal is delayed by a half SPI clock cycle versus the transmit/receive data stream. The first transmit bit has to output prior to the first clock edge. The master and slave receive the first bit with the first edge.
15-8	PRESCALE	2h-FFh	SPI prescaler. It determines the bit transfer rate if the SPI is the network master and is directly derived from the SPI module clock. If the SPI is configured as slave, PRESCALE needs to be configured to a valid value, but PRESCALE is ignored.  The clock rate can be calculated as: SPI clock frequency = SPI module clock/(PRESCALE + 1) <b>Note:</b> PRESCALE values less than 2h are not supported.
7-5	Reserved	0	Reads return zero and writes have no effect.
4-0	CHARLEN	0-1Fh	SPI data word length. Legal values are 2h (data word length = 2 bit) to 10h (data word length = 16). Illegal values, such as 0 or 1Fh are not detected and their effect is indeterminate.

### 23.3.19 SPI Interrupt Vector Register 1 (INTVEC1)

The SPI interrupt vector register 1 (INTVEC1) is shown in [Figure 23-40](#) and described in [Table 23-27](#).

**Figure 23-40. SPI Interrupt Vector Register 1 (INTVEC1)**



LEGEND: R = Read only; -n = value after reset

**Table 23-27. SPI Interrupt Vector Register 1 (INTVEC1) Field Descriptions**

Bit	Field	Value	Description														
31-6	Reserved	0	Reads return zero and writes have no effect.														
5-1	INTVECT1	0-1Fh	Interrupt vector for interrupt line INT1. INTVECT1 returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest priority interrupt source first. The interrupts available for SPI in the descending order of their priorities are as given below. <ul style="list-style-type: none"> <li>• Transmission error Interrupt</li> <li>• Receive buffer overrun interrupt</li> <li>• Receive buffer full interrupt</li> <li>• Transmit buffer empty interrupt</li> </ul> The INTVECT1 field just reflects the status of SPIFLG in a vectorized format. So, any updates to SPIFLG will automatically reflect in the vector value in this register. <p>Vectors for each of these interrupts will be reflected on the INTVECT1 bits, when they occur. Reading the vectors for the receive buffer overrun and receive buffer full interrupts will automatically clear the respective flags in the SPIFLG. Reading the vector register when transmitter empty is indicated does not clear the TXINTFLG in SPIFLG. Writing a new data to SPIDAT0/SPIDAT1 clears the transmitter empty interrupt. On reading the INTVECT1 bits, the vector of the next highest priority interrupt (if any) will be then reflected on the INTVECT1 bits. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt will be reflected on the INTVECT1 bits.</p> The following are the SPI interrupt vectors for line INT1: <table style="width: 100%; border: none;"> <tr> <td style="padding-left: 20px;">0</td> <td>No interrupt pending</td> </tr> <tr> <td style="padding-left: 20px;">1h-10h</td> <td>Reserved</td> </tr> <tr> <td style="padding-left: 20px;">11h</td> <td>Error interrupt pending. Refer to lower halfword of SPIINT0 to determine more details about the type of error.</td> </tr> <tr> <td style="padding-left: 20px;">12h</td> <td>The pending interrupt is receive buffer full interrupt.</td> </tr> <tr> <td style="padding-left: 20px;">13h</td> <td>The pending interrupt is receive buffer overrun interrupt.</td> </tr> <tr> <td style="padding-left: 20px;">14h</td> <td>The pending interrupt is transmit buffer empty interrupt.</td> </tr> <tr> <td style="padding-left: 20px;">15h-1Fh</td> <td>Reserved</td> </tr> </table>	0	No interrupt pending	1h-10h	Reserved	11h	Error interrupt pending. Refer to lower halfword of SPIINT0 to determine more details about the type of error.	12h	The pending interrupt is receive buffer full interrupt.	13h	The pending interrupt is receive buffer overrun interrupt.	14h	The pending interrupt is transmit buffer empty interrupt.	15h-1Fh	Reserved
0	No interrupt pending																
1h-10h	Reserved																
11h	Error interrupt pending. Refer to lower halfword of SPIINT0 to determine more details about the type of error.																
12h	The pending interrupt is receive buffer full interrupt.																
13h	The pending interrupt is receive buffer overrun interrupt.																
14h	The pending interrupt is transmit buffer empty interrupt.																
15h-1Fh	Reserved																
0	Reserved	0	Reads return zero and writes have no effect.														

## 64-Bit Timer Plus

---

---

---

This chapter describes the operation of the software-programmable 64-bit Timer Plus. See your device-specific data manual to determine how many Timer modules are available on your device.

Topic	Page
<b>24.1 Introduction</b> .....	<b>1044</b>
<b>24.2 Registers</b> .....	<b>1062</b>

## 24.1 Introduction

The 64-bit Timer Plus can be programmed in 64-bit mode, dual 32-bit unchained mode, or dual 32-bit chained mode. Some Timer Plus implementations have signal connections to internal device reset that can be used in watchdog timer mode. New features over previous timers include: external clock/event input, period reload, external event capture, and timer counter register read reset.

### 24.1.1 Purpose of the Peripheral

The timer can support four basic modes of operation: a 64-bit general-purpose (GP) timer, dual unchained 32-bit GP timers, dual chained 32-bit timers, or a watchdog timer. The GP timer modes can be used to generate periodic interrupts and DMA synchronization events. The watchdog timer mode is used to provide a recovery mechanism for the device in the event of a fault condition (such as a non-exiting code loop).

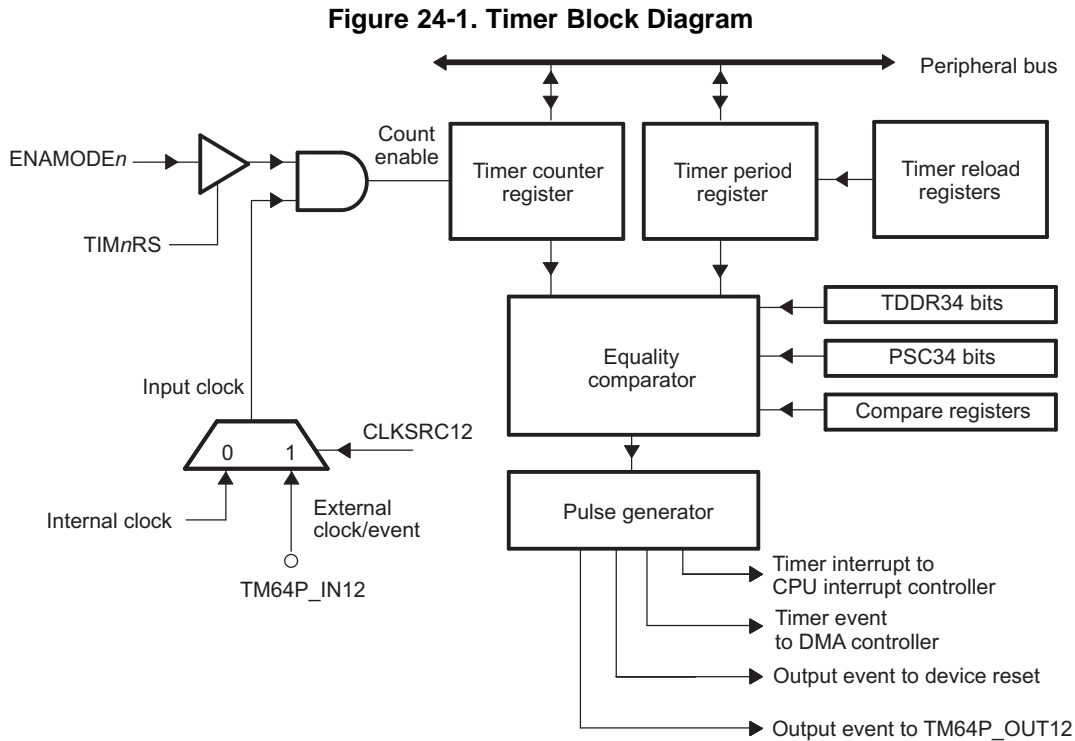
### 24.1.2 Features

The 64-bit timer consists of the following features -- some features may not be supported on all timer instantiations (see your device-specific data manual for supported features):

- 64-bit count-up counter
- Timer modes:
  - 64-bit general-purpose timer mode
  - Dual 32-bit unchained general-purpose timer mode
  - Dual 32-bit chained timer mode
  - Watchdog timer mode
- 2 possible clock sources:
  - Internal clock
  - External clock/event input via timer input pins
- 3 possible operation modes:
  - One-time operation (timer runs for one period then stops)
  - Continuous operation (timer automatically resets to zero after each period and continues to operate)
  - Continuous operation with period reload (timer automatically assumes the value of the reload registers after each period and continues to operate)
- Generates interrupts to CPU
- Generates sync events to DMA
- Generates output event to device reset (watchdog only)
- Generates output event to timer output pins (if pins are available)
- External event capture via timer input pins (if pins are available)

### 24.1.3 Block Diagram

A block diagram of the timer is shown in Figure 24-1. Detailed information about the architecture and operation of the timers is in Section 24.1.5 and Section 24.1.6.



### 24.1.4 Industry Standard Compatibility Statement

This peripheral is not intended to conform to any specific industry standard.

### Architecture

#### 24.1.5 Architecture – General-Purpose Timer Mode

This section describes the timer in the general-purpose (GP) timer mode.

##### 24.1.5.1 Backward Compatible Mode

The Timer Plus supports the following additional features over the other timers:

- External clock/event input
- Period reload
- External event capture mode
- Timer counter register read reset mode
- Timer counter capture registers
- Register for interrupt/DMA generation control and status

By default, period reload, external event capture mode, timer counter register read reset mode, timer counter capture registers, and interrupt/DMA/TM64P\_OUT generation control and status are not available. To enable these features, you must set the PLUSEN bit in the timer global control register (TGCR). These features are described throughout the following sections. External clock/event input is always available, regardless of the state of the backward compatible bit.



### 24.1.5.2 Clock Control

The timer can use an internal or external clock source for the counter period. The following sections explain how to select the clock source.

As shown in [Table 24-1](#) and [Figure 24-2](#), the timer clock source is selected using the clock source (CLKSRC12) bit in the timer control register (TCR). Two clock sources are available to drive the timer clock:

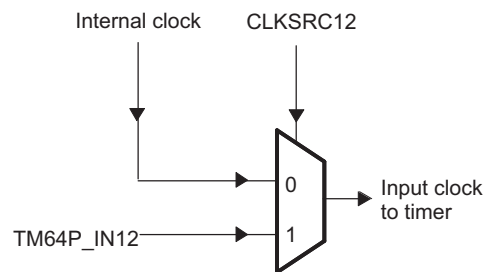
- internal clock by setting CLKSRC12 = 0.
- external clock on input pin TM64P\_IN12 by setting CLKSRC12 = 1.

At reset, the clock source is the internal clock. Details on each of the clock source configuration options are included in the following sections.

**Table 24-1. Timer Clock Source Selection**

CLKSRC12	Input Clock
0	Internal clock (default)
1	External clock on timer input

**Figure 24-2. Timer Clock Source Block Diagram**



#### 24.1.5.2.1 Using the Internal Clock Source to the Timer

The internal clock source to the timer is generated by the PLL controller. This clock source determines the speed of the timer since the timer counts up in units of source clock cycles. When determining the period and prescaler settings for the timer, choose the desired period in units of source clock cycles. For details on the generation of the on-chip clocks, see the *Phase-Locked Loop Controller (PLL)* chapter.

The CLKSRC12 parameter in the timer control register (TCR) determines whether an internal or external clock is used as the clock source for the timer. If the timer is configured in 64-bit mode or 32-bit chained mode, CLKSRC12 controls the clock source for the entire timer. If the timer is configured in dual 32-bit unchained mode (TIMMODE = 01 in TGCR), CLKSRC12 controls the timer 1:2 side of the timer only.

To select the internal clock as the clock source for the timer, CLKSRC12 in TCR must be cleared to 0.

#### 24.1.5.2.2 Using the External Clock Source to the Timer

An external clock source can be provided to clock the timer through the timer input pin TM64P\_IN12. The CLKSRC12 parameter in the timer control register (TCR) determines whether an internal or external clock is used as the clock source for the timer. If the timer is configured in 64-bit mode or 32-bit chained mode, CLKSRC12 controls the clock source for the entire timer. If the timer is configured in dual 32-bit unchained mode (TIMMODE = 01 in TGCR), CLKSRC12 controls the timer 1:2 side of the timer only.

At reset, the clock source defaults to the internal clock. Details on each of the clock source configuration options are included in the following sections. To select the external clock as the clock source for the timer, CLKSRC12 in TCR must be set to 1. The external clock source frequency must be no greater than the timer peripheral reference clock (see your device-specific data manual).

### 24.1.5.3 Signal Descriptions

As shown in Figure 24-2, the TM64P\_IN12 pin may be used as input to the timer. This signal can be used to drive the clock/event count or be used as an external event input for event capture mode. Pin TM64P\_OUT12 may be used as an output from the timer to generate a clock or pulse signal.

### 24.1.5.4 Timer Modes

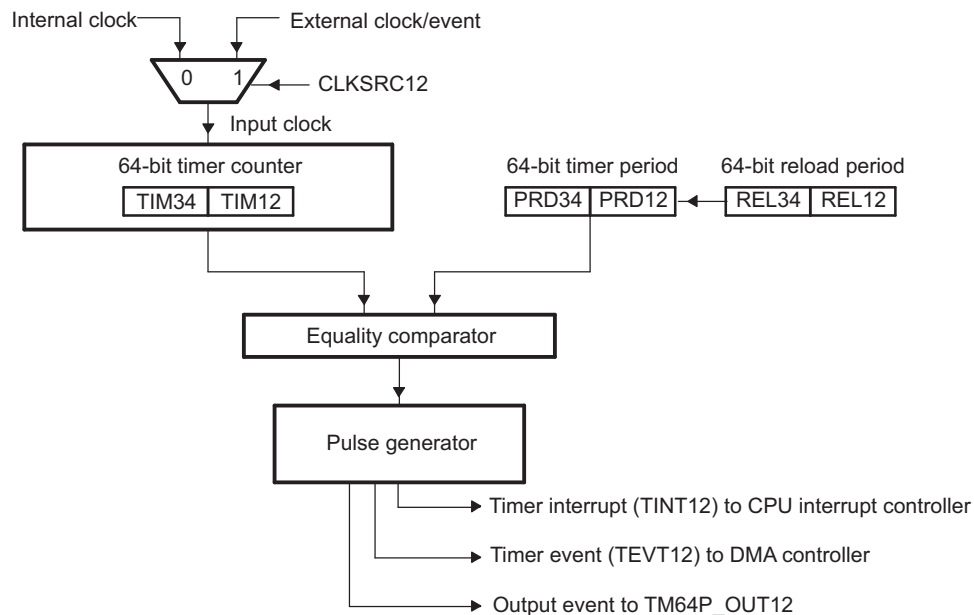
The following section describes the general-purpose (GP) timer modes.

#### 24.1.5.4.1 64-Bit Timer Mode

The timer can be configured as a 64-bit timer by clearing the TIMMODE bit in the timer global control register (TGCR) to 0. At reset, 0 is the default setting for the TIMMODE bit.

In this mode, the timer operates as a single 64-bit up-counter (Figure 24-3). The counter registers (TIM12 and TIM34) form a 64-bit timer counter register and the period registers (PRD12 and PRD34) form a 64-bit timer period register. When the timer is enabled, the timer counter starts incrementing by 1 at every timer input clock cycle. When the timer counter matches the timer period, a maskable timer interrupt (TINT12) and a timer EDMA (TEVT12) are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using control bits in TGCR.

Figure 24-3. 64-Bit Timer Mode Block Diagram



#### 24.1.5.4.1.1 Enabling the 64-Bit Timer

The TIM12RS and TIM34RS bits in TGCR control whether the timer is in reset or capable of operating. For the timer to operate in 64-bit timer mode, the TIM12RS and TIM34RS bits must be set to 1.

The ENAMODE12 bit in the timer control register (TCR) controls whether the timer is disabled, enabled to run once, enabled to run continuously, or enabled to run continuously with period reload; the ENAMODE34 bit has no effect in 64-bit timer mode. When the timer is disabled (ENAMODE12 = 0), the timer does not run and maintains its current count value. When the timer is enabled for one time operation (ENAMODE12 = 1), it counts up until the counter value equals the period value and then stops. When the timer is enabled for continuous operation (ENAMODE12 = 2h), the counter counts up until it reaches the period value, then resets itself to zero and begins counting again. When the timer is enabled for continuous operation with period reload (ENAMODE12 = 3h), the counter counts up until it reaches the period value, then resets itself to zero, reloads the period registers (PRD12 and PRD34) with the value in the period reload registers (REL12 and REL34), and begins counting again.

Table 24-2 shows the bit values in TGCR to configure the 64-bit timer.

**Table 24-2. 64-Bit Timer Configurations**

64-Bit Timer Configuration	TGCR Bit		TCR Bit
	TIM12RS	TIM34RS	ENAMODE12
To place the 64-bit timer in reset	0	0	0
To disable the 64-bit timer (out of reset)	1h	1h	0
To enable the 64-bit timer for one-time operation	1h	1h	1h
To enable the 64-bit timer for continuous operation	1h	1h	2h
To enable the 64-bit timer for continuous operation with period reload	1h	1h	3h

Once the timer stops, if an external clock is used as the timer clock, the timer must remain disabled for at least one external clock period or the timer will not start counting again. When using the external clock, the count value is synchronized to the internal clock.

Note that when both the timer counter and timer period are cleared to 0, the timer can be enabled but the timer counter does not increment because the timer period is 0.

#### 24.1.5.4.1.2 Reading the Counter Registers

When reading the timer count in 64-bit timer mode, the CPU must first read TIM12 followed by TIM34. When TIM12 is read, the timer copies TIM34 into a shadow register. When reading TIM34, the hardware logic returns the shadow register value. This ensures that the values read from the registers are not affected by the fact that the timer may continue to run as the registers are read. When reading the timers in 32-bit mode, TIM12 and TIM34 may be read in any order.

#### 24.1.5.4.1.3 64-Bit Timer Configuration Procedure

To configure the GP timer to operate as a 64-bit timer, follow the steps below:

1. Select 64-bit mode (TIMMODE in TGCR).
2. Remove the timer from reset (TIM12RS and TIM34RS in TGCR).
3. Select the desired timer period (PRD12 and PRD34). Program with the desired timer period value - 1.
4. Enable the timer (ENAMODE12 in TCR).
5. If ENAMODE12 = 3h, write the desired timer period for the next timer cycle in the period reload registers (REL12 and REL34). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

#### 24.1.5.4.2 Dual 32-Bit Timer Modes

Each of the general-purpose timers can be configured as dual 32-bit timers by configuring the TIMMODE bit in the timer global control register (TGCR). In dual 32-bit timer mode, the two 32-bit timers can be operated independently (unchained mode) or in conjunction with each other (chained mode).

##### 24.1.5.4.2.1 Chained Mode

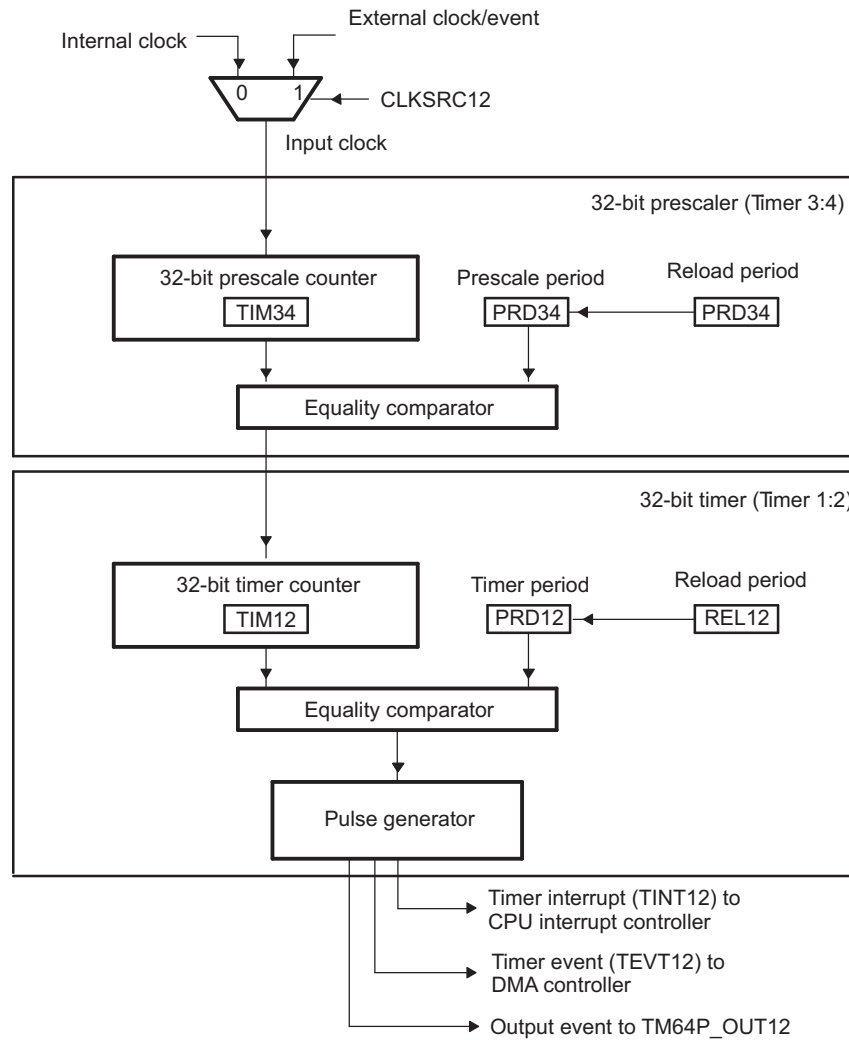
The general-purpose timers can each be configured as a dual 32-bit chained timer by setting the TIMMODE bit to 3h in TGCR.

In the chained mode ([Figure 24-4](#)), one 32-bit timer (timer 3:4) is used as a 32-bit prescaler and the other 32-bit timer (timer 1:2) is used as a 32-bit timer. The 32-bit prescaler is used to clock the 32-bit timer. The 32-bit prescaler uses one counter register (TIM34) to form a 32-bit prescale counter register and one period register (PRD34) to form a 32-bit prescale period register.

When the timer is enabled, the prescale counter starts incrementing by 1 at every timer input clock cycle. One cycle after the prescale counter matches the prescale period, a clock signal is generated and the prescale counter register is reset to 0 (see the example in [Figure 24-5](#)).

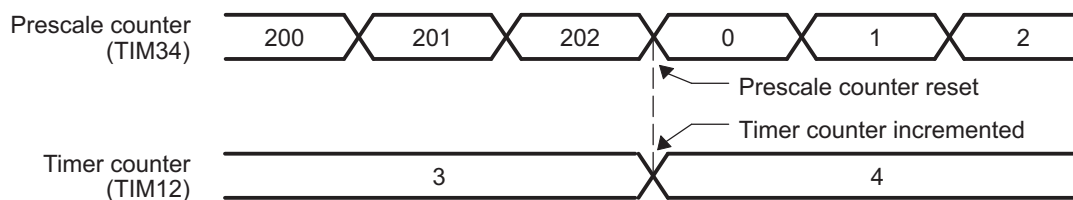
The other 32-bit timer (timer 1:2) uses one counter register (TIM12) to form a 32-bit timer counter register and one period register (PRD12) to form a 32-bit timer period register. This timer is clocked by the output clock from the prescaler. The timer counter increments by 1 at every prescaler output clock cycle. When the timer counter matches the timer period, a maskable timer interrupt (TINT12) and a timer EDMA event (TEVT12) are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using the TIM12RS and TIM34RS bits in TGCR. In the chained mode, the upper 16-bits of the timer control register (TCR) are not used.

**Figure 24-4. Dual 32-Bit Timers Chained Mode Block Diagram**



**Figure 24-5. Dual 32-Bit Timers Chained Mode Example**

32-bit prescaler settings: count = TIM34 = 200; period = PRD34 = 202  
 32-bit timer settings: count = TIM12 = 3; period = PRD12 = 4



#### 24.1.5.4.2.1.1 Enabling the 32-Bit Timer Chained Mode

The TIM12RS and TIM34RS bits in TGCR determine whether the timer is in reset, or if it is capable of operating. The TIM12RS bit controls the reset of the timer 1:2 side of the timer and the TIM34RS bits control the reset of the timer 3:4 side of the timer. For the timer to operate, the TIM12RS and TIM34RS bits must be set to 1.

The ENAMODE12 bit in the timer control register (TCR) controls whether the timer is disabled, enabled to run once, enabled to run continuously, or enabled to run continuously with period reload; the ENAMODE34 bit has no effect in 32-bit timer chained mode. When the timer is disabled (ENAMODE12 = 0), the timer does not run and maintains its current count value. When the timer is enabled for one time operation (ENAMODE12 = 1), it counts up until the counter value equals the period value and then stops. When the timer is enabled for continuous operation (ENAMODE12 = 2h), the counter counts up until it reaches the period value, then resets itself to zero and begins counting again. When the timer is enabled for continuous operation with period reload (ENAMODE12 = 3h), the counter counts up until it reaches the period value, then resets itself to zero, reloads the period registers (PRD12 and PRD34) with the value in the period reload registers (REL12 and REL34), and begins counting again.

Table 24-3 shows the bit values in TGCR to configure the 32-bit timer in chained mode.

**Table 24-3. 32-Bit Timer Chained Mode Configurations**

32-Bit Timer Configuration	TGCR Bit		TCR Bit
	TIM12RS	TIM34RS	ENAMODE12
To place the 32-bit timer chained mode in reset	0	0	0
To disable the 32-bit timer chained mode (out of reset)	1h	1h	0
To enable the 32-bit timer chained mode for one-time operation	1h	1h	1h
To enable the 32-bit timer chained mode for continuous operation	1h	1h	2h
To enable the 32-bit timer chained mode for continuous operation with period reload (Timer 3 only)	1h	1h	3h

Once the timer stops, if an external clock is used as the timer clock, the timer must remain disabled for at least one external clock period or the timer will not start counting again. When using the external clock, the count value is synchronized to the internal clock.

Note that when both the timer counter and timer period are cleared to 0, the timer can be enabled but the timer counter does not increment because the timer period is 0.

#### 24.1.5.4.2.1.2 32-Bit Timer Chained Mode Configuration Procedure

To configure the GP timer to operate as a dual 32-bit chained mode timer, follow the steps below:

1. Select 32-bit chained mode (TIMMODE in TGCR).
2. Remove the timer from reset (TIM12RS and TIM34RS in TGCR).
3. Select the desired timer period (PRD12). Program with the desired timer period value - 1.
4. Select the desired timer prescaler value (PRD34).
5. Enable the timer (ENAMODE12 in TCR).
6. If ENAMODE12 = 3h, write the desired timer period for the next timer cycle in the period reload registers (REL12 and REL34). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

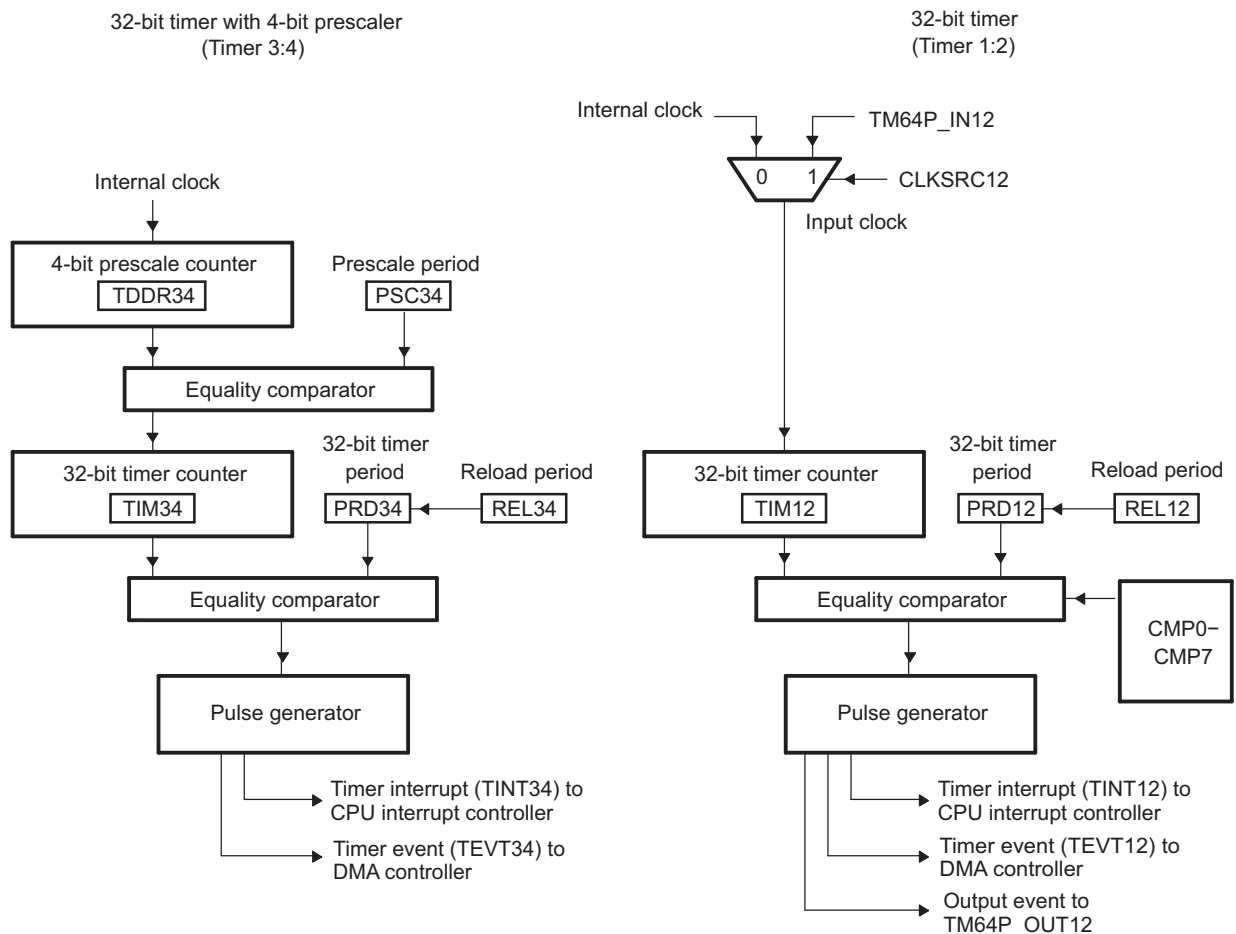
### 24.1.5.4.2.2 Unchained Mode

The general-purpose timers can be configured as a dual 32-bit unchained timers by setting the TIMMODE bit to 1 in TGCR.

In the unchained mode (Figure 24-6), the timer operates as two independent 32-bit timers. One 32-bit timer (timer 3:4) operates as a 32-bit timer being clocked by a 4-bit prescaler. The other 32-bit timer (timer 1:2) operates as a 32-bit timer with no prescaler.

Independent of the normal timer behavior, eight compare registers (CMP $n$ ) are compared against the value of the TIM12 register when the PLUSEN bit in TGCR is set. Upon a successful non-zero match, an interrupt and a DMA event are generated without affecting the TIM12 value, behavior, or associated counter registers. Note that some timer instantiations may not map the CMP interrupt and DMA events to the CPU and DMA engines (see your device-specific data manual for information).

**Figure 24-6. Dual 32-Bit Timers Unchained Mode Block Diagram**



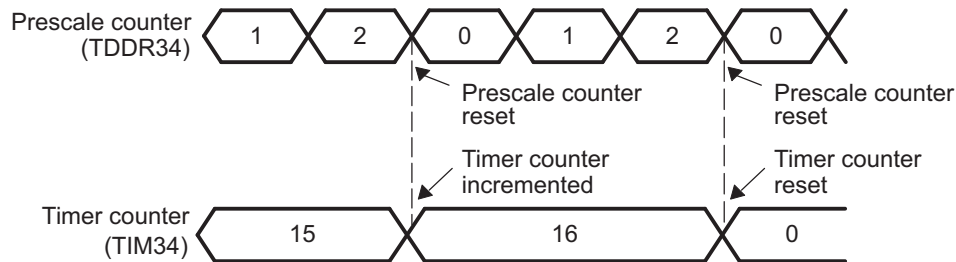
**24.1.5.4.2.2.1 32-Bit Timer With a 4-Bit Prescaler**

In the unchained mode, the 4-bit prescale is clocked by the internal clock. The 4-bit prescaler uses the timer divide-down ratio (TDDR34) bit in TGCR to form a 4-bit prescale counter register and the prescale counter bits (PSC34) to form a 4-bit prescale period register (see Figure 24-6). When the timer is enabled, the prescale counter starts incrementing by 1 at every timer input clock cycle. One cycle after the prescale counter matches the prescale period, a clock signal is generated for the 32-bit timer.

The 32-bit timer uses TIM34 as a 32-bit timer counter register and PRD34 as a 32-bit timer period register. The 32-bit timer is clocked by the output clock from the 4-bit prescaler (see the example in Figure 24-7). The timer counter increments by 1 at every prescaler output clock cycle. When the timer counter matches the period, a maskable timer interrupt (TINT34) and a timer DMA event (TEVT34) are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using the TIM34RS bit in TGCR. For timer 3:4, the lower 16 bits of the timer control register (TCR) have no control.

**Figure 24-7. Dual 32-Bit Timers Unchained Mode Example**

4-bit prescaler settings: count = TDDR34 = 1; period = PSC34 = 2  
 32-bit timer settings: count = TIM34 = 15; period = PRD34 = 16



**24.1.5.4.2.2.2 32-Bit Timer with No Prescaler**

The other 32-bit timer (timer 1:2) uses TIM12 as the 32-bit counter register and PRD12 as a 32-bit timer period register (see Figure 24-6). When the timer is enabled, the timer counter increments by 1 at every timer input clock cycle. When the timer counter matches the timer period, a maskable timer interrupt (TINT12), a timer DMA event (TEVT12), and a timer output event on TM64P\_OUT12 are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using the TIM12RS bit in TGCR. For timer 1:2, the upper 16 bit of the timer control register (TCR) have no control.



### 24.1.5.4.2.2.3 Enabling the 32-Bit Unchained Mode Timer

The TIM12RS and TIM34RS bits in TGCR determine whether the timer is in reset, or if it is capable of operating. The TIM12RS bit controls the reset of the timer 1:2 side of the timer and the TIM34RS bit controls the reset of the timer 3:4 side of the timer. For the timer to operate, the TIM12RS and/or TIM34RS bits must be set to 1.

The ENAMODE $n$  bit in the timer control register (TCR) controls whether the timer is disabled, enabled to run once, or enabled to run continuously.

- When the timer is disabled (ENAMODE $n$  = 0), the timer does not run and maintains its current count value.
- When the timer is enabled for one time operation (ENAMODE $n$  = 1), it counts up until the counter value equals the period value and then stops.
- When the timer is enabled for continuous operation (ENAMODE $n$  = 2h), the counter counts up until it reaches the period value, then resets itself to zero and begins counting again.
- When the timer is enabled for continuous operation with period reload (ENAMODE $n$  = 3h), the counter counts up until it reaches the period value, then resets itself to zero, reloads the period registers (PRD12 and/or PRD34) with the value in the period reload registers (REL12 and/or REL34), and begins counting again.

Table 24-4 shows the bit values in TGCR to configure the 32-bit timer in unchained mode.

Once the timer stops, if an external clock is used as the timer clock, the timer must remain disabled for at least one external clock period or the timer will not start counting again. When using the external clock, the count value is synchronized to the internal clock.

Note that when both the timer counter and timer period are cleared to 0, the timer can be enabled but the timer counter does not increment because the timer period is 0.

**Table 24-4. 32-Bit Timer Unchained Mode Configurations**

32-Bit Timer Configuration	TGCR Bit		TCR Bit	
	TIM12RS	TIM34RS	ENAMODE12	ENAMODE34
To place the 32-bit timer unchained mode with 4-bit prescaler in reset	x	0	x	0
To disable the 32-bit timer unchained mode with 4-bit prescaler (out of reset)	x	1h	x	0
To enable the 32-bit timer unchained mode with 4-bit prescaler for one-time operation	x	1h	x	1h
To enable the 32-bit timer unchained mode with 4-bit prescaler for continuous operation	x	1h	x	2h
To enable the 32-bit timer unchained mode with 4-bit prescaler for continuous operation with period reload	x	1h	x	3h
To place the 32-bit timer unchained mode with no prescaler in reset	0	x	0	x
To disable the 32-bit timer unchained mode with no prescaler (out of reset)	1h	x	0	x
To enable the 32-bit timer unchained mode with no prescaler for one-time operation	1h	x	1h	x
To enable the 32-bit timer unchained mode with no prescaler for continuous operation	1h	x	2h	x
To enable the 32-bit timer unchained mode with no prescaler for continuous operation with period reload	1h	x	3h	x

#### 24.1.5.4.2.2.4 32-Bit Timer Unchained Mode Configuration Procedure

To configure timer 1:2, follow the steps below:

1. Select 32-bit unchained mode (TIMMODE in TGCR).
2. Remove the timer 1:2 from reset (TIM12RS in TGCR).
3. Select the desired timer period for timer 1:2 (PRD12). Program with the desired timer period value - 1.
4. Select the desired clock source for timer 1:2 (CLKSRC12 in TCR).
5. Enable timer 1:2 (ENAMODE12 in TCR).
6. If ENAMODE12 = 3h, write the desired timer period for the next timer cycle in the period reload register (REL12). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

To configure timer 3:4, follow the steps below:

1. Select 32-bit unchained mode (TIMMODE in TGCR).
2. Remove the timer 3:4 from reset (TIM34RS in TGCR).
3. Select the desired timer period for timer 3:4 (PRD34). Program with the desired timer period value - 1.
4. Select the desired prescaler value for timer 3:4 (PSC34 in TGCR).
5. Enable timer 3:4 (ENAMODE34 in TCR).
6. If ENAMODE34 = 3h, write the desired timer period for the next timer cycle in the period reload register (REL34). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

#### 24.1.5.4.2.2.5 Event Capture Mode

When the PLUSEN bit in the timer global control register (TGCR) is set, Event Capture Mode is available for TIM12 when the timer is configured in 32-bit unchained mode. When Event Capture Mode is enabled, the timer cycle is restarted when an external input event occurs on pin TM64P\_IN12. In particular, when an external input event occurs, the timer stops counting, generates output events (TINT12, TEVT12, and TM64P\_OUT12), copies values from the timer counter register TIM12 to the timer capture register CAP12, reloads the timer period register PRD12 if in continuous mode with period reload (ENAMODE = 3h), and then restarts counting in continuous mode. Event Capture Mode is available only when the timer clock source is the internal timer (CLKSRC = 0) and the timer is in continuous mode (ENAMODE = 2h or 3h).

Capture mode is enabled using the Capture mode enable bit CAPMODE12 in the timer control register (TCR). The type of input event is selected by the capture event mode bit CAPEVTMODE12 in the timer control register (TCR). All of the following input event types are available:

- Rising edge of input signal
- Falling edge of input signal
- Rising or falling edge of input signal

#### 24.1.5.4.2.2.6 Timer Counter Register Read Reset Mode

Read Reset Mode is available when the PLUSEN bit in the timer global control register (TGCR) is set and the timer is configured in 32-bit unchained mode. When Read Reset Mode is enabled, the timer cycle will restart when the timer counter registers are read (TIM12 and/or TIM34). In particular, when the timer registers are read, the timer stops counting, copies values from the timer counter registers (TIM12 and/or TIM34) to the timer capture registers (CAP12 and/or CAP34), reloads the timer period registers (PRD12 and/or PRD34) if in continuous mode with period reload (ENAMODE = 3h), and then restarts counting in continuous mode. Timer output events (TINT $n$ , TEVT $n$ , and TM64P\_OUT $n$ ) are not generated during this process. Read Reset Mode is enabled using the read reset mode enable bit (READRSTMODE) in the timer control register (TCR).

### 24.1.5.4.3 Timer Capture Registers

When the timer has a timeout due to a normal expiration of timer, external input event in Event Capture Mode, or read of timer counter registers in Read Reset Mode, the values of the timer counter registers (TIM12 and TIM34) are copied onto the timer counter capture registers (CAP12 and CAP34). Note that the value in TDDR is not captured when a read of TIM34 happens.

### 24.1.5.4.4 Counter and Period Registers Used in GP Timer Modes

Table 24-5 summarizes how the counter registers (TIM $n$ ) and period registers (PRD $n$ ) are used in each GP timer mode.

**Table 24-5. Counter and Period Registers Used in GP Timer Modes**

Timer Mode	Counter Registers	Period Registers
64-bit general-purpose	TIM34:TIM12	PRD34:PRD12
Dual 32-bit chained:		
Prescaler (Timer 3:4)	TIM34	PRD34
Timer (Timer 1:2)	TIM12	PRD12
Dual 32-bit unchained:		
Timer (Timer 1:2)	TIM12	PRD12
Timer with prescaler (Timer 3:4)	TDDR34 bits and TIM34	PSC34 bits and PRD34

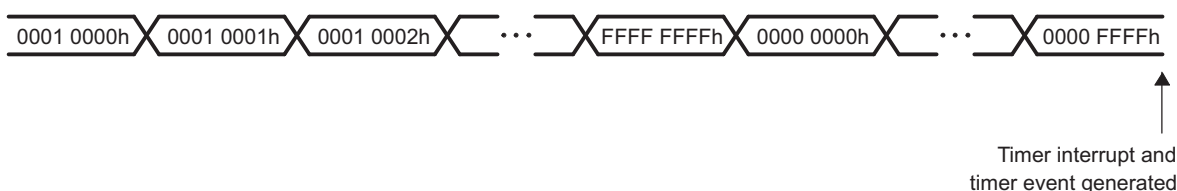
### 24.1.5.5 Timer Operation Boundary Conditions

The following boundary conditions affect the timer operation.

#### 24.1.5.5.1 Timer Counter Overflow

Timer counter overflow can happen when the timer counter register is set to a value greater than the value in the timer period register. The counter reaches its maximum value (FFFF FFFFh or FFFF FFFF FFFF FFFFh), rolls over to 0, and continues counting until it reaches the timer period. An example is in Figure 24-8.

**Figure 24-8. 32-Bit Timer Counter Overflow Example**



#### 24.1.5.5.2 Writing to Registers of an Active Timer

Writes to most timer registers are not allowed when the timer is active, except for setting the timer period reload registers (REL12 and REL34) and stopping and resetting the timers. In the 64-bit and dual 32-bit timer modes, registers that are protected by hardware are:

- TIM12
- TIM34
- PRD12
- PRD34
- TCR (except the ENAMODE bit)
- TGCR (except the TIM12RS and TIM34RS bits)

### 24.1.5.6 General-Purpose Timer Power Management

The timer can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter. The timer can be placed in an idle mode to conserve power when it is not being used.

### 24.1.6 Architecture – Watchdog Timer Mode

This section describes the use of the timer as a watchdog timer. In order to fully function in watchdog timer mode, the timer must be internally connected to the device hardware reset signal. For information on which timer instantiation can function as a watchdog timer, see your device-specific data manual.

#### 24.1.6.1 Watchdog Timer

As a 64-bit watchdog timer, the peripheral can be used to prevent system lockup when the software becomes trapped in loops with no controlled exit.

After a hardware reset, the watchdog timer is disabled. The timer then can be configured as a watchdog timer using the timer mode (TIMMODE) bit in the timer global control register (TGCR) and the watchdog timer enable (WDEN) bit in the watchdog timer control register (WDTCR). In the watchdog timer mode, the timer requires a special service sequence to be executed periodically. Without this periodic servicing, the timer counter increments until it matches the timer period and causes a watchdog timeout event.

When the timeout event occurs, the watchdog timer resets the entire processor.

#### 24.1.6.2 Watchdog Timer Mode Restrictions

The watchdog timer mode has the following restrictions:

- No external clock source
- No one-time enabling

#### 24.1.6.3 Watchdog Timer Mode Operation

The watchdog timer mode is selected and enabled when:

- TIMMODE = 2h in TGCR
- WDEN = 1 in WDTCR

Figure 24-9 shows the timer when it is used in the watchdog timer mode. The counter registers (TIM12 and TIM34) form a 64-bit timer counter register and the period registers (PRD12 and PRD34) form a 64-bit period register. When the timer counter matches the timer period, the timer generates a watchdog timeout event which resets the entire processor.

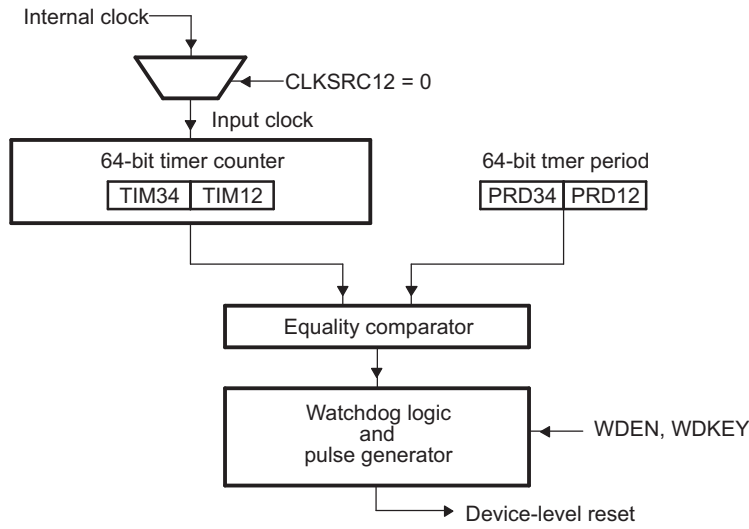
To activate the watchdog timer, a certain sequence of events must be followed, as shown in the state diagram of Figure 24-10.

Once the watchdog timer is activated, it can be disabled only by a watchdog timeout event or by a hardware reset. A special key sequence is required to prevent the watchdog timer from being accidentally serviced while the software is trapped in a loop or by some other software failure.

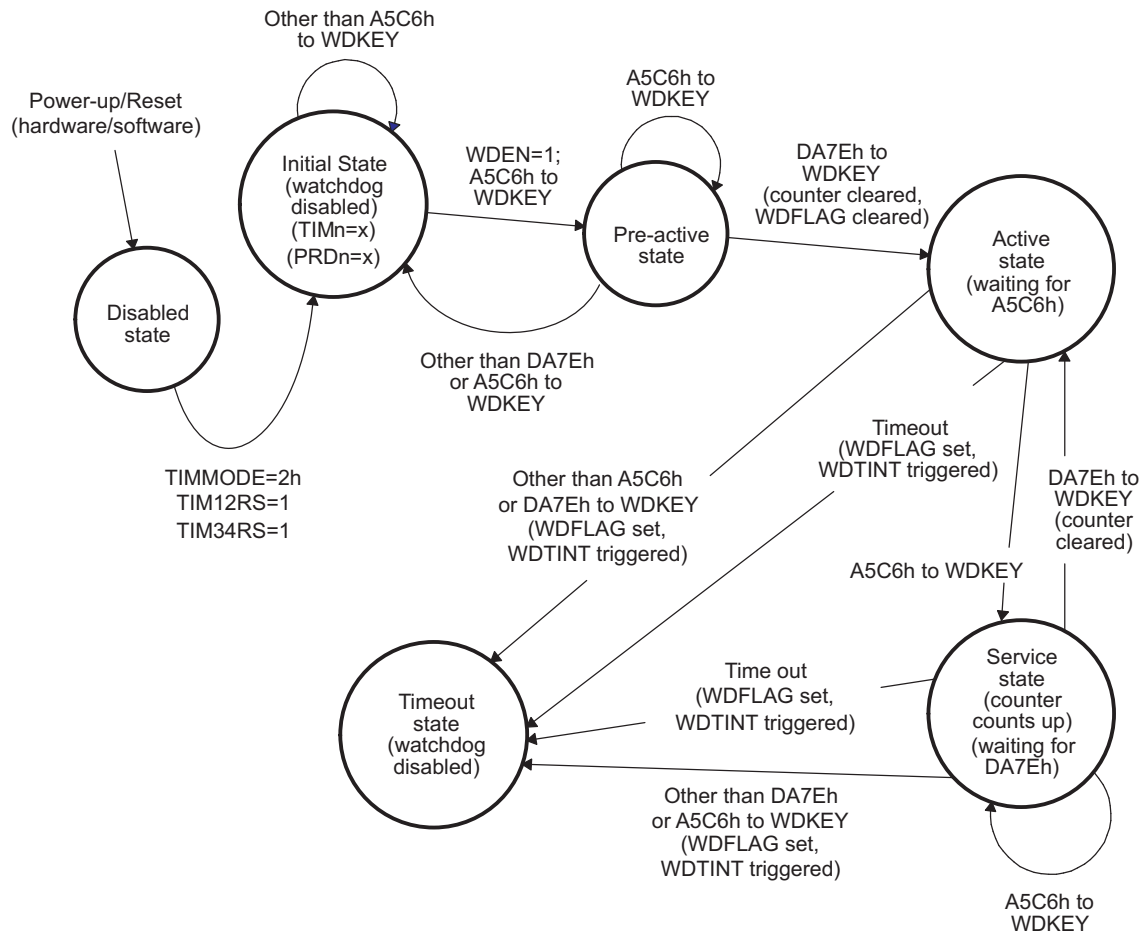
To prevent a watchdog timeout event, the timer has to be serviced periodically by writing A5C6h followed by DA7Eh to the watchdog timer service key (WDKEY) bits in WDTCR before the timer finishes counting up. Both A5C6h and DA7Eh are allowed to be written to the WDKEY bits, but only the correct sequence of A5C6h followed by DA7Eh to the WDKEY bits services the watchdog timer. Any other writes to the WDKEY bits triggers the watchdog timeout event immediately.

When the watchdog timer is in the Timeout state, the watchdog timer is disabled, the WDEN bit is cleared to 0, and the timer is reset.

**Figure 24-9. Watchdog Timer Mode Block Diagram**



**Figure 24-10. Watchdog Timer Operation State Diagram**



After a hardware reset, the watchdog timer is disabled; however, reads or writes to the watchdog timer registers are allowed. Once the WDEN bit is set (enabling the watchdog timer) and A5C6h is written to the WDKEY bits, the watchdog timer enters the Pre-active state. In the Pre-active state:

- A write to WDTCR is allowed only when the write comes with the correct key (A5C6h or DA7Eh) to the WDKEY bits.
- A write of DA7Eh to the WDKEY bits when the WDEN bit is set to 1 resets the counters and activates the watchdog timer.

The watchdog timer must be configured before the watchdog timer enters the Active state. The WDEN bit must be set to 1 before writing DA7Eh to the WDKEY bits in the Pre-active state. Every time the watchdog timer is serviced by the correct WDKEY sequence, the watchdog timer counter is automatically reset.

#### 24.1.6.4 Watchdog Timer Register Write Protection

Once the watchdog timer enters the Pre-active state (see [Figure 24-10](#)), writes to TIM12, TIM34, PRD12, PRD34, and WDTCR are write protected (except for the WDKEY field). While the watchdog timer is in the Timeout state, writing to the WDEN bit has no effect.

Once the watchdog timer enters its Initial state (see [Figure 24-10](#)), do not write to TGCR.

#### 24.1.6.5 Watchdog Timer Power Management

The watchdog timer cannot be placed in power-down mode.

#### 24.1.7 Reset Considerations

The timer has two reset sources: hardware reset and the timer reset (TIM12RS and TIM34RS) bits in the timer global control register (TGCR).

##### 24.1.7.1 Software Reset Considerations

When the TIM12RS bit in TGCR is cleared to 0, the TIM12 register is held with the current value.

When the TIM34RS bit in TGCR is cleared to 0, the TIM34 register is held with the current value.

##### 24.1.7.2 Hardware Reset Considerations

When a hardware reset is asserted, all timer registers are set to their default values.

#### 24.1.8 Interrupt Support

Each of the timers can send either one of two separate interrupt events (TINT $n$ ) to the CPU, depending on the operating mode of the timer. The timer interrupts are generated when the count value in the counter register reaches the value specified in the period register.

When the PLUSEN bit in the timer global control register (TGCR) is set, matches between TIM12 and CMP $n$  in dual 32-bit unchained mode will also generate interrupts. Setting the PLUSEN bit also enables additional features for control, status, and generation of interrupts. See [Section 24.1.11](#) for more information.

#### 24.1.9 DMA Event Support

Each of the timers can send either one of two separate timer events (TEVT $n$ ) to the DMA engine, depending on the operating mode of the timer. The timer events are generated when the count value in the counters register reaches the value specified in the period register.

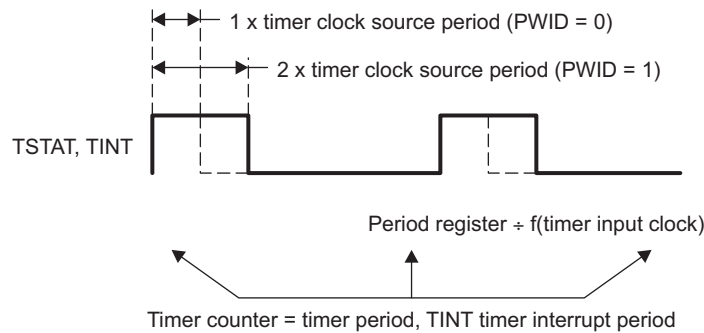
When the PLUSEN bit in the timer global control register (TGCR) is set, matches between TIM12 and CMP $n$  in dual 32-bit unchained mode will also generate DMA events. Setting the PLUSEN bit also enables additional features for control, status, and generation of dma events are enabled. See [Section 24.1.11](#) for more information.

### 24.1.10 TM64P\_OUT Event Support

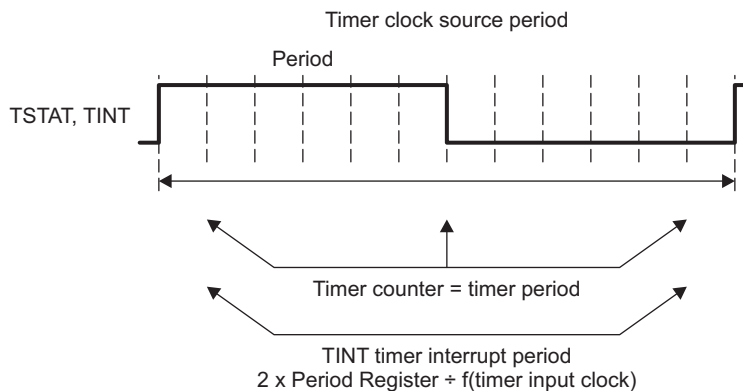
The timer can generate an output pulse (Figure 24-11) or clock (Figure 24-12) signals on the TM64P\_OUT12 pin. The output signal is generated when the count value in the counter registers reaches the value specified in the period registers (TSTAT12 drives the TM64P\_OUT12 pin). Table 24-6 gives equations for various TSTAT12 timing parameters in pulse and clock modes.

The output mode is selected with the clock/pulse bit (CP<sub>n</sub>) in the timer control register (TCR). In pulse mode, the PWID12 bit in TCR sets the pulse width between 1 to 4 timer clock periods. The TM64P\_OUT12 pin may be inverted using the INVOUTP12 bit in TCR.

**Figure 24-11. Timer Operation in Pulse Mode (CP<sub>n</sub> = 0)**



**Figure 24-12. Timer Operation in Clock Mode (CP<sub>n</sub> = 1)**



**Table 24-6. TSTAT Parameters in Pulse and Clock Modes**

Mode	Frequency	Period	Width High	Width Low
Pulse	$\frac{f(\text{clock source})}{\text{timer period register}}$	$\frac{\text{timer period register}}{f(\text{clock source})}$	$\frac{(\text{PWID} + 1)}{f(\text{clock source})}$	$\frac{\text{timer period register} - (\text{PWID} + 1)}{f(\text{clock source})}$
Clock	$\frac{f(\text{clock source})}{2 \times \text{timer period register}}$	$\frac{2 \times \text{timer period register}}{f(\text{clock source})}$	$\frac{\text{timer period register}}{f(\text{clock source})}$	$\frac{\text{timer period register}}{f(\text{clock source})}$



## External Timer Pin GPIO Mode

The external timer pins (TM64P\_IN12 and TM64P\_OUT12) can be individually configured to function as general-purpose input/output (GPIO) pins. In GPIO mode, the pins are able to detect and drive arbitrary data. The pins are also able to source external interrupt events. Some timer instantiations may not have external pins, see your device-specific data manual for pin information.

The GPIO interrupt and GPIO enable register (GPINTGPEN) enables the GPIO mode and associated interrupts. The GPIO data and GPIO direction register (GPDATGPDIR) determines if GPIO-enabled pins are used as input or output pins; and it is the means by which data is read-from or written-to the GPIO pins.

Normal timer counting modes cannot be used when the GPIO mode is enabled -- TIM12RS in the timer global control register (TGCR) cannot be brought out of reset when either GPEN012 or GPEN112 in GPINTGPEN is asserted.

### 24.1.11 Interrupt/DMA Event Generation Control and Status

When the PLUSEN bit in the timer global control register (TGCR) is set, the timer supports additional features for control and status of interrupt and DMA event generation. Interrupt/DMA events are generated when the count value in the timer counter registers reaches the value specified in the timer period registers and interrupt/DMA events are also generated when the Event Capture Mode is enabled and an external event occurs.

To generate events in the case when the value in the timer counter registers equals the value specified in the timer period registers, set the period compare interrupt enable bit (PRDINTEN $n$ ) in the interrupt control and status register (INTCTLSTAT). The event status for this case is reflected in the period compare interrupt status bit (PRDINTSTAT $n$ ), which is also in INTCTLSTAT. The PRDINTSTAT $n$  bit is cleared by writing a 1 to the bit.

Similarly, to generate events in Event Capture Mode, set the event interrupt enable bit (EVTINTEN $n$ ) in INTCTLSTAT. The event status for this case is reflected in the external interrupt status bit (EVTINTSTAT $n$ ) in INTCTLSTAT. The EVTINTSTAT $n$  bit is cleared by writing a 1 to the bit.

### 24.1.12 Power Management

The general-purpose timers can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

### 24.1.13 Emulation Considerations

Each timer has an emulation management register (EMUMGT). As shown in [Table 24-7](#), the FREE and SOFT bits of EMUMGT determine how the timer responds to an emulation suspend event. An emulation suspend event corresponds to any type of emulator access to the CPU, such as a hardware or software breakpoint or a probe point.

Note that during emulation, the timer count values will increment once every timer peripheral clock (not CPU clock). So when single-stepping through code, the timer values will not update on every CPU clock cycle.

The timer can respond to emulation events from the CPU based on the configuration of the Emulation Suspend Source Register (SUSPSRC) in the System Configuration Module. See the *System Configuration (SYSCFG) Module* chapter for information on SUSPSRC and how it is configured.



**Table 24-7. Timer Emulation Modes Selection**

FREE	SOFT	Emulation Mode
0	0	The timer stops immediately.
0	1	The timer stops when the timer counter value increments and reaches the value in the timer period register.
1	x	The timer runs free regardless of SOFT bit status.

## 24.2 Registers

Table 24-8 lists the memory-mapped registers for the 64-bit Timer Plus. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 24-8 should be considered as reserved locations and the register contents should not be modified.

**Table 24-8. Timer Registers**

Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	<a href="#">Section 24.2.1</a>
4h	EMUMGT	Emulation Management Register	<a href="#">Section 24.2.2</a>
8h	GPINTGPEN	GPIO Interrupt and GPIO Enable Register	<a href="#">Section 24.2.3</a>
Ch	GPDATGPDIR	GPIO Data and GPIO Direction Register	<a href="#">Section 24.2.4</a>
10h	TIM12	Timer Counter Register 12	<a href="#">Section 24.2.5</a>
14h	TIM34	Timer Counter Register 34	<a href="#">Section 24.2.5</a>
18h	PRD12	Timer Period Register 12	<a href="#">Section 24.2.6</a>
1Ch	PRD34	Timer Period Register 34	<a href="#">Section 24.2.6</a>
20h	TCR	Timer Control Register	<a href="#">Section 24.2.7</a>
24h	TGCR	Timer Global Control Register	<a href="#">Section 24.2.8</a>
28h	WDTCR	Watchdog Timer Control Register	<a href="#">Section 24.2.9</a>
34h	REL12	Timer Reload Register 12	<a href="#">Section 24.2.10</a>
38h	REL34	Timer Reload Register 34	<a href="#">Section 24.2.11</a>
3Ch	CAP12	Timer Capture Register 12	<a href="#">Section 24.2.12</a>
40h	CAP34	Timer Capture Register 34	<a href="#">Section 24.2.13</a>
44h	INTCTLSTAT	Timer Interrupt Control and Status Register	<a href="#">Section 24.2.14</a>
60h	CMP0	Compare Register 0	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>
64h	CMP1	Compare Register 1	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>
68h	CMP2	Compare Register 2	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>
6Ch	CMP3	Compare Register 3	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>
70h	CMP4	Compare Register 4	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>
74h	CMP5	Compare Register 5	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>
78h	CMP6	Compare Register 6	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>

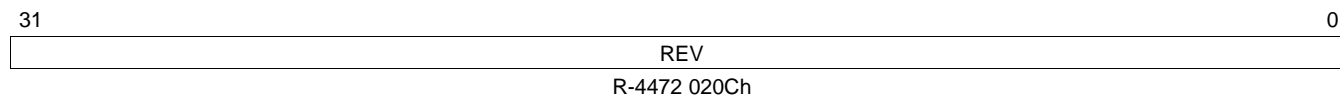
**Table 24-8. Timer Registers (continued)**

<b>Offset</b>	<b>Acronym</b>	<b>Register Description</b>	<b>Section</b>
7Ch	CMP7	Compare Register 7	<a href="#">Timer Compare Registers (CMP0-CMP7)</a>

### 24.2.1 Revision ID Register (REVID)

The revision ID register (REVID) contains the peripheral revision. The REVID is shown in [Figure 24-13](#) and described in [Table 24-9](#).

**Figure 24-13. Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

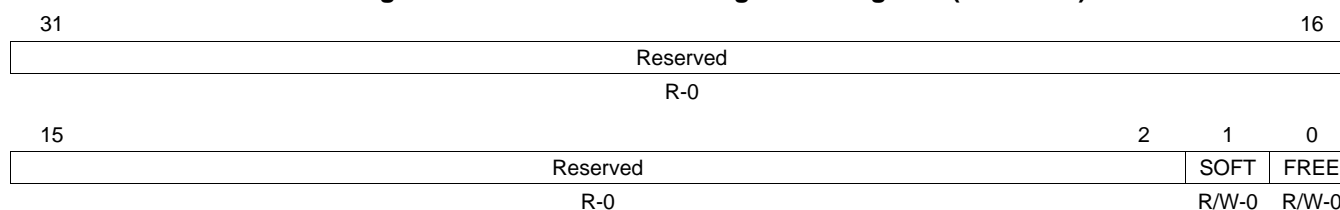
**Table 24-9. Revision ID Register (REVID) Field Descriptions**

Bit	Field	Value	Description
31-0	REV	4472 020Ch	Revision ID of the Timer.

### 24.2.2 Emulation Management Register (EMUMGT)

The emulation management register (EMUMGT) is shown in [Figure 24-14](#) and described in [Table 24-10](#).

**Figure 24-14. Emulation Management Register (EMUMGT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-10. Emulation Management Register (EMUMGT) Field Descriptions**

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT	0	Determines emulation mode functionality of the timer. When the FREE bit is cleared to 0, the SOFT bit selects the timer mode. The timer stops immediately.
		1	The timer stops when the counter increments and reaches the value in the timer period register (PRD <sub>n</sub> ).
0	FREE	0	Determines emulation mode functionality of the timer. When the FREE bit is cleared to 0, the SOFT bit selects the timer mode. The SOFT bit selects the timer mode.
		1	The timer runs free regardless of the SOFT bit.

### 24.2.3 GPIO Interrupt Control and Enable Register (GPINTGPEN)

The GPIO interrupt control and enable register (GPINTGPEN) is shown in [Figure 24-15](#) and described in [Table 24-11](#).

**Figure 24-15. GPIO Interrupt Control and Enable Register (GPINTGPEN)**

31	Reserved						24
R/W-0							
23	Reserved				18	17	16
R/W-0				R/W-0		R/W-0	
15	Reserved						8
R/W-0							
7	6	5	4	3	2	1	0
Reserved	GPINT12INVO		GPINT12INVI	Reserved		GPINT12ENO	GPINT12ENI
R-0	R/W-0		R/W-0	R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-11. GPIO Interrupt Control and Enable Register (GPINTGPEN) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	GPENO12	0	Enable TM64P_OUT12 to function in GPIO mode. TM64P_OUT12 is used as a TIMER output pin.
		1	TM64P_OUT12 is used as a GPIO pin.
16	GPENI12	0	Enable TM64P_IN12 to function in GPIO mode TM64P_IN12 is used as a TIMER input pin.
		1	TM64P_IN12 is used as a GPIO pin.
15-6	Reserved	0	Reserved
5	GPINT12INVO	0	Invert interrupt/event signal from TM64P_OUT12 when GPINT12ENO = 1. Rising signal edge on TM64P_OUT12 generates the interrupt/event.
		1	Falling signal edge on TM64P_OUT12 generates the interrupt/event.
4	GPINT12INVI	0	Invert interrupt/event signal for TM64P_IN12 when GPINT12ENI = 1. Rising signal edge on TM64P_IN12 generates the interrupt/event.
		1	Falling signal edge on TM64P_IN12 generates the interrupt/event.
3-2	Reserved	0	Reserved
1	GPINT12ENO	0	Enable TM64P_OUT12 to source interrupts/events in GPIO mode. Timer interrupts/events are sourced in TIMER mode.
		1	Timer interrupts/events are sourced externally from TM64P_OUT12.
0	GPINT12ENI	0	Enable TM64P_IN12 to source interrupts/events in GPIO mode. Timer interrupts/events are sourced in TIMER mode.
		1	Timer interrupts/events are sourced externally from TM64P_IN12.

### 24.2.4 GPIO Data and Direction Register (GPDATGPDIR)

The GPIO data and direction register (GPDATGPDIR) is shown in [Figure 24-16](#) and described in [Table 24-12](#).

**Figure 24-16. GPIO Data and Direction Register (GPDATGPDIR)**

31	Reserved	18	17	16
	R/W-0		GPDIRO12	GPDIRI12
			R/W-0	R/W-0
15	Reserved	2	1	0
	R/W-0		GPDATO12	GPDATI12
			R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-12. GPIO Data and Direction Register (GPDATGPDIR) Field Descriptions**

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	GPDIRO12	0 1	Select direction of TM64P_OUT12 in GPIO mode. 0 TM64P_OUT12 functions as an input pin in GPIO mode. 1 TM64P_OUT12 functions as an output pin in GPIO mode (TM64P_OUT12 cannot capture GPIO interrupt events when configured as output).
16	GPDIRI12	0 1	Select direction of TM64P_IN12 in GPIO mode. 0 TM64P_IN12 functions as an input pin in GPIO mode. 1 TM64P_IN12 functions as an output pin in GPIO mode (TM64P_IN12 cannot capture GPIO interrupt events when configured as output).
15-2	Reserved	0	Reserved
1	GPDATO12	0 1	Data on TM64P_OUT12 in GPIO mode. Only valid when GPENO12 = 1. <b>When GPDIRO12 = 0 (input):</b> 0 TM64P_OUT12 is detected logic low. 1 TM64P_OUT12 is detected logic high.
		0 1	<b>When GPDIRO12 = 1 (output):</b> 0 TM64P_OUT12 is driven logic low. 1 TM64P_OUT12 is driven logic high.
0	GPDATI12	0 1	Data on TM64P_IN12 in GPIO mode. Only valid when GPENI12 = 1. <b>When GPDIRI12 = 0 (input):</b> 0 TM64P_IN12 is detected logic low. 1 TM64P_IN12 is detected logic high.
		0 1	<b>When GPDIRI12 = 1 (output):</b> 0 TM64P_IN12 is driven logic low. 1 TM64P_IN12 is driven logic high.

## 24.2.5 Timer Counter Registers (TIM12 and TIM34)

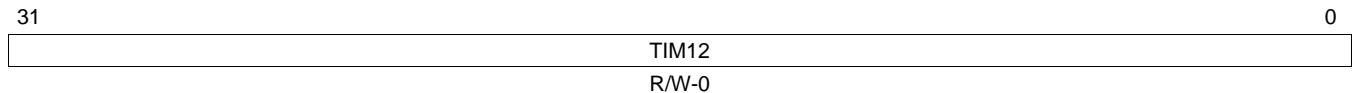
The timer counter register is a 64-bit wide register. This 64-bit register is divided into two 32-bit registers, TIM12 and TIM34.

In the dual 32-bit timer mode, the 64-bit register is divided with TIM12 acting as one 32-bit counter and TIM34 acting as another. These two registers can be configured as chained or unchained.

### 24.2.5.1 Timer Counter Register 12 (TIM12)

The timer counter register 12 (TIM12) is shown in [Figure 24-17](#) and described in [Table 24-13](#)

**Figure 24-17. Timer Counter Register 12 (TIM12)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

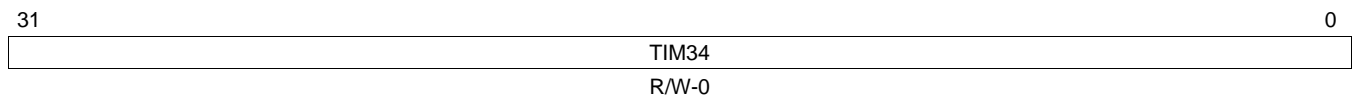
**Table 24-13. Timer Counter Register 12 (TIM12) Field Descriptions**

Bit	Field	Value	Description
31-0	TIM12	0-FFFF FFFFh	TIM12 count bits. This 32-bit value is the current count of the main counter.

### 24.2.5.2 Timer Counter Register 34 (TIM34)

The timer counter register 34 (TIM34) is shown in [Figure 24-18](#) and described in [Table 24-14](#).

**Figure 24-18. Timer Counter Register 34 (TIM34)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-14. Timer Counter Register 34 (TIM34) Field Descriptions**

Bit	Field	Value	Description
31-0	TIM34	0-FFFF FFFFh	TIM34 count bits. This 32-bit value is the current count of the main counter.

## 24.2.6 Timer Period Registers (PRD12 and PRD34)

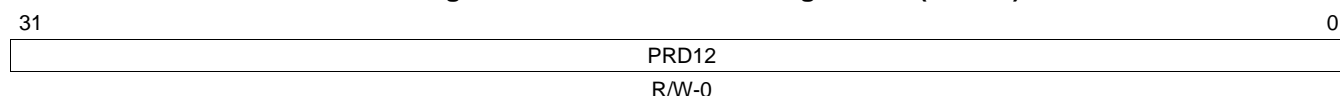
The timer period register is a 64-bit wide register. This 64-bit register is divided into two 32-bit registers, PRD12 and PRD34.

Similar to TIM $n$  in the dual 32-bit timer mode, PRD $n$  can be divided into 2 registers: for timer 1:2, PRD12 and for timer 3:4, PRD34. These two registers can be used in conjunction with the two timer counter registers TIM12 and TIM34.

### 24.2.6.1 Timer Period Register 12 (PRD12)

The timer period register 12 (PRD12) is shown in [Figure 24-19](#) and described in [Table 24-15](#).

**Figure 24-19. Timer Period Register 12 (PRD12)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

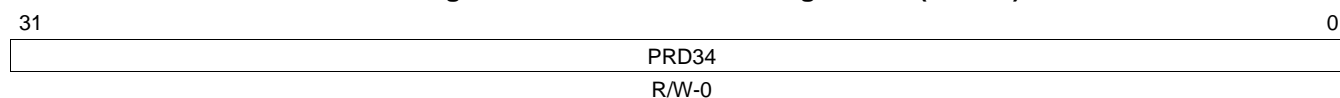
**Table 24-15. Timer Period Register (PRD12) Field Descriptions**

Bit	Field	Value	Description
31-0	PRD12	0-FFFF FFFFh	PRD12 period bits. This 32-bit value is the number of timer input clock cycles to count.

### 24.2.6.2 Timer Period Register 34 (PRD34)

The timer period register 34 (PRD34) is shown in [Figure 24-20](#) and described in [Table 24-16](#).

**Figure 24-20. Timer Period Register 34 (PRD34)**



LEGEND: R/W = Read/Write; R = Read only; - $n$  = value after reset

**Table 24-16. Timer Period Register (PRD34) Field Descriptions**

Bit	Field	Value	Description
31-0	PRD34	0-FFFF FFFFh	PRD34 period bits. This 32-bit value is the number of timer input clock cycles to count.

## 24.2.7 Timer Control Register (TCR)

The timer control register (TCR) is shown in [Figure 24-21](#) and described in [Table 24-17](#).

**Figure 24-21. Timer Control Register (TCR)**

31	Reserved				27	26	25	24	
R/W-0				READRSTMODE34		Reserved			
R/W-0				R/W-0		R/W-0			
23	22	21							16
ENAMODE34		Reserved							
R/W-0		R/W-0							
15	14	13	12	11	10	9	8		
Reserved		CAPVTMODE12		CAPMODE12	READRSTMODE12	TIEN12	CLKSRC12		
R-0		R/W-0		R/W-0	R/W-0	R/W-0	R/W-0		
7	6	5	4	3	2	1	0		
ENAMODE12		PWID12		CP12	INVINP12	INVOUTP12	TSTAT12		
R/W-0		R/W-0		R/W-0	R/W-0	R/W-0	R-0		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-17. Timer Control Register (TCR) Field Descriptions**

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26	READRSTMODE34	0 1	Read reset mode enable bit. Determines the effect of a timer counter read on TIM34. Read reset mode is only available in dual 32-bit unchained. Output events (interrupt/EDMA/other) are not generated when read reset occurs. 0 There is no effect when timer counter register TIM34 is read. 1 Timer counter is reset when timer counter register TIM34 is read.
25-24	Reserved	0	Reserved
23-22	ENAMODE34	0-3h 0 1h 2h 3h	Enabling mode: determines the enabling modes fo the timer. 0 The timer is disabled (not counting) and maintains current value. 1h The timer is enabled one time. The timer stops after the counter reaches the period. 2h The timer is enabled continuously, TIM34 increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching and continues. 3h The timer is enabled continuously with period reload, TIMn increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching, reloads the period register with the values in the reload registers (RELn), and continues counting.
21-14	Reserved	0	Reserved
13-12	CAPEVTMODE12	0-3h 0 1h 2h 3h	Capture event mode. Uses these bits to specify the type of event for Capture mode. 0 Event occurs on timer input rising edge. 1h Event occurs on time input falling edge. 2h Event occurs on both rising and falling edges. 3h Reserved
11	CAPMODE12	0 1	Capture mode enable bit. Determines if external event can reset timer. Capture mode is only available in dual 32-bit unchained mode and when CLKSRC = 0 and ENAMODE = 2h or 3h. Output events (interrupt/EDMA/other) are generated when capture mode event occurs. 0 Timer is not in capture mode. 1 Timer is in capture mode. External event can reset timer.
10	READRSTMODE12	0 1	Read reset mode enable bit. Determines the effect of a timer counter read on TIM12. Read reset mode is only available in dual 32-bit unchained. Output events (interrupt/EDMA/other) are not generated when read reset occurs. 0 There is no effect when timer counter register TIM12 is read. 1 Timer counter is reset when timer counter register TIM12 is read.



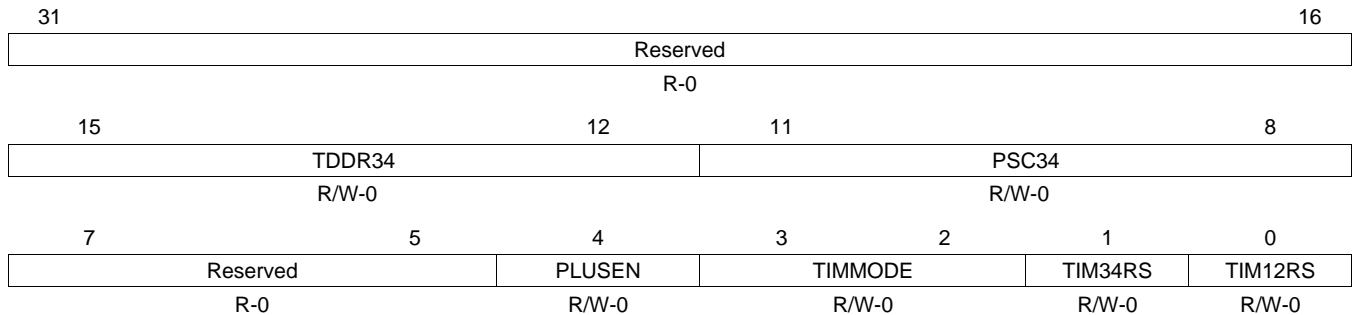
**Table 24-17. Timer Control Register (TCR) Field Descriptions (continued)**

Bit	Field	Value	Description
9	TIEN12	0 1	Timer input gate enable bit. Allows timer input pin TM64P_IN12 to gate the internal timer clock source (CLKSRC = 0). Timer starts counting when TM64P_IN12 transitions from low to high. Timer stops counting when TM64P_IN12 transitions from high to low. Timer clock is not gated by TM64P_IN12. Timer clock is gated by TM64P_IN12.
8	CLKSRC12	0 1	CLKSRC determines the selected clock source for the timer. Internal clock External clock on TM64P_IN12
7-6	ENAMODE12	0-3h 0 1h 2h 3h	Enabling mode: determines the enabling modes for the timer. The timer is disabled (not counting) and maintains current value. The timer is enabled one time. The timer stops after the counter reaches the period. The timer is enabled continuously, TIM <sub>n</sub> increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching and continues. The timer is enabled continuously with period reload, TIM <sub>n</sub> increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching, reloads the period register with the values in the reload registers (REL <sub>n</sub> ), and continues counting.
5-4	PWID12	0-3h 0 1h 2h 3h	Pulse width - Determines the pulse width on the TSTAT12 bit (and the TM64P_OUT12 pin) when the clock/pulse mode is set to pulse. TSTAT12 stays active for one timer clock cycle when the timer counter reaches the period. TSTAT12 stays active for two timer clock cycles when the timer counter reaches the period. TSTAT12 stays active for three timer clock cycles when the timer counter reaches the period. TSTAT12 stays active for four timer clock cycles when the timer counter reaches the period.
3	CP12	0 1	Clock/Pulse bit - Determines whether the TM64P_OUT12 output event should behave as a 50% duty-cycle clock or a signal pulse. Pulse Mode. TM64P_OUT12 goes active after the timer counter reaches the period. The pulse width is determined by PWID12. Clock Mode. TM64P_OUT12 will behave as a 50% duty cycle signal. It toggles high-to-low or low-to-high when the timer counter reaches zero.
2	INVINP12	0 1	Invert TM64P_IN12. Only affects operation if CLKSRC = 1. Uninverted TM64P_IN12 signal drives timer. Inverted TM64P_IN12 signal drives timer.
1	INVOUTP12	0 1	Invert TM64P_OUT12. TM64P_OUT12 signal is not inverted. TM64P_OUT12 signal is inverted.
0	TSTAT12	0 1	Timer status. Drives the value of timer output TM64P_OUT12 when it is configured to function as timer output. TM64P_OUT12 signal is not asserted. TM64P_OUT12 signal is asserted.

## 24.2.8 Timer Global Control Register (TGCR)

The timer global control register (TGCR) is shown in [Figure 24-22](#) and described in [Table 24-18](#).

**Figure 24-22. Timer Global Control Register (TGCR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

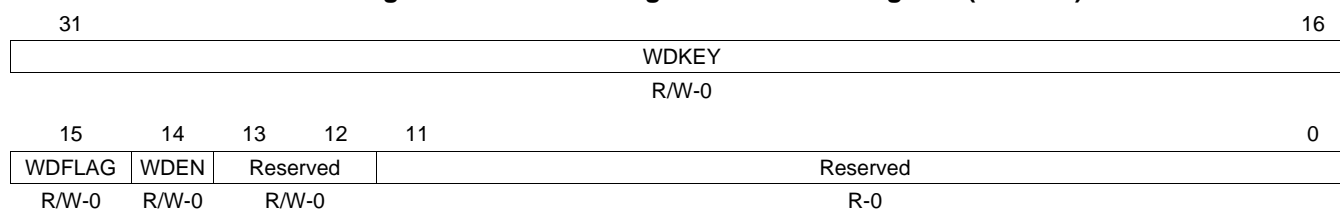
**Table 24-18. Timer Global Control Register (TGCR) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-12	TDDR34	0-Fh	Timer linear divide-down ratio specifies the timer divide-down ratio for timer 3:4. When the timer is enabled, TDDR34 increments every timer clock. The TIM34 counter increments on the cycle after TDDR34 matches PSC34. TDDR34 resets to 0 and continues. When TIM34 matches PRD34, timer 3:4 stops, if timer 3:4 is enabled one time; TIM34 resets to 0 on the cycle after matching PRD34 and timer 3:4 continues, if timer 3:4 is enabled continuously.
11-8	PSC34	0-Fh	TIM34 pre-scalar counter specifies the count for timer 3:4.
7-5	Reserved	0	Reserved
4	PLUSEN	0 1	Enable new timer plus features. 0 Enable backward compatibility. New timer features are unavailable. 1 Disable backward compatibility. New timer features are available.
3-2	TIMMODE	0-3h	TIMMODE determines the timer mode. 0 The timer is in 64-bit GP timer mode. 1h The timer is in dual 32-bit timer unchained mode. 2h The timer is in 64-bit watchdog timer mode. 3h The timer is in dual 32-bit timer, chained mode.
1	TIM34RS	0 1	Timer 3:4 reset. 0 Timer 3:4 is in reset. 1 Timer 3:4 is not in reset. Timer 3:4 can be used as a 32-bit timer. Note that for the timer to function properly in 64-bit timer mode, both TIM34RS and TIM12RS must be set to 1. Changing this bit does not affect the timer, if the timer is in the watchdog active state.
0	TIM12RS	0 1	Timer 1:2 reset. 0 Timer 1:2 is in reset. 1 Timer 1:2 is not in reset. Timer 1:2 can be used as a 32-bit timer. Note that for the timer to function properly in 64-bit timer mode, both TIM34RS and TIM12RS must be set to 1. Changing this bit does not affect the timer, if the timer is in the watchdog active state.

### 24.2.9 Watchdog Timer Control Register (WDTCR)

The watchdog timer control register (WDTCR) is shown in [Figure 24-23](#) and described in [Table 24-19](#).

**Figure 24-23. Watchdog Timer Control Register (WDTCR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

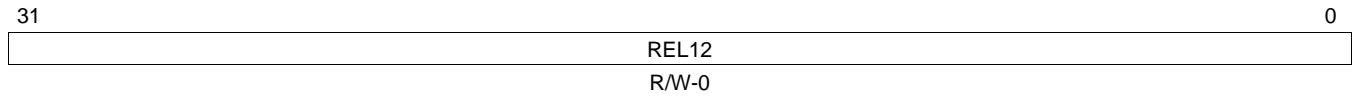
**Table 24-19. Watchdog Timer Control Register (WDTCR) Field Descriptions**

Bit	Field	Value	Description
31-16	WDKEY	0-FFFFh	16-bit watchdog timer service key. Only the sequence of an A5C6h followed by a DA7Eh services the watchdog. Not applicable in regular timer mode.
15	WDFLAG	0	No watchdog time-out occurred.
		1	Watchdog time-out occurred.
14	WDEN	0	Disable watchdog timer
		1	Enable watchdog timer
13-12	Reserved	0	Reserved. This bit field must be written as 00b.
11-0	Reserved	0	Reserved

### 24.2.10 Timer Reload Register 12 (REL12)

The timer reload register 12 (REL12) is shown in [Figure 24-24](#) and described in [Table 24-20](#).

**Figure 24-24. Timer Reload Register 12 (REL12)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

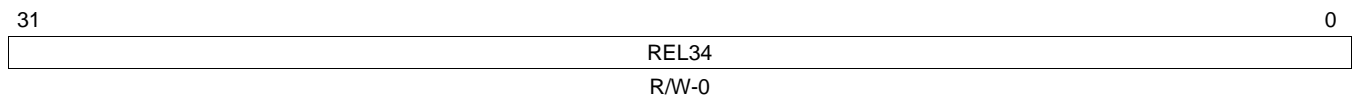
**Table 24-20. Timer Reload Register 12 (REL12) Field Descriptions**

Bit	Field	Value	Description
31-0	REL12	0-FFFF FFFFh	Period reload bits.

### 24.2.11 Timer Reload Register 34 (REL34)

The timer reload register 34 (REL34) is shown in [Figure 24-25](#) and described in [Table 24-21](#).

**Figure 24-25. Timer Reload Register 34 (REL34)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

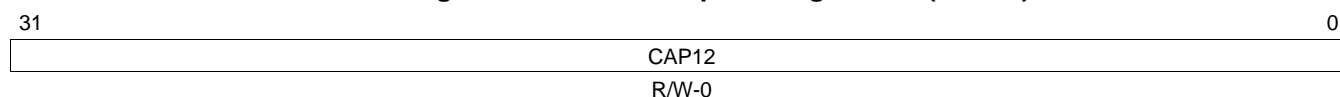
**Table 24-21. Timer Reload Register 34 (REL34) Field Descriptions**

Bit	Field	Value	Description
31-0	REL34	0-FFFF FFFFh	Period reload bits.

### 24.2.12 Timer Capture Register 12 (CAP12)

The timer capture register 12 (CAP12) is shown in [Figure 24-26](#) and described in [Table 24-22](#).

**Figure 24-26. Timer Capture Register 12 (CAP12)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

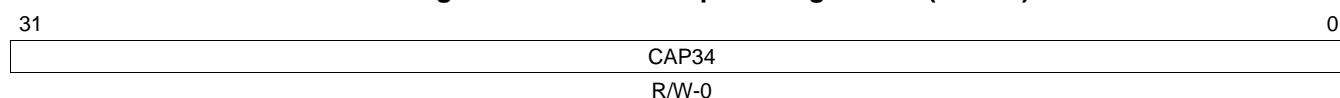
**Table 24-22. Timer Capture Register 12 (CAP12) Field Descriptions**

Bit	Field	Value	Description
31-0	CAP12	0-FFFF FFFFh	Captured timer counter bits.

### 24.2.13 Timer Capture Register 34 (CAP34)

The timer capture register 34 (CAP34) is shown in [Figure 24-27](#) and described in [Table 24-23](#).

**Figure 24-27. Timer Capture Register 34 (CAP34)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-23. Timer Capture Register 34 (CAP34) Field Descriptions**

Bit	Field	Value	Description
31-0	CAP34	0-FFFF FFFFh	Captured timer counter bits.

### 24.2.14 Timer Interrupt Control and Status Register (INTCTLSTAT)

The timer interrupt control and status register (INTCTLSTAT) is shown in [Figure 24-28](#) and described in [Table 24-24](#).

**Figure 24-28. Timer Interrupt Control and Status Register (INTCTLSTAT)**

31	Reserved				24
R-0					
23	20	19	18	17	16
Reserved		EVTINTSTAT34	EVTINTEN34	PRDINTSTAT34	PRDINTEN34
R-0		R/W1C-0	R/W-0	R/W1C-0	R/W-0
15	Reserved				8
R-0					
7	4	3	2	1	0
Reserved		EVTINTSTAT12	EVTINTEN12	PRDINTSTAT12	PRDINTEN12
R-0		R/W1C-0	R/W-0	R/W1C-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear bit; -n = value after reset

**Table 24-24. Timer Interrupt Control and Status Register (INTCTLSTAT) Field Descriptions**

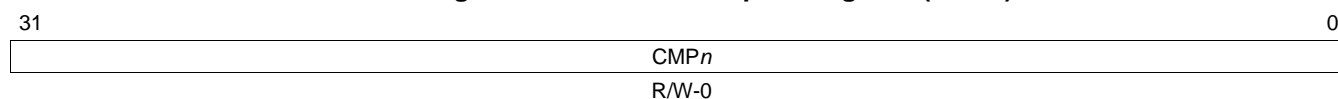
Bit	Field	Value	Description
31-20	Reserved	0	Reserved
19	EVTINTSTAT34	0	Interrupt status which reflects the condition that an external event caused a timeout when timer is in capture mode. Write a 1 to clear this bit.
		1	Interrupt has occurred.
18	EVTINTEN34	0	Enables the interrupt generation when timer is in capture mode.
		1	Disable interrupt when in event capture mode.
		1	Enable interrupt when in event capture mode.
17	PRDINTSTAT34	0	Interrupt status which reflects the condition that timer counter matched the period register when timer is enabled. Write a 1 to clear this bit.
		1	Interrupt has not occurred.
		1	Interrupt has occurred.
16	PRDINTEN34	0	Enable interrupt generation when timer is enabled in 64-bit/32-bit chained/unchained/watchdog modes.
		1	Disable interrupt
		1	Enable interrupt
15-4	Reserved	0	Reserved
3	EVTINTSTAT12	0	Interrupt status which reflects the condition that an external event caused a timeout when timer is in capture mode. Write a 1 to clear this bit.
		1	Interrupt has not occurred.
		1	Interrupt has occurred.
2	EVTINTEN12	0	Enables the interrupt generation when timer is in capture mode.
		1	Disable interrupt when in event capture mode.
		1	Enable interrupt when in event capture mode.
1	PRDINTSTAT12	0	Interrupt status which reflects the condition that timer counter matched the period register when timer is enabled. Write a 1 to clear this bit.
		1	Interrupt has not occurred.
		1	Interrupt has occurred.

**Table 24-24. Timer Interrupt Control and Status Register (INTCTLSTAT) Field Descriptions (continued)**

Bit	Field	Value	Description
0	PRDINTEN12		Enable interrupt generation when timer is enabled in 64-bit/32-bit chained/unchained/watchdog modes.
		0	Disable interrupt
		1	Enable interrupt

**Timer Compare Registers (CMP0-CMP7)**

The timer compare register (CMP $n$ ) is shown in [Figure 24-29](#) and described in [Table 24-25](#).

**Figure 24-29. Timer Compare Register (CMP $n$ )**


LEGEND: R/W = Read/Write; - $n$  = value after reset

**Table 24-25. Timer Compare Register (CMP $n$ ) Field Descriptions**

Bit	Field	Value	Description
31-0	CMP $n$	0-FFFF FFFFh	Timer compare register. When PLUSEN = 1 in the timer global control register (TGCR) and the timer is configured in 32-bit unchained mode, TIM12 is compared to all 8 compare registers (CMP0-CMP7). When CMP $n$ matches TIM12, a timer CMP $n$ interrupt and DMA event are generated. A CMP $n$ match will not affect the TIM12 count or behavior.

## ***Universal Asynchronous Receiver/Transmitter (UART)***

---

---

This chapter describes the universal asynchronous receiver/transmitter (UART) peripheral. See your device-specific data manual to determine how many UARTs are available on your device.

<b>Topic</b>	<b>Page</b>
<b>25.1 Introduction .....</b>	<b>1078</b>
<b>25.2 Peripheral Architecture.....</b>	<b>1080</b>
<b>25.3 Registers .....</b>	<b>1091</b>



## 25.1 Introduction

### 25.1.1 Purpose of the Peripheral

The UART peripheral is based on the industry standard TL16C550 asynchronous communications element, which in turn is a functional upgrade of the TL16C450. Functionally similar to the TL16C450 on power up (single character or TL16C450 mode), the UART can be placed in an alternate FIFO (TL16C550) mode. This relieves the CPU of excessive software overhead by buffering received and transmitted characters. The receiver and transmitter FIFOs store up to 16 bytes including three additional bits of error status per byte for the receiver FIFO.

The UART performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. The CPU can read the UART status at any time. The UART includes control capability and a processor interrupt system that can be tailored to minimize software management of the communications link.

The UART includes a programmable baud generator capable of dividing the UART input clock by divisors from 1 to 65535 and producing a 16 $\times$  reference clock or a 13 $\times$  reference clock for the internal transmitter and receiver logic. For detailed timing and electrical specifications for the UART, see your device-specific data manual.

### 25.1.2 Features

Check your device-specific data manual to see the list of features that are supported and that are not supported by the UART.

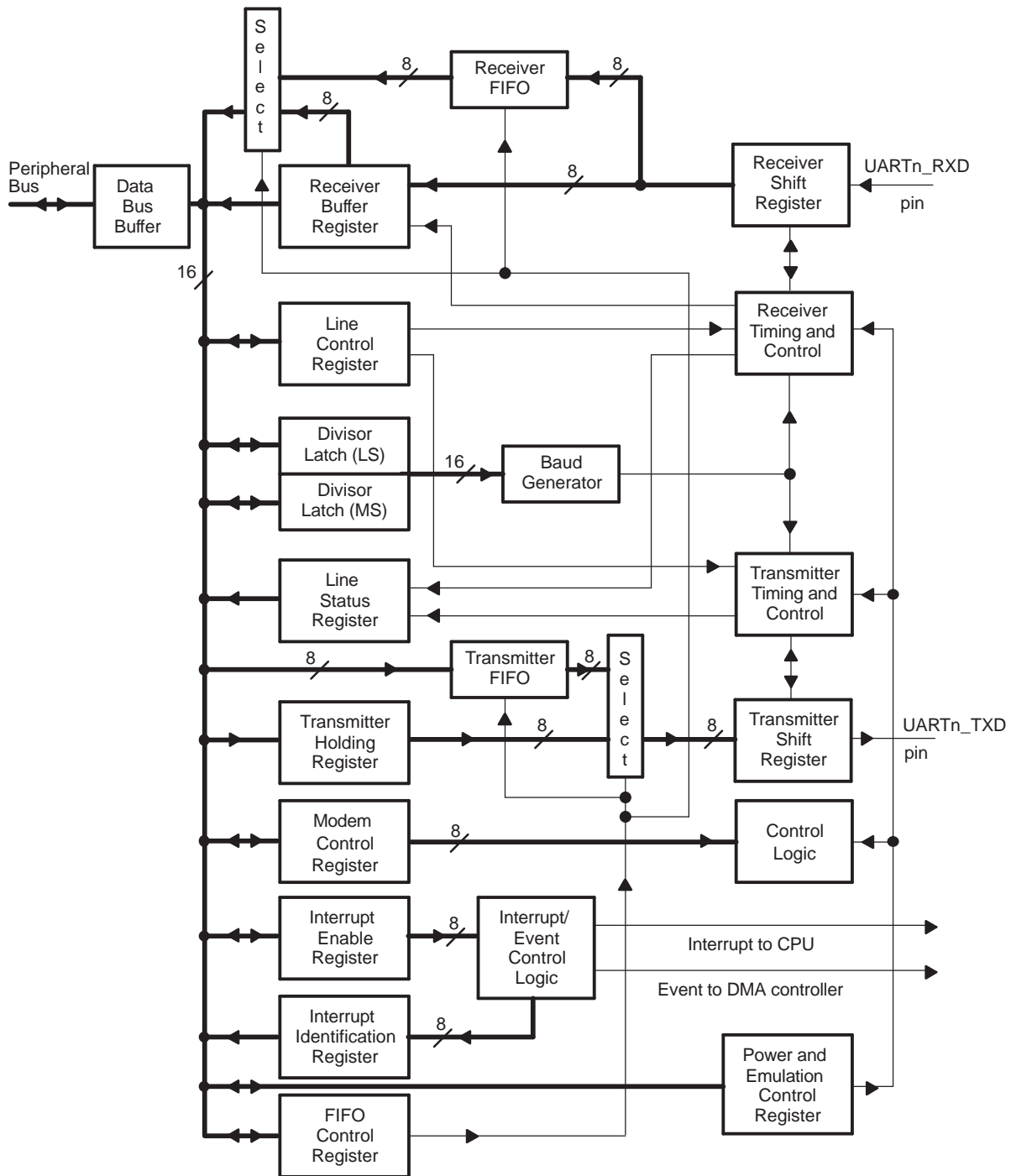
### 25.1.3 Functional Block Diagram

A functional block diagram of the UART is shown in [Figure 25-1](#).

### 25.1.4 Industry Standard(s) Compliance Statement

The UART peripheral is based on the industry standard TL16C550 asynchronous communications element, which is a functional upgrade of the TL16C450. The information in this chapter assumes you are familiar with these standards.

Figure 25-1. UART Block Diagram



NOTE: The value *n* indicates the applicable UART; that is, UART0, UART1, and so on.

## 25.2 Peripheral Architecture

### 25.2.1 Clock Generation and Control

The UART bit clock is derived from an input clock to the UART. See your device-specific data manual to check the maximum data rate supported by the UART.

Figure 25-2 is a conceptual clock generation diagram for the UART. The processor clock generator receives a signal from an external clock source and produces a UART input clock with a programmed frequency. The UART contains a programmable baud generator that takes an input clock and divides it by a divisor in the range between 1 and  $(2^{16} - 1)$  to produce a baud clock (BCLK). The frequency of BCLK is sixteen times (16 $\times$ ) the baud rate (each received or transmitted bit lasts 16 BCLK cycles) or thirteen times (13 $\times$ ) the baud rate (each received or transmitted bit lasts 13 BCLK cycles). When the UART is receiving, the bit is sampled in the 8th BCLK cycle for 16 $\times$  over sampling mode and on the 6th BCLK cycle for 13 $\times$  over-sampling mode. The 16 $\times$  or 13 $\times$  reference clock is selected by configuring the OSM\_SEL bit in the mode definition register (MDR). The formula to calculate the divisor is:

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 16} \quad [\text{MDR.OSM\_SEL} = 0]$$

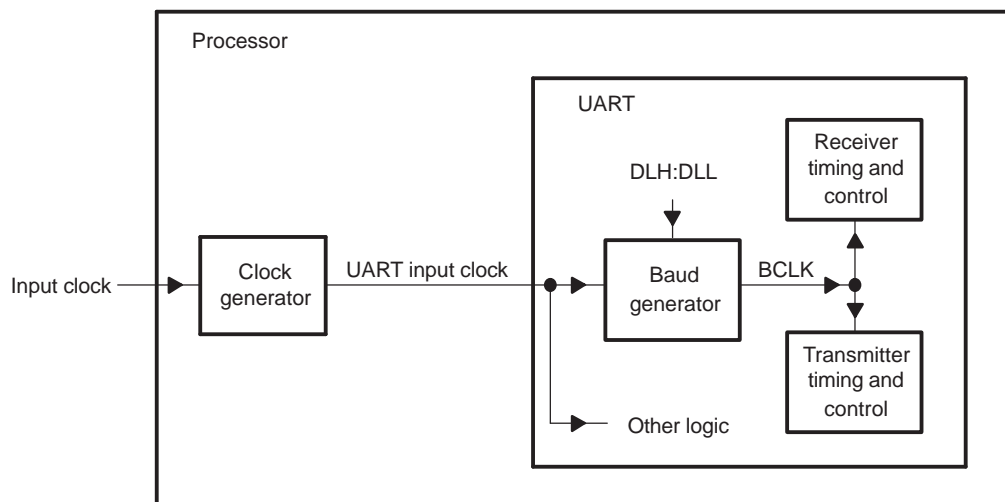
$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 13} \quad [\text{MDR.OSM\_SEL} = 1]$$

Two 8-bit register fields (DLH and DLL), called divisor latches, hold this 16-bit divisor. DLH holds the most significant bits of the divisor, and DLL holds the least significant bits of the divisor. For information about these register fields, see Section 25.3. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

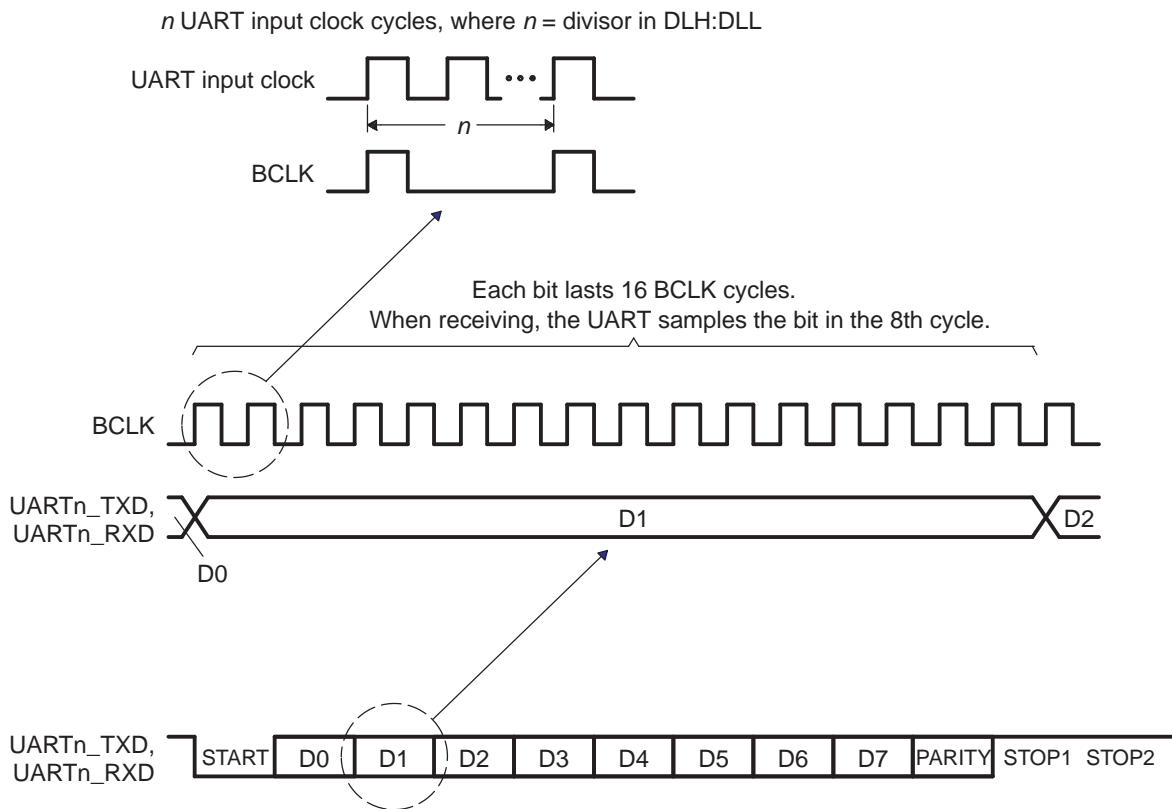
Figure 25-3 summarizes the relationship between the transferred data bit, BCLK, and the UART input clock. Note that the timing relationship depicted in Figure 25-3 shows that each bit lasts for 16 BCLK cycles. This is in case of 16 $\times$  over-sampling mode. For 13 $\times$  over-sampling mode each bit lasts for 13 BCLK cycles.

Example baud rates and divisor values relative to a 150-MHz UART input clock and 16 $\times$  over-sampling mode are shown in Table 25-1.

**Figure 25-2. UART Clock Generation Diagram**



**Figure 25-3. Relationships Between Data Bit, BCLK, and UART Input Clock**



**Table 25-1. Baud Rate Examples for 150-MHZ UART Input Clock and 16x Over-sampling Mode**

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

**Table 25-2. Baud Rate Examples for 150-MHZ UART Input Clock and 13x Over-sampling Mode**

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	4808	2399	-0.01
4800	2404	4799.646	-0.01
9600	1202	9599.386	-0.01
19200	601	19198.771	-0.01
38400	300	38461.538	0.16
56000	206	56011.949	0.02
128000	90	128205.128	0.16
3000000	4	2884615.385	-4.00

## 25.2.2 Signal Descriptions

The UARTs utilize a minimal number of signal connections to interface with external devices. The UART signal descriptions are included in [Table 25-3](#). Note that the number of UARTs and their supported features vary on each device, see your device-specific data manual for more details.

**Table 25-3. UART Signal Descriptions**

Signal Name <sup>(1)</sup>	Signal Type	Function
UART <sub>n</sub> _TXD	Output	Serial data transmit
UART <sub>n</sub> _RXD	Input	Serial data receive
UART <sub>n</sub> _CTS <sup>(2)</sup>	Input	Clear-to-Send handshaking signal
UART <sub>n</sub> _RTS <sup>(2)</sup>	Output	Request-to-Send handshaking signal

<sup>(1)</sup> The value *n* indicates the applicable UART; that is, UART0, UART1, etc.

<sup>(2)</sup> This signal is not supported in all UARTs. See your device-specific data manual to check if it is supported.

## 25.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. See your device-specific data manual to determine how pin multiplexing affects the UART.

## 25.2.4 Protocol Description

### 25.2.4.1 Transmission

The UART transmitter section includes a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

### 25.2.4.2 Reception

The UART receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Receiver section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

### 25.2.4.3 Data Format

The UART transmits in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + STOP bit (1, 1.5, 2)

It transmits 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1, 1.5, or 2 STOP bits, depending on the STOP bit selection.

The UART receives in the following format:

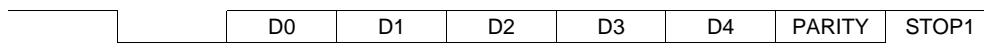
1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + 1 STOP bit

It receives 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1 STOP bit.

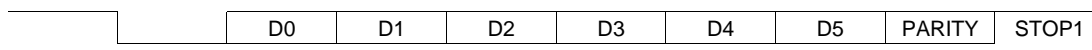
The protocol formats are shown in [Figure 25-4](#).

**Figure 25-4. UART Protocol Formats**

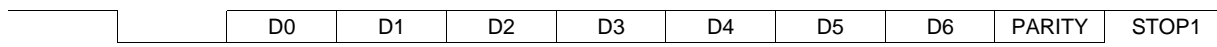
Transmit/Receive for 5-bit data, parity Enable, 1 STOP bit



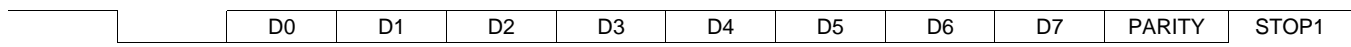
Transmit/Receive for 6-bit data, parity Enable, 1 STOP bit



Transmit/Receive for 7-bit data, parity Enable, 1 STOP bit



Transmit/Receive for 8-bit data, parity Enable, 1 STOP bit



## 25.2.5 Operation

### 25.2.5.1 Transmission

The UART transmitter section includes a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

THR receives data from the internal data bus, and when TSR is ready, the UART moves the data from THR to TSR. The UART serializes the data in TSR and transmits the data on the UART<sub>n</sub>\_TXD pin.

In the non-FIFO mode, if THR is empty and the THR empty (THRE) interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when a character is loaded into THR or the interrupt identification register (IIR) is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or IIR is read.

### 25.2.5.2 Reception

The UART receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the 16x receiver clock. Receiver section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

RSR receives the data bits from the UART<sub>n</sub>\_RXD pin. Then RSR concatenates the data bits and moves the resulting value into RBR (or the receiver FIFO). The UART also stores three bits of error status information next to each received character, to record a parity error, framing error, or break.

In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register (FCR), and it is cleared when the FIFO contents drop below the trigger level.

### 25.2.5.3 FIFO Modes

The following two modes can be used for servicing the receiver and transmitter FIFOs:

- FIFO interrupt mode. The FIFO is enabled and the associated interrupts are enabled. Interrupts are sent to the CPU to indicate when specific events occur.
- FIFO poll mode. The FIFO is enabled but the associated interrupts are disabled. The CPU polls status bits to detect specific events.

Because the receiver FIFO and the transmitter FIFO are controlled separately, either one or both can be placed into the interrupt mode or the poll mode.

#### 25.2.5.3.1 FIFO Interrupt Mode

When the receiver FIFO is enabled in the FIFO control register (FCR) and the receiver interrupts are enabled in the interrupt enable register (IER), the interrupt mode is selected for the receiver FIFO. The following are important points about the receiver interrupts:

- The receiver data-ready interrupt is issued to the CPU when the FIFO has reached the trigger level that is programmed in FCR. It is cleared when the CPU or the DMA controller reads enough characters from the FIFO such that the FIFO drops below its programmed trigger level.
- The receiver line status interrupt is generated in response to an overrun error, a parity error, a framing error, or a break. This interrupt has higher priority than the receiver data-ready interrupt. For details, see [Section 25.2.8](#).
- The data-ready (DR) bit in the line status register (LSR) indicates the presence or absence of characters in the receiver FIFO. The DR bit is set when a character is transferred from the receiver shift register (RSR) to the empty receiver FIFO. The DR bit remains set until the FIFO is empty again.
- A receiver time-out interrupt occurs if all of the following conditions exist:
  - At least one character is in the FIFO,
  - The most recent character was received more than four continuous character times ago. A character time is the time allotted for 1 START bit,  $n$  data bits, 1 PARITY bit, and 1 STOP bit, where  $n$  depends on the word length selected with the WLS bits in the line control register (LCR). See [Table 25-4](#).
  - The most recent read of the FIFO has occurred more than four continuous character times before.
- Character times are calculated by using the baud rate.
- When a receiver time-out interrupt has occurred, it is cleared and the time-out timer is cleared when the CPU or the EDMA controller reads one character from the receiver FIFO. The interrupt is also cleared if a new character is received in the FIFO or if the URRST bit is cleared in the power and emulation management register (PWREMU\_MGMT).
- If a receiver time-out interrupt has not occurred, the time-out timer is cleared after a new character is received or after the CPU or EDMA reads the receiver FIFO.

When the transmitter FIFO is enabled in FCR and the transmitter holding register empty (THRE) interrupt is enabled in IER, the interrupt mode is selected for the transmitter FIFO. The THRE interrupt occurs when the transmitter FIFO is empty. It is cleared when the transmitter hold register (THR) is loaded (1 to 16 characters may be written to the transmitter FIFO while servicing this interrupt) or the interrupt identification register (IIR) is read.

**Table 25-4. Character Time for Word Lengths**

Word Length ( $n$ )	Character Time	Four Character Times
5	Time for 8 bits	Time for 32 bits
6	Time for 9 bits	Time for 36 bits
7	Time for 10 bits	Time for 40 bits
8	Time for 11 bits	Time for 44 bits



### 25.2.5.3.2 FIFO Poll Mode

When the receiver FIFO is enabled in the FIFO control register (FCR) and the receiver interrupts are disabled in the interrupt enable register (IER), the poll mode is selected for the receiver FIFO. Similarly, when the transmitter FIFO is enabled and the transmitter interrupts are disabled, the transmitted FIFO is in the poll mode. In the poll mode, the CPU detects events by checking bits in the line status register (LSR):

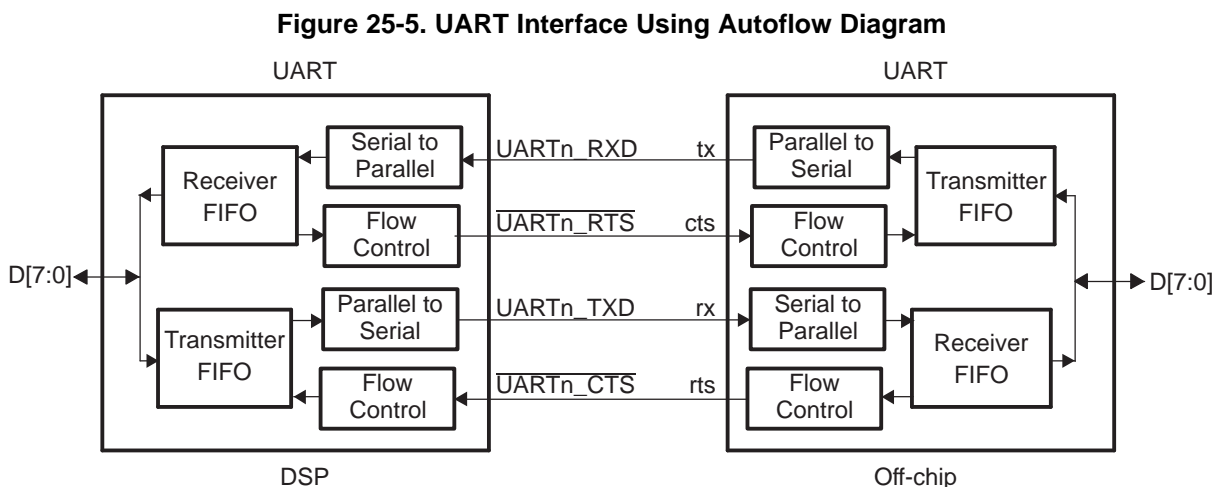
- The RXFIFOE bit indicates whether there are any errors in the receiver FIFO.
- The TEMT bit indicates that both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
- The THRE bit indicates when THR is empty.
- The BI (break), FE (framing error), PE (parity error), and OE (overrun error) bits specify which error or errors have occurred.
- The DR (data-ready) bit is set as long as there is at least one byte in the receiver FIFO.

Also, in the FIFO poll mode:

- The interrupt identification register (IIR) is not affected by any events because the interrupts are disabled.
- The UART does not indicate when the receiver FIFO trigger level is reached or when a receiver time-out occurs.

### 25.2.5.4 Autoflow Control

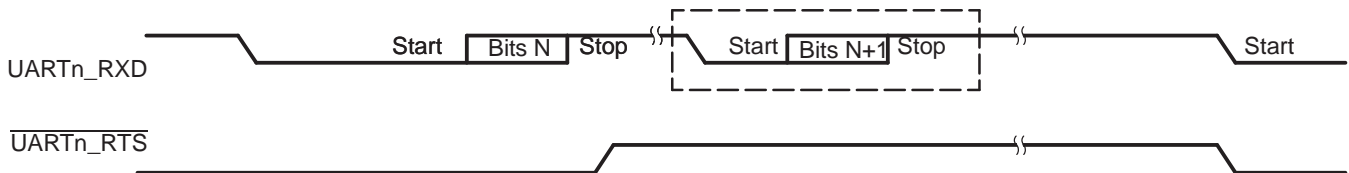
The UART can employ autoflow control by connecting the  $\overline{\text{UARTn\_CTS}}$  and  $\overline{\text{UARTn\_RTS}}$  signals. Note that all UARTs do not support autoflow control, see your device-specific data manual for supported features. The  $\overline{\text{UARTn\_CTS}}$  input must be active before the transmitter FIFO can transmit data. The  $\overline{\text{UARTn\_RTS}}$  becomes active when the receiver needs more data and notifies the sending device. When  $\overline{\text{UARTn\_RTS}}$  is connected to  $\overline{\text{UARTn\_CTS}}$ , data transmission does not occur unless the receiver FIFO has space for the data. Therefore, when two UARTs are connected as shown in Figure 25-5 with autoflow enabled, overrun errors are eliminated.



#### 25.2.5.4.1 UARTn\_RTS Behavior

UARTn\_RTS data flow control originates in the receiver block (see [Figure 25-1](#)). When the receiver FIFO level reaches a trigger level of 1, 4, 8, or 14 (see [Figure 25-6](#)), UARTn\_RTS is deasserted. The sending UART may send an additional byte after the trigger level is reached (assuming the sending UART has another byte to send), because it may not recognize the deassertion of UARTn\_RTS until after it has begun sending the additional byte. For trigger level 1, 4, and 8, UARTn\_RTS is automatically reasserted once the receiver FIFO is emptied. For trigger level 14, UARTn\_RTS is automatically reasserted once the receiver FIFO drops below the trigger level.

**Figure 25-6. Autoflow Functional Timing Waveforms for UARTn\_RTS**

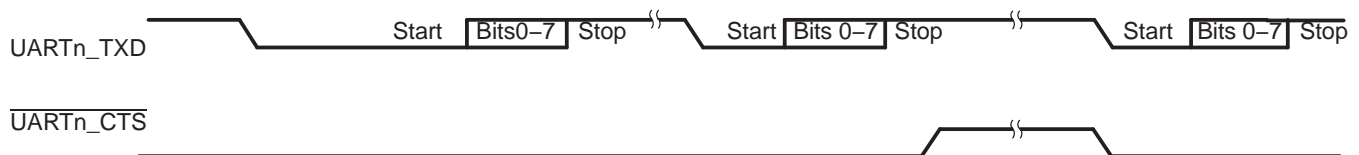


- (1) N = Receiver FIFO trigger level.
- (2) The two blocks in dashed lines cover the case where an additional byte is sent.

#### 25.2.5.4.2 UARTn\_CTS Behavior

The transmitter checks UARTn\_CTS before sending the next data byte. If UARTn\_CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, UARTn\_CTS must be released before the middle of the last STOP bit that is currently being sent (see [Figure 25-7](#)). When flow control is enabled, UARTn\_CTS level changes do not trigger interrupts because the device automatically controls its own transmitter. Without autoflow control, the transmitter sends any data present in the transmitter FIFO and a receiver overrun error may result.

**Figure 25-7. Autoflow Functional Timing Waveforms for UARTn\_CTS**



- (1) When UARTn\_CTS is active (low), the transmitter keeps sending serial data out.
- (2) When UARTn\_CTS goes high before the middle of the last STOP bit of the current byte, the transmitter finishes sending the current byte but it does not send the next byte.
- (3) When UARTn\_CTS goes from high to low, the transmitter begins sending data again.

#### 25.2.5.5 Loopback Control

The UART can be placed in the diagnostic mode using the LOOP bit in the modem control register (MCR), which internally connects the UART output back to the UART input. In this mode, the transmit and receive data paths, the transmitter and receiver interrupts, and the modem control interrupts can be verified without connecting to another UART.

## 25.2.6 Reset Considerations

### 25.2.6.1 Software Reset Considerations

Two bits in the power and emulation management register (PWREMU\_MGMT) control resetting the parts of the UART:

- The UTRST bit controls resetting the transmitter only. If UTRST = 1, the transmitter is active; if UTRST = 0, the transmitter is in reset.
- The URRST bit controls resetting the receiver only. If URRST = 1, the receiver is active; if URRST = 0, the receiver is in reset.

In each case, putting the receiver and/or transmitter in reset will reset the state machine of the affected portion but does not affect the UART registers.

### 25.2.6.2 Hardware Reset Considerations

When the processor RESET pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the UART state machine is reset and the UART registers are forced to their default states. The default states of the registers are shown in [Section 25.3](#).

## 25.2.7 Initialization

The following steps are required to initialize the UART:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Set the desired baud rate by writing the appropriate clock divisor values to the divisor latch registers (DLL and DLH).
3. If the FIFOs will be used, select the desired trigger level and enable the FIFOs by writing the appropriate values to the FIFO control register (FCR). The FIFOEN bit in FCR must be set first, before the other bits in FCR are configured.
4. Choose the desired protocol settings by writing the appropriate values to the line control register (LCR).
5. If autoflow control is desired, write appropriate values to the modem control register (MCR). Note that all UARTs do not support autoflow control, see your device-specific data manual for supported features.
6. Choose the desired response to emulation suspend events by configuring the FREE bit and enable the UART by setting the UTRST and URRST bits in the power and emulation management register (PWREMU\_MGMT).

## 25.2.8 Interrupt Support

### 25.2.8.1 Interrupt Events and Requests

The UART generates the interrupt requests described in [Table 25-5](#). All requests are multiplexed through an arbiter to a single UART interrupt request to the CPU, as shown in [Figure 25-8](#). Each of the interrupt requests has an enable bit in the interrupt enable register (IER) and is recorded in the interrupt identification register (IIR).

If an interrupt occurs and the corresponding enable bit is set to 1, the interrupt request is recorded in IIR and is forwarded to the CPU. If an interrupt occurs and the corresponding enable bit is cleared to 0, the interrupt request is blocked. The interrupt request is neither recorded in IIR nor forwarded to the CPU.

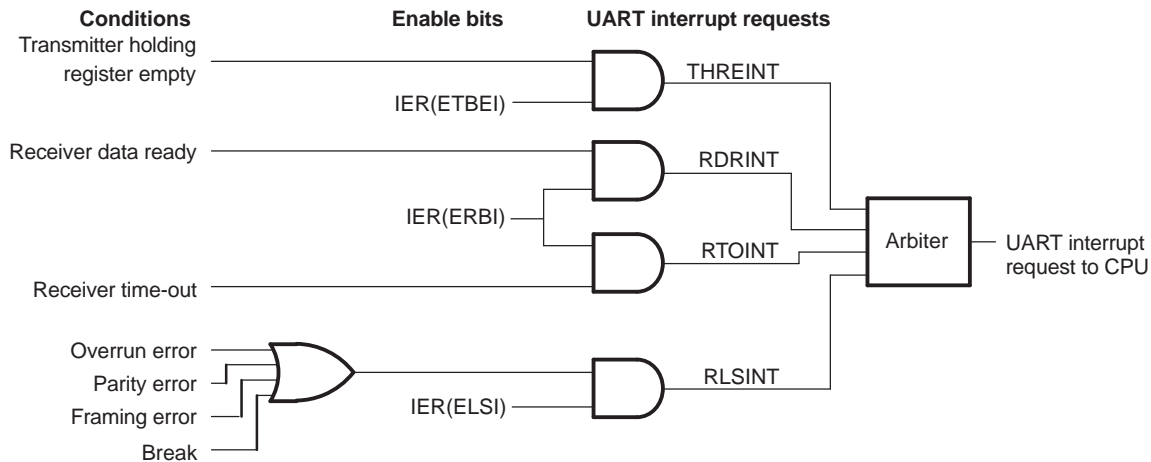
### 25.2.8.2 Interrupt Multiplexing

The UARTs have dedicated interrupt signals to the CPU and the interrupts are not multiplexed with any other interrupt source.

**Table 25-5. UART Interrupt Requests Descriptions**

UART Interrupt Request	Interrupt Source	Comment
THREINT	THR-empty condition: The transmitter holding register (THR) or the transmitter FIFO is empty. All of the data has been copied from THR to the transmitter shift register (TSR).	If THREINT is enabled in IER, by setting the ETBEI bit, it is recorded in IIR. As an alternative to using THREINT, the CPU can poll the THRE bit in the line status register (LSR).
RDAINT	Receive data available in non-FIFO mode or trigger level reached in the FIFO mode.	If RDAINT is enabled in IER, by setting the ERBI bit, it is recorded in IIR. As an alternative to using RDAINT, the CPU can poll the DR bit in the line status register (LSR). In the FIFO mode, this is not a functionally equivalent alternative because the DR bit does not respond to the FIFO trigger level. The DR bit only indicates the presence or absence of unread characters.
RTOINT	Receiver time-out condition (in the FIFO mode only): No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 25-4), and there is at least one character in the receiver FIFO during this time.	The receiver time-out interrupt prevents the UART from waiting indefinitely, in the case when the receiver FIFO level is below the trigger level and thus does not generate a receiver data-ready interrupt. If RTOINT is enabled in IER, by setting the ERBI bit, it is recorded in IIR. There is no status bit to reflect the occurrence of a time-out condition.
RLSINT	Receiver line status condition: An overrun error, parity error, framing error, or break has occurred.	If RLSINT is enabled in IER, by setting the ELSI bit, it is recorded in IIR. As an alternative to using RLSINT, the CPU can poll the following bits in the line status register (LSR): overrun error indicator (OE), parity error indicator (PE), framing error indicator (FE), and break indicator (BI).

**Figure 25-8. UART Interrupt Request Enable Paths**



### 25.2.9 DMA Event Support

In the FIFO mode, the UART generates the following two DMA events:

- **Receive event (URXEVT):** The trigger level for the receiver FIFO (1, 4, 8, or 14 characters) is set with the RXFIFTL bit in the FIFO control register (FCR). Every time the trigger level is reached or a receiver time-out occurs, the UART sends a receive event to the EDMA controller. In response, the EDMA controller reads the data from the receiver FIFO by way of the receiver buffer register (RBR). Note that the receive event is not asserted if the data at the top of the receiver FIFO is erroneous even if the trigger level has been reached.
- **Transmit event (UTXEVT):** When the transmitter FIFO is empty (when the last byte in the transmitter FIFO has been copied to the transmitter shift register), the UART sends an UTXEVT signal to the EDMA controller. In response, the EDMA controller refills the transmitter FIFO by way of the transmitter holding register (THR). The UTXEVT signal is also sent to the DMA controller when the UART is taken out of reset using the UTRST bit in the power and emulation management register (PWREMU\_MGMT).

Activity in DMA channels can be synchronized to these events. In the non-FIFO mode, the UART generates no DMA events. Any DMA channel synchronized to either of these events must be enabled at the time the UART event is generated. Otherwise, the DMA channel will miss the event and, unless the UART generates a new event, no data transfer will occur.

### 25.2.10 Power Management

The UART peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the UART peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

### 25.2.11 Emulation Considerations

The FREE bit in the power and emulation management register (PWREMU\_MGMT) determines how the UART responds to an emulation suspend event such as an emulator halt or breakpoint. If FREE = 0 and a transmission is in progress, the UART halts after completing the one-word transmission; if FREE = 0 and a transmission is not in progress, the UART halts immediately. If FREE = 1, the UART does not halt and continues operating normally.

Note also that most emulator accesses are transparent to UART operation. Emulator read operations do not affect any register contents, status bits, or operating states, with the exception of the interrupt identification register (IIR). Emulator writes, however, may affect register contents and may affect UART operation, depending on what register is accessed and what value is written.

The UART registers can be read from or written to during emulation suspend events, even if the UART activity has stopped.

### 25.2.12 Exception Processing

#### 25.2.12.1 Divisor Latch Not Programmed

Since the processor reset signal has no effect on the divisor latch, the divisor latch will have an unknown value after power up. If the divisor latch is not programmed after power up, the baud clock (BCLK) will not operate and will instead be set to a constant logic 1 state.

The divisor latch values should always be reinitialized following a processor reset.

#### 25.2.12.2 Changing Operating Mode During Busy Serial Communication

Since the serial link characteristics are based on how the control registers are programmed, the UART will expect the control registers to be static while it is busy engaging in a serial communication. Therefore, changing the control registers while the module is still busy communicating with another serial device will most likely cause an error condition and should be avoided.

## 25.3 Registers

The system programmer has access to and control over any of the UART registers that are listed in [Table 25-6](#). These registers, which control UART operations, receive data, and transmit data, are available at 32-bit addresses in the device memory map. See your device-specific data manual for the memory address of these registers.

- RBR, THR, and DLL share one address. When the DLAB bit in LCR is 0, reading from the address gives the content of RBR, and writing to the address modifies THR. When DLAB = 1, all accesses at the address read or modify DLL. DLL can also be accessed with address offset 20h.
- IER and DLH share one address. When DLAB = 0, all accesses read or modify IER. When DLAB = 1, all accesses read or modify DLH. DLH can also be accessed with address offset 24h.
- IIR and FCR share one address. Regardless of the value of the DLAB bit, reading from the address gives the content of IIR, and writing modifies FCR.

**Table 25-6. UART Registers**

Offset	Acronym	Register Description	Section
0h	RBR	Receiver Buffer Register (read only)	<a href="#">Section 25.3.1</a>
0h	THR	Transmitter Holding Register (write only)	<a href="#">Section 25.3.2</a>
4h	IER	Interrupt Enable Register	<a href="#">Section 25.3.3</a>
8h	IIR	Interrupt Identification Register (read only)	<a href="#">Section 25.3.4</a>
8h	FCR	FIFO Control Register (write only)	<a href="#">Section 25.3.5</a>
Ch	LCR	Line Control Register	<a href="#">Section 25.3.6</a>
10h	MCR	Modem Control Register	<a href="#">Section 25.3.7</a>
14h	LSR	Line Status Register	<a href="#">Section 25.3.8</a>
18h	MSR	Modem Status Register	<a href="#">Section 25.3.9</a>
1Ch	SCR	Scratch Pad Register	<a href="#">Section 25.3.10</a>
20h	DLL	Divisor LSB Latch	<a href="#">Section 25.3.11</a>
24h	DLH	Divisor MSB Latch	<a href="#">Section 25.3.11</a>
28h	REVID1	Revision Identification Register 1	<a href="#">Section 25.3.12</a>
2Ch	REVID2	Revision Identification Register 2	<a href="#">Section 25.3.12</a>
30h	PWREMU_MGMT	Power and Emulation Management Register	<a href="#">Section 25.3.13</a>
34h	MDR	Mode Definition Register	<a href="#">Section 25.3.14</a>

### 25.3.1 Receiver Buffer Register (RBR)

The receiver buffer register (RBR) is shown in [Figure 25-9](#) and described in [Table 25-7](#).

The UART receiver section consists of a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the 16x receiver clock or 13x receiver clock by programming OSM\_SEL bit field of MDR register. Receiver section control is a function of the line control register (LCR).

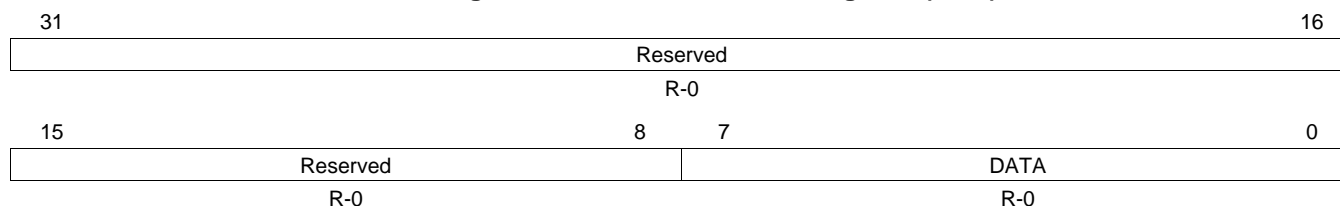
RSR receives serial data from the UARTn\_RXD pin. Then RSR concatenates the data and moves it into RBR (or the receiver FIFO). In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled (DR = 1 in IER), an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register (FCR), and it is cleared when the FIFO contents drop below the trigger level.

#### Access considerations:

RBR, THR, and DLL share one address. To read RBR, write 0 to the DLAB bit in LCR, and read from the shared address. When DLAB = 0, writing to the shared address modifies THR. When DLAB = 1, all accesses at the shared address read or modify DLL.

DLL also has a dedicated address. If you use the dedicated address, you can keep DLAB = 0, so that RBR and THR are always selected at the shared address.

**Figure 25-9. Receiver Buffer Register (RBR)**



LEGEND: R = Read only; -n = value after reset

**Table 25-7. Receiver Buffer Register (RBR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DATA	0-FFh	Received data

### 25.3.2 Transmitter Holding Register (THR)

The transmitter holding register (THR) is shown in [Figure 25-10](#) and described in [Table 25-8](#).

The UART transmitter section consists of a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the line control register (LCR).

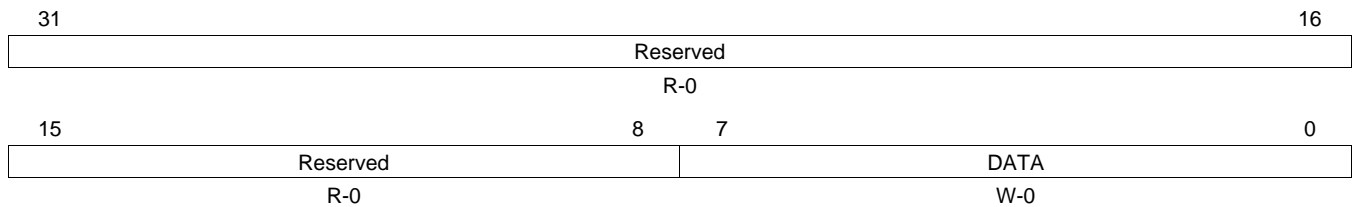
THR receives data from the internal data bus and when TSR is idle, the UART moves the data from THR to TSR. The UART serializes the data in TSR and transmits the data on the TX pin. In the non-FIFO mode, if THR is empty and the THR empty (THRE) interrupt is enabled (ETBEI = 1 in IER), an interrupt is generated. This interrupt is cleared when a character is loaded into THR or the interrupt identification register (IIR) is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or IIR is read.

**Access considerations:**

RBR, THR, and DLL share one address. To load THR, write 0 to the DLAB bit of LCR, and write to the shared address. When DLAB = 0, reading from the shared address gives the content of RBR. When DLAB = 1, all accesses at the address read or modify DLL.

DLL also has a dedicated address. If you use the dedicated address, you can keep DLAB = 0, so that RBR and THR are always selected at the shared address.

**Figure 25-10. Transmitter Holding Register (THR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 25-8. Transmitter Holding Register (THR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DATA	0-FFh	Data to transmit



### 25.3.3 Interrupt Enable Register (IER)

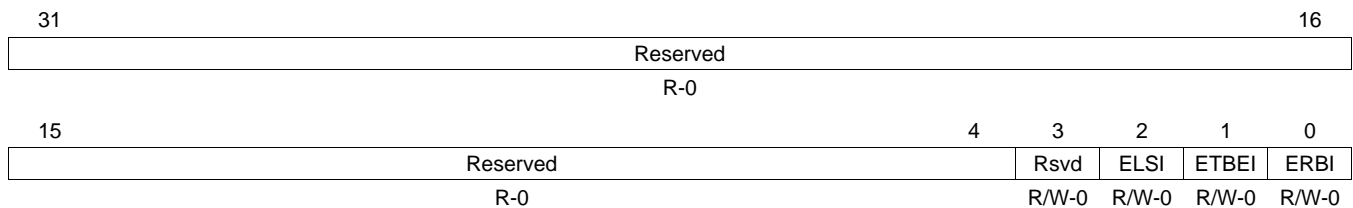
The interrupt enable register (IER) is used to individually enable or disable each type of interrupt request that can be generated by the UART. Each interrupt request that is enabled in IER is forwarded to the CPU. IER is shown in Figure 25-11 and described in Table 25-9.

#### Access considerations:

IER and DLH share one address. To read or modify IER, write 0 to the DLAB bit in LCR. When DLAB = 1, all accesses at the shared address read or modify DLH.

DLH also has a dedicated address. If you use the dedicated address, you can keep DLAB = 0, so that IER is always selected at the shared address.

**Figure 25-11. Interrupt Enable Register (IER)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-9. Interrupt Enable Register (IER) Field Descriptions**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	EDSSI	0	Enable Modem Status Interrupt
2	ELSI	0	Receiver line status interrupt enable. Receiver line status interrupt is disabled.
		1	Receiver line status interrupt is enabled.
1	ETBEI	0	Transmitter holding register empty interrupt enable. Transmitter holding register empty interrupt is disabled.
		1	Transmitter holding register empty interrupt is enabled.
0	ERBI	0	Receiver data available interrupt and character timeout indication interrupt enable. Receiver data available interrupt and character timeout indication interrupt is disabled.
		1	Receiver data available interrupt and character timeout indication interrupt is enabled.

### 25.3.4 Interrupt Identification Register (IIR)

The interrupt identification register (IIR) is a read-only register at the same address as the FIFO control register (FCR), which is a write-only register. When an interrupt is generated and enabled in the interrupt enable register (IER), IIR indicates that an interrupt is pending in the IPEND bit and encodes the type of interrupt in the INTID bits. Reading IIR clears any THR empty (THRE) interrupts that are pending.

IIR is shown in [Figure 25-12](#) and described in [Figure 25-12](#).

The UART has an on-chip interrupt generation and prioritization capability that permits flexible communication with the CPU. The UART provides three priority levels of interrupts:

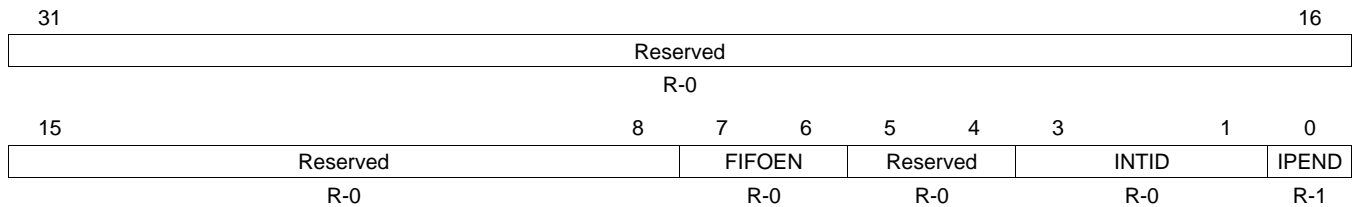
- Priority 1 - Receiver line status (highest priority)
- Priority 2 - Receiver data ready or receiver timeout
- Priority 3 - Transmitter holding register empty

The FIFOEN bit in IIR can be checked to determine whether the UART is in the FIFO mode or the non-FIFO mode.

**Access consideration:**

IIR and FCR share one address. Regardless of the value of the DLAB bit in LCR, reading from the address gives the content of IIR, and writing to the address modifies FCR.

**Figure 25-12. Interrupt Identification Register (IIR)**



LEGEND: R = Read only; -n = value after reset

**Table 25-10. Interrupt Identification Register (IIR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	FIFOEN	0-3h 0 1h-2h 3h	FIFOs enabled. Non-FIFO mode Reserved FIFOs are enabled. FIFOEN bit in the FIFO control register (FCR) is set to 1.
5-4	Reserved	0	Reserved
3-1	INTID	0-7h 0 1h 2h 3h 4h-5h 6h 7h	Interrupt type. See <a href="#">Table 25-11</a> . Reserved Transmitter holding register empty (priority 3) Receiver data available (priority 2) Receiver line status (priority 1, highest) Reserved Character timeout indication (priority 2) Reserved
0	IPEND	0 1	Interrupt pending. When any UART interrupt is generated and is enabled in IER, IPEND is forced to 0. IPEND remains 0 until all pending interrupts are cleared or until a hardware reset occurs. If no interrupts are enabled, IPEND is never forced to 0. Interrupts pending. No interrupts pending.

**Table 25-11. Interrupt Identification and Interrupt Clearing Information**

Priority Level	IIR Bits				Interrupt Type	Interrupt Source	Event That Clears Interrupt
	3	2	1	0			
None	0	0	0	1	None	None	None
1	0	1	1	0	Receiver line status	Overrun error, parity error, framing error, or break is detected.	For an overrun error, reading the line status register (LSR) clears the interrupt. For a parity error, framing error, or break, the interrupt is cleared only after all the erroneous data have been read.
2	0	1	0	0	Receiver data-ready	Non-FIFO mode: Receiver data is ready. FIFO mode: Trigger level reached. If four character times (see <a href="#">Table 25-4</a> ) pass with no access of the FIFO, the interrupt is asserted again.	Non-FIFO mode: The receiver buffer register (RBR) is read. FIFO mode: The FIFO drops below the trigger level. <sup>(1)</sup>
2	1	1	0	0	Receiver time-out	FIFO mode only: No characters have been removed from or input to the receiver FIFO during the last four character times (see <a href="#">Table 25-4</a> ), and there is at least one character in the receiver FIFO during this time.	One of the following events: <ul style="list-style-type: none"> <li>A character is read from the receiver FIFO.<sup>(1)</sup></li> <li>A new character arrives in the receiver FIFO.</li> <li>The URRST bit in the power and emulation management register (PWREMU_MGMT) is loaded with 0.</li> </ul>
3	0	0	1	0	Transmitter holding register empty	Non-FIFO mode: Transmitter holding register (THR) is empty. FIFO mode: Transmitter FIFO is empty.	A character is written to the transmitter holding register (THR) or the interrupt identification register (IIR) is read.

<sup>(1)</sup> In the FIFO mode, the receiver data-ready interrupt or receiver time-out interrupt is cleared by the CPU or by the DMA controller, whichever reads from the receiver FIFO first.

### 25.3.5 FIFO Control Register (FCR)

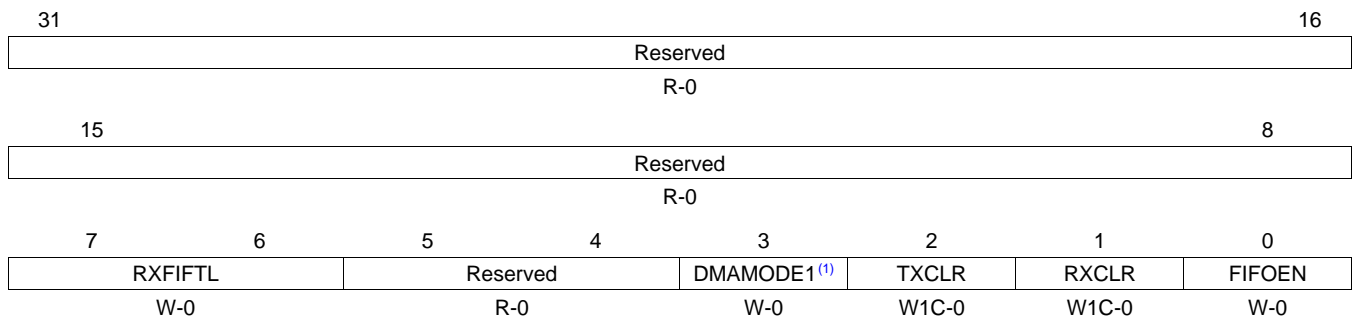
The FIFO control register (FCR) is a write-only register at the same address as the interrupt identification register (IIR), which is a read-only register. Use FCR to enable and clear the FIFOs and to select the receiver FIFO trigger level. FCR is shown in [Figure 25-13](#) and described in [Table 25-12](#). The FIFOEN bit must be set to 1 before other FCR bits are written to or the FCR bits are not programmed.

#### Access consideration:

IIR and FCR share one address. Regardless of the value of the DLAB bit, reading from the address gives the content of IIR, and writing to the address modifies FCR.

#### CAUTION

For proper communication between the UART and the EDMA controller, the DMAMODE1 bit must be set to 1. Always write a 1 to the DMAMODE1 bit, and after a hardware reset, change the DMAMODE1 bit from 0 to 1.

**Figure 25-13. FIFO Control Register (FCR)**

LEGEND: R = Read only; W = Write only; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

<sup>(1)</sup> Always write 1 to the DMAMODE1 bit. After a hardware reset, change the DMAMODE1 bit from 0 to 1. DMAMODE = 1 is required for proper communication between the UART and the DMA controller.

**Table 25-12. FIFO Control Register (FCR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	RXFIFTL	0-3h	Receiver FIFO trigger level. RXFIFTL sets the trigger level for the receiver FIFO. When the trigger level is reached, a receiver data-ready interrupt is generated (if the interrupt request is enabled). Once the FIFO drops below the trigger level, the interrupt is cleared.
		0	1 byte
		1h	4 bytes
		2h	8 bytes
		3h	14 bytes
5-4	Reserved	0	Reserved
3	DMAMODE1		DMA MODE1 enable if FIFOs are enabled. Always write 1 to DMAMODE1. After a hardware reset, change DMAMODE1 from 0 to 1. DMAMODE1 = 1 is a requirement for proper communication between the UART and the EDMA controller.
		0	DMA MODE1 is disabled.
		1	DMA MODE1 is enabled.
2	TXCLR		Transmitter FIFO clear. Write a 1 to TXCLR to clear the bit.
		0	No effect.
		1	Clears transmitter FIFO and resets the transmitter FIFO counter. The shift register is not cleared.
1	RXCLR		Receiver FIFO clear. Write a 1 to RXCLR to clear the bit.
		0	No effect.
		1	Clears receiver FIFO and resets the receiver FIFO counter. The shift register is not cleared.
0	FIFOEN		Transmitter and receiver FIFOs mode enable. FIFOEN must be set before other FCR bits are written to or the FCR bits are not programmed. Clearing this bit clears the FIFO counters.
		0	Non-FIFO mode. The transmitter and receiver FIFOs are disabled, and the FIFO pointers are cleared.
		1	FIFO mode. The transmitter and receiver FIFOs are enabled.

### 25.3.6 Line Control Register (LCR)

The line control register (LCR) is shown in [Figure 25-14](#) and described in [Table 25-13](#).

The system programmer controls the format of the asynchronous data communication exchange by using LCR. In addition, the programmer can retrieve, inspect, and modify the content of LCR; this eliminates the need for separate storage of the line characteristics in system memory.

**Figure 25-14. Line Control Register (LCR)**

31	Reserved								16								
R-0																	
15	Reserved							8	7	6	5	4	3	2	1	0	
R-0										DLAB	BC	SP	EPS	PEN	STB	WLS	
R-0										R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-13. Line Control Register (LCR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	DLAB	0	Divisor latch access bit. The divisor latch registers (DLL and DLH) can be accessed at dedicated addresses or at addresses shared by RBR, THR, and IER. Using the shared addresses requires toggling DLAB to change which registers are selected. If you use the dedicated addresses, you can keep DLAB = 0.
		0	Allows access to the receiver buffer register (RBR), the transmitter holding register (THR), and the interrupt enable register (IER) selected. At the address shared by RBR, THR, and DLL, the CPU can read from RBR and write to THR. At the address shared by IER and DLH, the CPU can read from and write to IER.
		1	Allows access to the divisor latches of the baud generator during a read or write operation (DLL and DLH). At the address shared by RBR, THR, and DLL, the CPU can read from and write to DLL. At the address shared by IER and DLH, the CPU can read from and write to DLH.
6	BC	0	Break control.
		0	Break condition is disabled.
		1	Break condition is transmitted to the receiving UART. A break condition is a condition where the UARTn_TXD signal is forced to the spacing (cleared) state.
5	SP	0	Stick parity. The SP bit works in conjunction with the EPS and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in <a href="#">Table 25-14</a> .
		0	Stick parity is disabled.
		1	Stick parity is enabled. <ul style="list-style-type: none"> <li>When odd parity is selected (EPS = 0), the PARITY bit is transmitted and checked as set.</li> <li>When even parity is selected (EPS = 1), the PARITY bit is transmitted and checked as cleared.</li> </ul>
4	EPS	0	Even parity select. Selects the parity when parity is enabled (PEN = 1). The EPS bit works in conjunction with the SP and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in <a href="#">Table 25-14</a> .
		0	Odd parity is selected (an odd number of logic 1s is transmitted or checked in the data and PARITY bits).
		1	Even parity is selected (an even number of logic 1s is transmitted or checked in the data and PARITY bits).
3	PEN	0	Parity enable. The PEN bit works in conjunction with the SP and EPS bits. The relationship between the SP, EPS, and PEN bits is summarized in <a href="#">Table 25-14</a> .
		0	No PARITY bit is transmitted or checked.
		1	Parity bit is generated in transmitted data and is checked in received data between the last data word bit and the first STOP bit.

**Table 25-13. Line Control Register (LCR) Field Descriptions (continued)**

Bit	Field	Value	Description
2	STB	0 1	<p>Number of STOP bits generated. STB specifies 1, 1.5, or 2 STOP bits in each transmitted character. When STB = 1, the WLS bit determines the number of STOP bits. The receiver clocks only the first STOP bit, regardless of the number of STOP bits selected. The number of STOP bits generated is summarized in <a href="#">Table 25-15</a>.</p> <p>1 STOP bit is generated.</p> <p>WLS bit determines the number of STOP bits:</p> <ul style="list-style-type: none"> <li>When WLS = 0, 1.5 STOP bits are generated.</li> <li>When WLS = 1h, 2h, or 3h, 2 STOP bits are generated.</li> </ul>
1-0	WLS	0-3h 0 1h 2h 3h	<p>Word length select. Number of bits in each transmitted or received serial character. When STB = 1, the WLS bit determines the number of STOP bits.</p> <p>5 bits</p> <p>6 bits</p> <p>7 bits</p> <p>8 bits</p>

**Table 25-14. Relationship Between ST, EPS, and PEN Bits in LCR**

ST Bit	EPS Bit	PEN Bit	Parity Option
x	x	0	Parity disabled: No PARITY bit is transmitted or checked
0	0	1	Odd parity selected: Odd number of logic 1s
0	1	1	Even parity selected: Even number of logic 1s
1	0	1	Stick parity selected with PARITY bit transmitted and checked as set
1	1	1	Stick parity selected with PARITY bit transmitted and checked as cleared

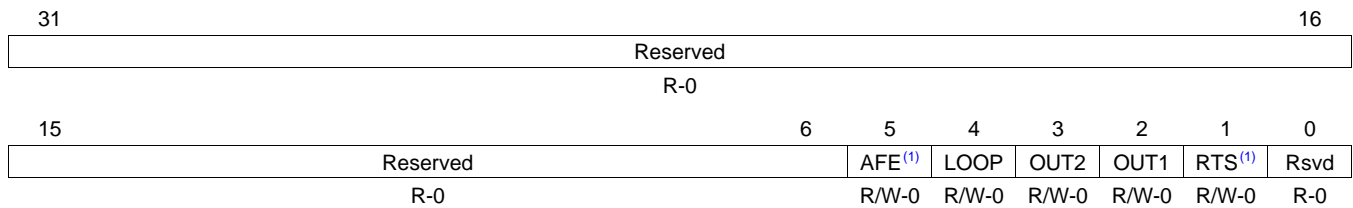
**Table 25-15. Number of STOP Bits Generated**

STB Bit	WLS Bits	Word Length Selected with WLS Bits	Number of STOP Bits Generated	Baud Clock (BCLK) Cycles
0	x	Any word length	1	16
1	0h	5 bits	1.5	24
1	1h	6 bits	2	32
1	2h	7 bits	2	32
1	3h	8 bits	2	32

### 25.3.7 Modem Control Register (MCR)

The modem control register (MCR) is shown in Figure 25-15 and described in Table 25-16. The modem control register provides the ability to enable/disable the autoflow functions, and enable/disable the loopback function for diagnostic purposes.

**Figure 25-15. Modem Control Register (MCR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

<sup>(1)</sup> All UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved and should be cleared to 0.

**Table 25-16. Modem Control Register (MCR) Field Descriptions**

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	AFE	0 1	Autoflow control enable. Autoflow control allows the $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ signals to provide handshaking between UARTs during data transfer. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved and should be cleared to 0.  0    Autoflow control is disabled. 1    Autoflow control is enabled: <ul style="list-style-type: none"> <li>When RTS = 0, <math>\overline{\text{UARTn\_CTS}}</math> is only enabled.</li> <li>When RTS = 1, <math>\overline{\text{UARTn\_RTS}}</math> and <math>\overline{\text{UARTn\_CTS}}</math> are enabled.</li> </ul>
4	LOOP	0 1	Loop back mode enable. LOOP is used for the diagnostic testing using the loop back feature.  0    Loop back mode is disabled. 1    Loop back mode is enabled. When LOOP is set, the following occur: <ul style="list-style-type: none"> <li>The <math>\overline{\text{UARTn\_TXD}}</math> signal is set high.</li> <li>The <math>\overline{\text{UARTn\_RXD}}</math> pin is disconnected</li> <li>The output of the transmitter shift register (TSR) is lopped back in to the receiver shift register (RSR) input.</li> </ul>
3	OUT2	0	OUT2 Control Bit
2	OUT1	0	OUT1 Control Bit
1	RTS	0 1	RTS control. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved and should be cleared to 0.  0 $\overline{\text{UARTn\_RTS}}$ is disabled, $\overline{\text{UARTn\_CTS}}$ is only enabled. 1 $\overline{\text{UARTn\_RTS}}$ and $\overline{\text{UARTn\_CTS}}$ are enabled.
0	Reserved	0	Reserved

### 25.3.8 Line Status Register (LSR)

The line status register (LSR) is shown in [Figure 25-16](#) and described in [Table 25-17](#). LSR provides information to the CPU concerning the status of data transfers. LSR is intended for read operations only; do not write to this register. Bits 1 through 4 record the error conditions that produce a receiver line status interrupt.

**Figure 25-16. Line Status Register (LSR)**

31	Reserved								16
R-0									
15	8	7	6	5	4	3	2	1	0
Reserved		RXFIFOE	TEMT	THRE	BI	FE	PE	OE	DR
R-0		R-0	R-1	R-1	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; -n = value after reset

**Table 25-17. Line Status Register (LSR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXFIFOE		Receiver FIFO error.
		0	There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver buffer register (RBR).
		1	There is a parity error, framing error, or break indicator in the receiver buffer register (RBR).
			<b>In FIFO mode:</b>
		0	There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver FIFO and there are no more errors in the receiver FIFO.
		1	At least one parity error, framing error, or break indicator in the receiver FIFO.
6	TEMT		Transmitter empty (TEMT) indicator.
			<b>In non-FIFO mode:</b>
		0	Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character.
		1	Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
			<b>In FIFO mode:</b>
		0	Either the transmitter FIFO or the transmitter shift register (TSR) contains a data character.
		1	Both the transmitter FIFO and the transmitter shift register (TSR) are empty.
5	THRE		Transmitter holding register empty (THRE) indicator. If the THRE bit is set and the corresponding interrupt enable bit is set (ETBEI = 1 in IER), an interrupt request is generated.
			<b>In non-FIFO mode:</b>
		0	Transmitter holding register (THR) is not empty. THR has been loaded by the CPU.
		1	Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR).
			<b>In FIFO mode:</b>
		0	Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. You can write to the transmitter FIFO if it is not full.
		1	Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).



**Table 25-17. Line Status Register (LSR) Field Descriptions (continued)**

Bit	Field	Value	Description
4	BI		Break indicator. The BI bit is set whenever the receive data input (UARTn_RXD) was held low for longer than a full-word transmission time. A full-word transmission time is defined as the total time to transmit the START, data, PARITY, and STOP bits. If the BI bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
		0	<b>In non-FIFO mode:</b> No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver buffer register (RBR).
		1	A break has been detected with the character in the receiver buffer register (RBR).
		0	<b>In FIFO mode:</b> No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver FIFO and the next character to be read from the FIFO has no break indicator.
		1	A break has been detected with the character at the top of the receiver FIFO.
3	FE		Framing error (FE) indicator. A framing error occurs when the received character does not have a valid STOP bit. In response to a framing error, the UART sets the FE bit and waits until the signal on the RX pin goes high. Once the RX signal goes high, the receiver is ready to detect a new START bit and receive new data. If the FE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
		0	<b>In non-FIFO mode:</b> No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR).
		1	A framing error has been detected with the character in the receiver buffer register (RBR).
		0	<b>In FIFO mode:</b> No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no framing error.
		1	A framing error has been detected with the character at the top of the receiver FIFO.
2	PE		Parity error (PE) indicator. A parity error occurs when the parity of the received character does not match the parity selected with the EPS bit in the line control register (LCR). If the PE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
		0	<b>In non-FIFO mode:</b> No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR).
		1	A parity error has been detected with the character in the receiver buffer register (RBR).
		0	<b>In FIFO mode:</b> No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no parity error.
		1	A parity error has been detected with the character at the top of the receiver FIFO.
1	OE		Overrun error (OE) indicator. An overrun error in the non-FIFO mode is different from an overrun error in the FIFO mode. If the OE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
		0	<b>In non-FIFO mode:</b> No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR).
		1	Overrun error has been detected. Before the character in the receiver buffer register (RBR) could be read, it was overwritten by the next character arriving in RBR.
		0	<b>In FIFO mode:</b> No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR).
		1	Overrun error has been detected. If data continues to fill the FIFO beyond the trigger level, an overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. An overrun error is indicated to the CPU as soon as it happens. The new character overwrites the character in the shift register, but it is not transferred to the FIFO.

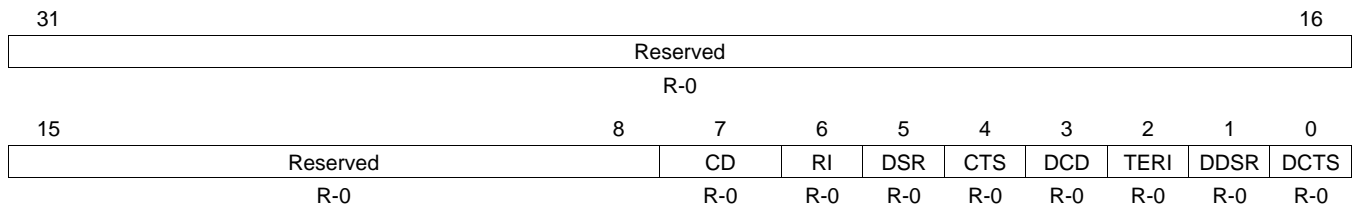
**Table 25-17. Line Status Register (LSR) Field Descriptions (continued)**

Bit	Field	Value	Description
0	DR		Data-ready (DR) indicator for the receiver. If the DR bit is set and the corresponding interrupt enable bit is set (ERBI = 1 in IER), an interrupt request is generated.
			<b>In non-FIFO mode:</b>
		0	Data is not ready, or the DR bit was cleared because the character was read from the receiver buffer register (RBR).
		1	Data is ready. A complete incoming character has been received and transferred into the receiver buffer register (RBR).
			<b>In FIFO mode:</b>
		0	Data is not ready, or the DR bit was cleared because all of the characters in the receiver FIFO have been read.
		1	Data is ready. There is at least one unread character in the receiver FIFO. If the FIFO is empty, the DR bit is set as soon as a complete incoming character has been received and transferred into the FIFO. The DR bit remains set until the FIFO is empty again.

### 25.3.9 Modem Status Register (MSR)

The Modem status register (MSR) is shown in [Figure 25-17](#) and described in [Table 25-18](#). MSR provides information to the CPU concerning the status of modem control signals. MSR is intended for read operations only; do not write to this register.

**Figure 25-17. Modem Status Register (MSR)**



LEGEND: R = Read only; -n = value after reset

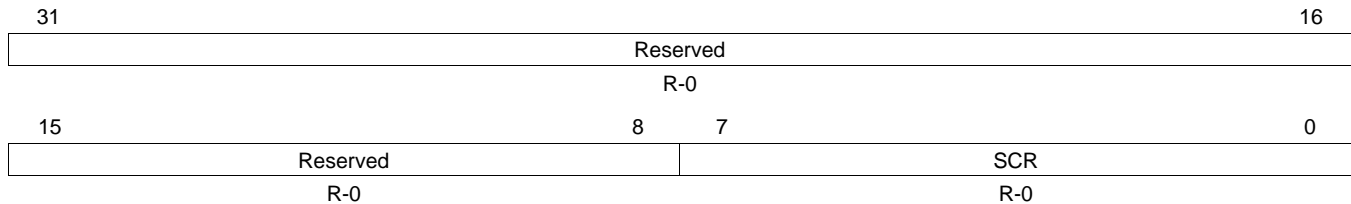
**Table 25-18. Modem Status Register (MSR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	CD	0	Complement of the Carrier Detect input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 3 (OUT2).
6	RI	0	Complement of the Ring Indicator input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 2 (OUT1).
5	DSR	0	Complement of the Data Set Ready input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 0 (DTR).
4	CTS	0	Complement of the Clear To Send input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 1 (RTS).
3	DCD	0	Change in DCD indicator bit. DCD indicates that the DCD input has changed state since the last time it was read by the CPU. When DCD is set and the modem status interrupt is enabled, a modem status interrupt is generated.
2	TERI	0	Trailing edge of RI (TERI) indicator bit. TERI indicates that the RI input has changed from a low to a high. When TERI is set and the modem status interrupt is enabled, a modem status interrupt is generated.
1	DDSR	0	Change in DSR indicator bit. DDSR indicates that the DSR input has changed state since the last time it was read by the CPU. When DDSR is set and the modem status interrupt is enabled, a modem status interrupt is generated.
0	DCTS	0	Change in CTS indicator bit. DCTS indicates that the CTS input has changed state since the last time it was read by the CPU. When DCTS is set (autoflow control is not enabled and the modem status interrupt is enabled), a modem status interrupt is generated. When autoflow control is enabled, no interrupt is generated.

### 25.3.10 Scratch Pad Register (SCR)

The Scratch Pad register (SCR) is shown in [Figure 25-18](#) and described in [Table 25-19](#). SCR is intended for programmer's use as a scratch pad. It temporarily holds the programmer's data without affecting UART operation.

**Figure 25-18. Scratch Pad Register (SCR)**



LEGEND: R = Read only; -n = value after reset

**Table 25-19. Scratch Pad Register (MSR) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SCR	0	These bits are intended for the programmer's use as a scratch pad in the sense that it temporarily holds the programmer's data without affecting any other UART operation.

### 25.3.11 Divisor Latches (DLL and DLH)

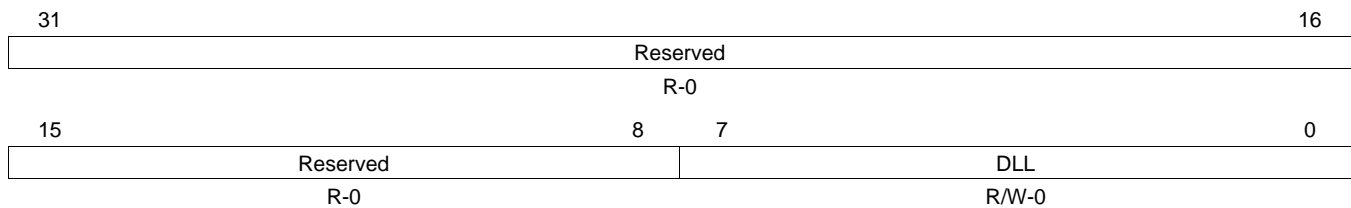
Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. The latches are in DLH and DLL. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

**Access considerations:**

- RBR, THR, and DLL share one address. When DLAB = 1 in LCR, all accesses at the shared address are accesses to DLL. When DLAB = 0, reading from the shared address gives the content of RBR, and writing to the shared address modifies THR.
- IER and DLH share one address. When DLAB = 1 in LCR, accesses to the shared address read or modify to DLH. When DLAB = 0, all accesses at the shared address read or modify IER.

DLL and DLH also have dedicated addresses. If you use the dedicated addresses, you can keep the DLAB bit cleared, so that RBR, THR, and IER are always selected at the shared addresses.

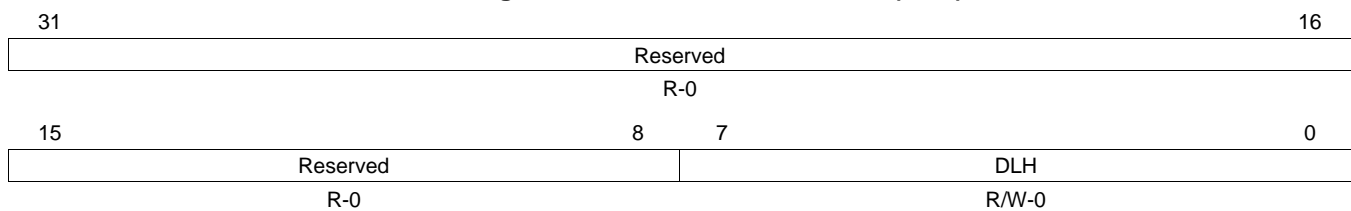
The divisor LSB latch (DLL) is shown in [Figure 25-19](#) and described in [Table 25-20](#). The divisor MSB latch (DLH) is shown in [Figure 25-20](#) and described in [Table 25-21](#).

**Figure 25-19. Divisor LSB Latch (DLL)**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-20. Divisor LSB Latch (DLL) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DLL	0-FFh	The 8 least-significant bits (LSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

**Figure 25-20. Divisor MSB Latch (DLH)**


LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

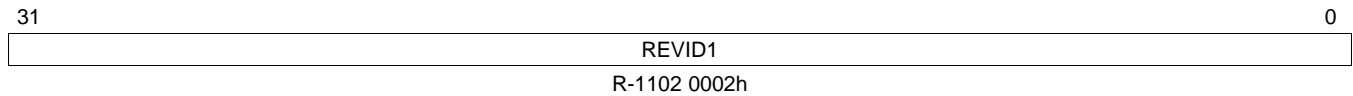
**Table 25-21. Divisor MSB Latch (DLH) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DLH	0-FFh	The 8 most-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

### 25.3.12 Revision Identification Registers (REVID1 and REVID2)

The revision identification registers (REVID1 and REVID2) contain peripheral identification data for the peripheral. REVID1 is shown in [Figure 25-21](#) and described in [Table 25-22](#). REVID2 is shown in [Figure 25-22](#) and described in [Table 25-23](#).

**Figure 25-21. Revision Identification Register 1 (REVID1)**

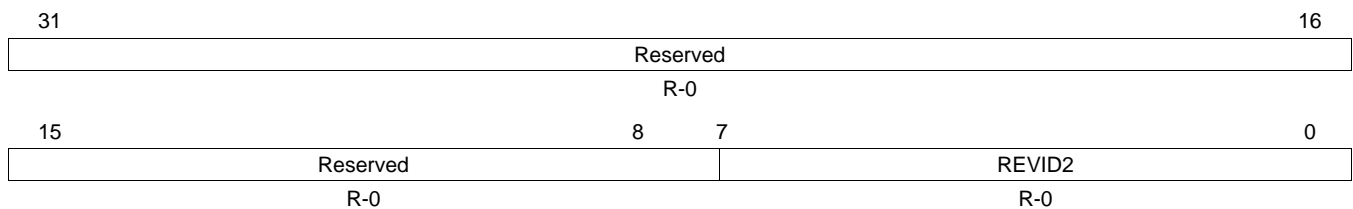


LEGEND: R = Read only; -n = value after reset

**Table 25-22. Revision Identification Register 1 (REVID1) Field Descriptions**

Bit	Field	Value	Description
31-0	REVID1	1102 0002h	Peripheral Identification Number

**Figure 25-22. Revision Identification Register 2 (REVID2)**



LEGEND: R = Read only; -n = value after reset

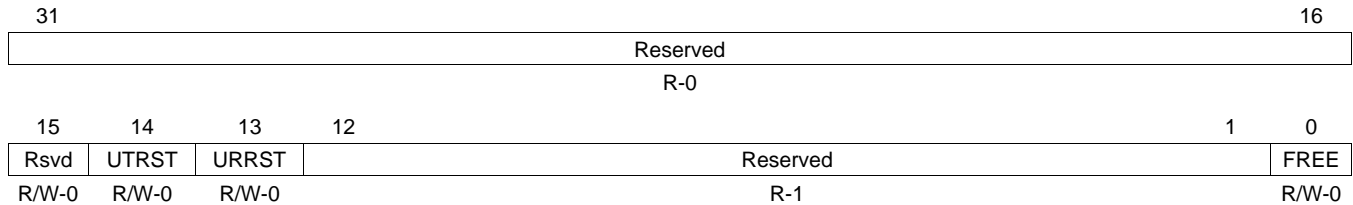
**Table 25-23. Revision Identification Register 2 (REVID2) Field Descriptions**

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	REVID2	0	Peripheral Identification Number

### 25.3.13 Power and Emulation Management Register (PWREMU\_MGMT)

The power and emulation management register (PWREMU\_MGMT) is shown in [Figure 25-23](#) and described in [Table 25-24](#).

**Figure 25-23. Power and Emulation Management Register (PWREMU\_MGMT)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

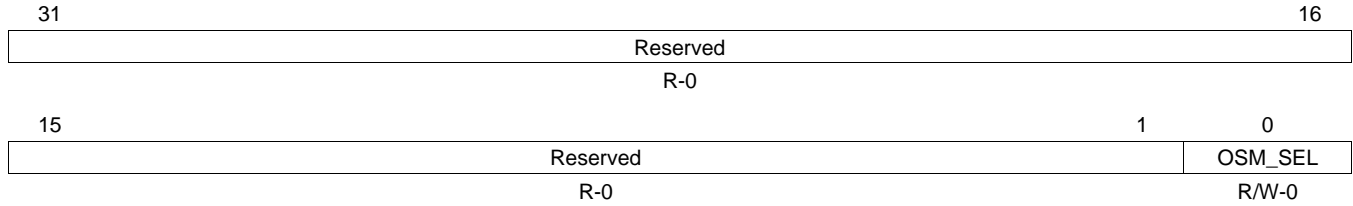
**Table 25-24. Power and Emulation Management Register (PWREMU\_MGMT) Field Descriptions**

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	Reserved	0	Reserved. This bit must always be written with a 0.
14	UTRST	0	UART transmitter reset. Resets and enables the transmitter. Transmitter is disabled and in reset state.
		1	Transmitter is enabled.
13	URRST	0	UART receiver reset. Resets and enables the receiver. Receiver is disabled and in reset state.
		1	Receiver is enabled.
12-1	Reserved	1	Reserved
0	FREE	0	Free-running enable mode bit. This bit determines the emulation mode functionality of the UART. When halted, the UART can handle register read/write requests, but does not generate any transmission/reception, interrupts or events. If a transmission is not in progress, the UART halts immediately. If a transmission is in progress, the UART halts after completion of the one-word transmission.
		1	Free-running mode is enabled; UART continues to run normally.

### 25.3.14 Mode Definition Register (MDR)

The Mode Definition register (MDR) determines the over-sampling mode for the UART. MDR is shown in [Figure 25-24](#) and described in [Table 25-25](#).

**Figure 25-24. Mode Definition Register (MDR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 25-25. Mode Definition Register (MDR) Field Descriptions**

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	OSM_SEL	0	Over-Sampling Mode Select. 16x over-sampling.
		1	13x over-sampling.



## Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

---

<b>Changes from July 10, 2016 to September 12, 2016 (from B Revision (July 2016) to C Revision)</b>	<b>Page</b>
---	-------------

---

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)