AM35x ARM Microprocessor

Technical Reference Manual

Literature Number: SPRUGR0C
October 2009–Revised November 2013

Contents

List of Figures

List of Tables

Read This First

Community Resources

The following link connects to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI Embedded Processors Wiki —Texas Instruments Embedded Processors Wiki

Established to assist developers using the many Embedded Processors from Texas Instruments to get started, help each other innovate, and foster the growth of general knowledge about the hardware and software surrounding these devices.

If You Need Assistance. . .

If you want to . . .	Do this . . .
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Call the CRC[1] hotline: **(800) 336−5236** Or write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251−1443
Order Texas Instruments documentation	Call the CRC[1] hotline: **(800) 336−5236**

(1) Texas Instruments Customer Response Center

About This Manual

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Information About Cautions and Warnings

This book may contain cautions and warnings.

CAUTION

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

WARNING

This is an example of a warning statement.

A warning statement describes a situation that could potentially cause harm to you.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

Register, Field, and Bit Calls

The naming convention applied for a call consists of:

- For a register call: *<Module name>.<Register name>*; for example: UART.UASR
- For a bit field call:
  - *<Module name>.<Register name>[End:Start] <Field name> field*; for example, UART.UASR[4:0] SPEED bit field
  - *<Field name> field <Module name>.<Register name>[End:Start]*; for example, SPEED bit field UART.UASR[4:0]
- For a bit call:
  - *<Module name>.<Register name>[pos] <Bit name> bit*; for example, UART.UASR[5] BIT_BY_CHAR bit
  - *<Bit name> bit <Module name>.<Register name>[pos]*; for example, BIT_BY_CHAR bit UART.UASR[5]

To help the reader navigate the document, each register call is hyperlinked to its register description in the register manual section. After each register description, a table summarizes all hyperlinked register calls.

To navigate in the PDF documents, see Acrobat Reader Tips.

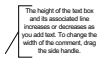
Coding Rules

The programming models or code listings follow the rules:

Type	Definition	Example
File	Starts with the module name	PRCM_test1.c MCBSP1_init.h
Variable	Global variables are prefixed by "g_" Pointers are prefixed by "p" Global pointers are prefixed by "g_p"	g_SDMA_LogicalChan pAddrCounter g_pSDMA_LogicalChan
Function	Starts with the module name	PRCM_SetupClocks() ArmIntC_MaskInterrupts()
Typedef	Ends with "_t"	PRCM_Struct_t
Definition	Starts with the module name and is followed by the register name	#define SMS_ERR_TYPE *((volatileUint32*)0x680080F4) #define MCBSP2_RCR1_REG *((volatile Uint32*)0x4807401C)
Enumeration	Starts with the module name	Typedef enum DMA_Mode_Label { INPUT_MODE OUTPUT_MODE } DMA_Mode_t;

Flow Chart Rules

Flow charts follow the following rules:

Shape	Name	Definition
	Process	Any computational steps or processing function of a program; defined operation(s) causing change in value, form, or location of information
	Decision	A decision or switching-type operation that determines which of a number of alternate paths is followed
	Predefined process or sub-process	One or more named operations or program steps specified in a subroutine or another set of flow charts
	Data or I/O	General I/O function; information available for processing (input) or recording of processed information (output)
	Terminator	Terminal point in a flow chart: start, stop, halt, delay, or interrupt; may show exit from a closed subroutine
	Annotation	Additional descriptive clarification, comment
	On page connector (reference)	Exit to, or entry from, another part of chart in the same page
	Off page connector (reference)	The flow continues on a different page.
	Summing Junction	Logical AND
	Or	Logical OR
	Parallel mode (ISO)	Beginning or end of two or more simultaneous operations
	Flow Line	Lines indicate the sequence of steps and the direction of flow.

Acrobat Reader Tips

Acrobat includes two methods to search for words in a PDF:

- The Find toolbar provides a basic set of options to locate a word in the current PDF.
- The Search window lista words or partial words that match your text in the current PDF.

These guidelines apply to Acrobat Reader 5.x, 6.0, and 7.0.

For more information on Acrobat Reader search features, see the Adobe Reader Help.

To search for words in a document using the Find dialog box:

1. Open the document.
2. To display the Find toolbar, right-click in the toolbar area and select Find.
3. In the Find box, type the word, words, or partial words for which you want to search.
4. From the Find Options menu, select options as desired.
5. To view each search result, click the Find toolbar, the Find Previous button, or the Find Next button to go backward or forward through the document.

To search for words in a document using the Search PDF window:

1. Open the document.
2. Click the Search button on the File toolbar or right-click on your document and select Search.
3. Type the word, words, or part of a word for which you want to search.
4. Click Search.
5. The results appear in page order and, if applicable, show a few words of context. Each result displays an icon to identify the type of occurrence. All other searchable areas display the Search Result icon.
6. To display the page that contains a search result, click an item in the Results list. The occurrence is highlighted.
7. To navigate to the next result, choose Edit > Search Results > Next Result (or Ctrl+G).
8. To navigate to the previous result, choose Edit > Search Results > Previous Result (or Shift+Ctrl+G).

Navigate through your previous view

To retrace your path within an Adobe PDF document:

- For the previous view: Choose View > Go To > Previous View or Alt+Left Arrow.
- For the next view: Choose View > Go To > Next View or Alt+Right Arrow. The Next View command is available only if you have chosen Previous View.

  If you view the PDF document in a browser, use options on the Navigation toolbar to move between views.

  - Right-click the toolbar area, and then choose *Navigation*.
  - Click the Go To Previous View button or the Go To Next View button.

NOTE: This navigation tip is useful to return to your previous view after clicking on a register call hyperlink.

OMAP3 Disclaimer

All programming models and use cases presented in this manual are provided for educative purposes only and may differ from or be optimized for your applications.

All OMAP peripheral devices presented in this manual are provided for illustration purposes and may be different from those in your system.

Trademarks

TMS320DMC64x, C64x, M-Shield and FlatLink3G are trademarks of Texas Instruments Incorporated.

ARM, JAZELLE, and THUMB are registered trademarks of ARM Limited.

ETM, ETB, ARM9, CoreSight, Cortex and Neon are trademarks of ARM Limited.

Bluetooth is a registered trademark of Bluetooth SIG, Inc. and is licensed to Texas Instruments.

Memory Stick is a registered trademark of Sony Corporation, and Memory Stick PRO is a trademark of Sony Corporation.

HDQ is a trademark of Benchmarq.

1-Wire is a registered trademark of Dallas Semiconductor.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

USSE and POWERVR are trademarks or registered trademarks of Imagination Technologies Ltd.

Mentor Graphics is a registered trademark of Mentor Graphics Corporation or its affiliated companies in the United States and other countries.

SonicsMX, Sonics3220 are trademarks or registered trademarks of Sonics, Inc.

Foveon X3 is a registered trademarks of Foveon, Inc.

Super CCD Honeycom is a registered trademark of Fuji Photo Film Co., Ltd.

Linux is a registered trademark of Linus Torvalds.

Symbian and all Symbian based trademarks and logos are trademarks of Symbian Software Limited.

Synopsys is a registered trademark of Synopsys, Inc.

MIPI is a registered trademark of the Mobile Industry Processor Interface (MIPI) Alliance.

OneNAND is a trademark of SAMSUNG.

All other trademarks are the property of their respective owners.

History

The following table summarizes the AM35x TRM versions.

Version	Literature Number	Date	Notes
*	SPRUGR0	October 2009	See (1)
A	SPRUGR0	June 2010	See (2)
B	SPRUGR0	July 2010	See (3)
C	SPRUGR0	November 2013	See (4)

(1) AM35x ARM Microprocessor Technical Reference Manual - * version (SPRUGR0) initial release.

(2) AM35x ARM Microprocessor Technical Reference Manual - version A (SPRUGR0).
- Chapter 7: DMA
- Chapter 12: Display Subsystem
- Chapter 14: UART/IrDA/CIR
- Chapter 16: Multichannel SPI
- Chapter 17: Multichannel Buffered Serial Port

(3) AM35x ARM Microprocessor Technical Reference Manual - version B (SPRUGR0B).
- Chapter 8: Interrupt Controller
- Chapter 14: UART/IrDA/CIR

(4) AM35x ARM Microprocessor Technical Reference Manual - version C (SPRUGR0C).
- Chapter 1: Introduction
- Chapter 4: Power, Reset, and Clock Management
- Chapter 18: Multichannel Buffered Serial Port
- Chapter 20: Universal Serial Bus (USB)

Introduction

This chapter introduces the features, supporting subsystems, and architecture of the AM35x ARM Microprocessors.

Topic **Page**

1.1 Overview

The AM35x ARM Microprocessors are integrated on TI's advanced 65-nm process technology.

NOTE: This technical reference manual describes all available features. Some features may not be available or supported on your particular device. For more information, see your device-specific data manual and Section 1.4, *Device Family*.

These devices are designed to provide maximum flexibility for ARM based applications including, but not limited to, Industrial Automation/Control, Single Board Computers, and Human Machine Interface.

These devices also feature M-Shield™ mobile security technology to enable secure e-commerce applications and the replay of copyright-protected digital media content.

Security features integrated on these devices support applications designed for:

- Protection against malicious attacks
- M-commerce
- Content protection for recordable media (CPRM)
- Digital rights management (DRM)

High-security (HS) devices rely on a security scheme based on hardware mechanisms and secure read-only memory (ROM) code, ensuring that only trusted code can access secure resources. These resources are in specific regions of memory as well as in peripherals, hardware cryptographic accelerators, and eFuse keys.

NOTE: To determine if a high-security (HS) version of your device is available and for more information on HS devices, see your device-specific data manual and Section 1.4, *Device Family*.

The device supports high-level operating systems such as:

- Windows CE
- Linux

Multiple RTOS are also supported.

These devices also include state-of-the-art power-management techniques required for high-performance mobile products.

The following subsystems are part of the device:

- Microprocessor unit (MPU) subsystem based on the ARM® Cortex™-A8 microprocessor
- SGX subsystem for 3D graphics acceleration to support display and gaming effects

NOTE: SGX is not available on all devices. For more information see your device-specific data manual and Section 1.4, *Device Family*.

- Camera image signal processor (VPFE) that supports multiple formats and interfacing options connected to a wide variety of image sensors
- Display subsystem with a wide variety of features for multiple concurrent image manipulation, and a programmable interface supporting a wide variety of displays. The display subsystem also supports NTSC/PAL video out.
- Level 3 (L3) and level 4 (L4) interconnects that provide high-bandwidth data transfers for multiple initiators to the internal and external memory controllers and to on-chip peripherals

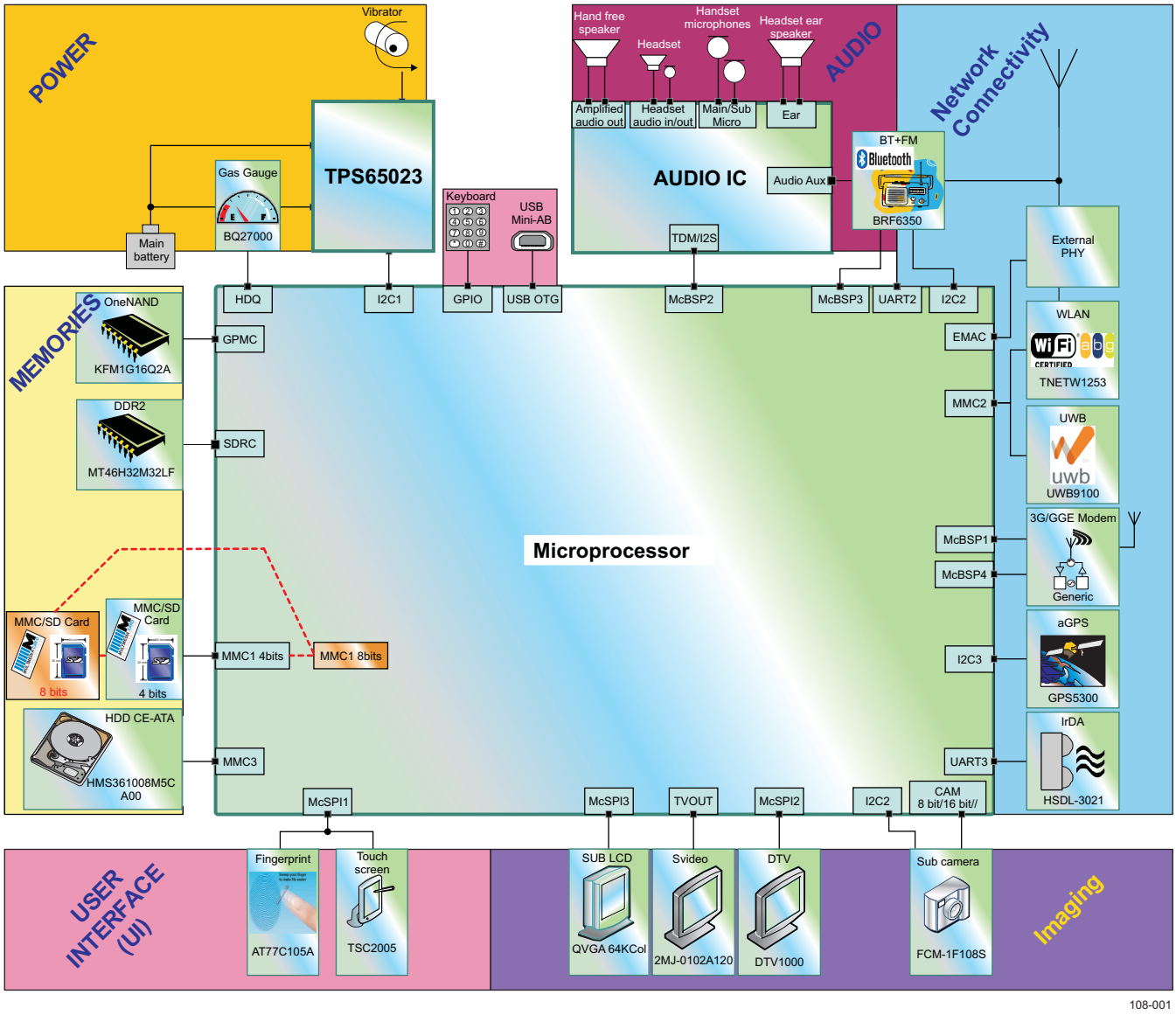
These devices also offer a comprehensive power and clock-management scheme that enables high-performance and low-power operation.

1.2 Environment

This section provides an overview of the device when integrated with the TPS65023 power integrated circuit (IC). For more information on the TPS65023 device, contact your TI representative.

Figure 1-1 provides an overview of a nonexhaustive environment for the high-tier device.

Figure 1-1. Environment Using TPS65023



108-001

NOTE: Some features are not available on all devices. For more information. see your device-specific data manual and Section 1.4, *Device Family*.

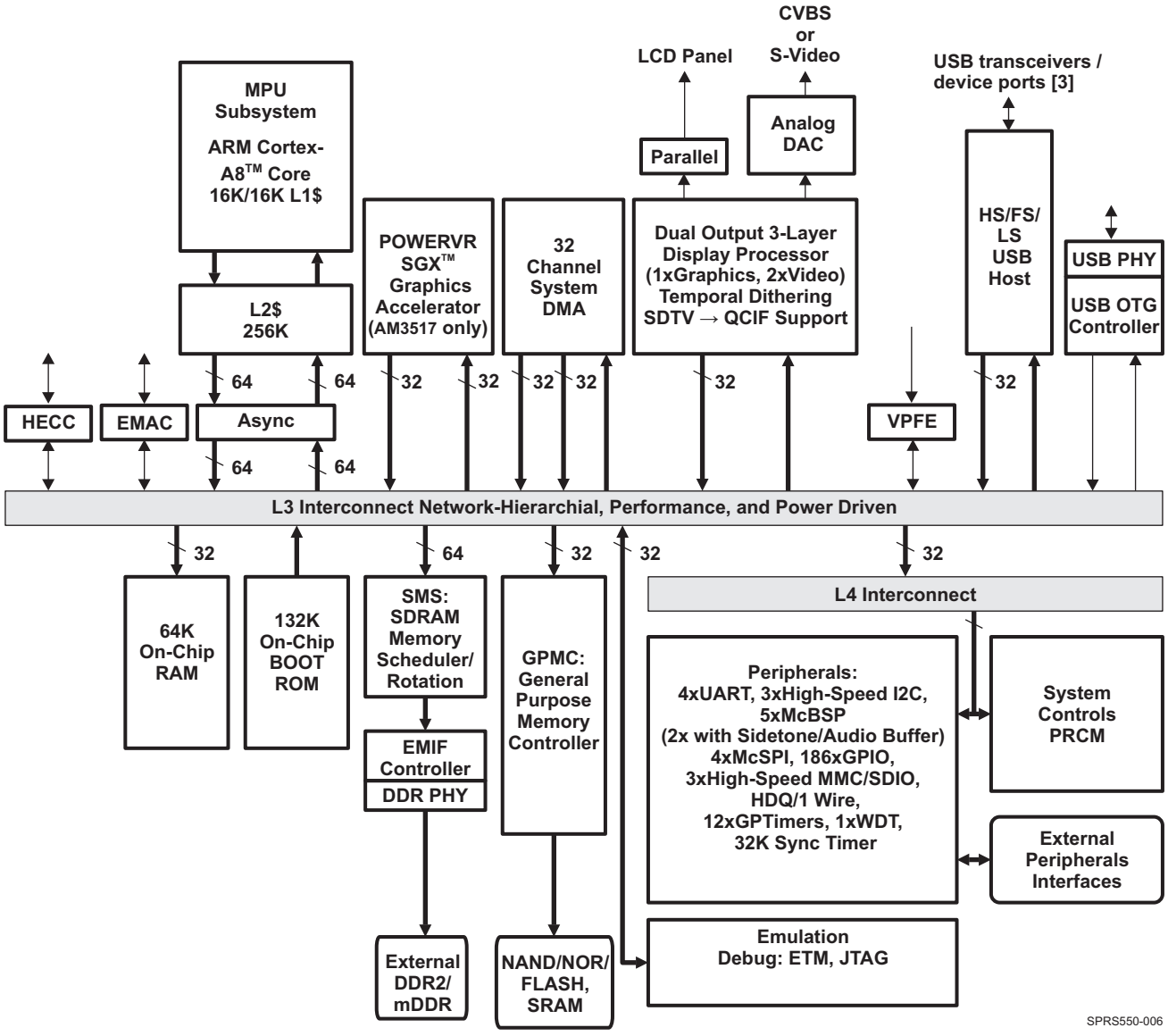
1.3 Description

These devices are offered in the following package:

- ZCN package: 491-pin BGA (17x17, 0.65mm pitch)
- ZER package: 484-pin PBGA (23x23, 1mm pitch)

Figure 1-2 shows the block diagram.

Figure 1-2. Block Diagram



NOTE: Some features are not available on all devices. For more information, see your device-specific data manual and Section 1.4, *Device Family*.

1.3.1 MPU Subsystem

The MPU subsystem integrates the following modules

- ARM subchip
  - ARM® Cortex™-A8 core
  - ARM Version 7™ ISA: Standard ARM instruction set + Thumb®-2, Jazelle® RCT Java accelerator, and media extensions
  - NEON™ SIMD coprocessor (VFP lite + media streaming instructions)
  - Cache memories
    - Level 1: 16KB instruction and 16KB data—4-way set associative cache, 64 bytes/line
    - Level 2: see Section 1.4, *Device Family*
- Interrupt controller (MPU INTC) of 96 synchronous interrupt lines
- Asynchronous interface with core logic
- Debug, trace, and emulation features: ICE-Crusher, ETM, ETB modules

1.3.2 On-Chip Memory

On-chip memory configuration offers memory resources for program and data storage:

- 112KB ROM
- 64KB single-access static random access memory (SRAM)

1.3.3 External Memory Interfaces

The device includes two external memory interfaces:

- General-purpose memory controller (GPMC)
  - NOR flash, NAND flash (with ECC Hamming code calculation), SRAM and Pseudo-SRAM asynchronous and synchronous protocols
  - Flexible asynchronous protocol control for external ASIC or peripheral interfacing
  - 16-bit data, up to 8 chip-selects (CSs)
  - 128M-byte addressable per chip-select, 1G-byte total address space
  - Nonmultiplexed device with limited address (2K bytes)
- SDRAM Controller (SDRC)
  - Double data rate (DDR2 and LPDDR) SDRAM
  - 16-bit or 32-bit data, 2 chip-selects, configurations for a maximum of 1 G-byte address space per chip-select
  - Work in conjunction with the SDRAM memory scheduler (SMS) companion module

1.3.4 DMA Controllers

The device embeds one generic DMA controller, the system DMA (sDMA) controller, used for memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers:

- One read port, one write port
- 32 prioritizable logical channels
- 96 hardware requests
- 256 x 32-bit FIFO dynamically allocable between active channels

The device also embeds two dedicated DMA controllers: display DMA and USB HS DMA.

1.3.5 Multimedia Accelerators

The device uses the following multimedia accelerators for display and gaming effects as well as high-end imaging and video applications:

- 3D graphics accelerator (SGX)
  - 3D graphics and video codecs supported on common hardware
  - Tile-based architecture
  - Universal scalable shader engine (USSE™) multithreaded engine incorporating pixel and vertex shader functionality reducing die area
  - Advanced shader feature set in excess of Microsoft VS3.0, PS3.0, and OGL2.0
  - Industry standard API supports OGL-ES 1.1 and 2.0, OpenVG 1.0, and OpenMax
  - Fine-grained task switching, load balancing, and power management
  - Programmable high-quality image anti-aliasing
  - Advanced geometry DMA driven operation for minimum CPU interaction
  - Fully virtualized memory addressing for OS operation in a unified memory architecture
  - Advanced and standard 2D operations (that is, vector graphics, BLTs, ROPs, etc.)
- Video Processing Front End (VPFE)
  - Supports most of the raw and smart image sensors available in the market
  - 16-bit parallel interface supported
  - Pixel clock up to 75 MHz

CAUTION

Clock configurations depend on the core voltage and maximum clock frequencies. Values in this document might not apply to production devices. Refer to your device-specific data manual for supported values for production devices.

- Display interface
  - Display controller
  - Color and monochrome displays up to 2048 x 2048 x 24-bpp resolution
  - 256 x 24-bit entries palette in red, green, blue (RGB)
  - 3,375 colors, 15 grayscales
  - Picture-in-picture (overlay), color-space conversion, rotation, color-phase rotation, and resizing support
  - Remote frame buffer interface
  - Liquid-crystal display (LCD) pixel interfaces (MIPI DPI 1.0) and LCD bus interfaces (MIPI DBI 1.0) supported
  - NTSC/PAL video encoder outputs with integrated digital-to-analog converters (DACs) output are supported on CVBS and S-video TV analog output signals
  - Embedded DMA controller

1.3.6 Security (HS Devices Only)

The secure firmware resides in a secure version of the ROM and includes hardware security features that enable HS devices and encryption/decryption accelerators. The following encryption/decryption accelerators are only available in HS devices:

- RNG
- 2 x DES/3DES
- SHA1/MD5
- SHA2/MD5
- 2 x AES with counter mode
- Fast PKA

The customer programmable fuse ROM (CPFROM) module is only available on high-security (HS) devices.

NOTE: To determine if a high-security (HS) version of your device is available and for more information on HS devices, see your device-specific data manual and Section 1.4, *Device Family*.

1.3.7 Comprehensive Power Management

The device includes the following power management features:

- Clock and reset generation and distribution
- Auto clock gating to save active power by gating clock when peripheral is not active

1.3.8 Peripherals

The device supports a comprehensive set of peripherals to provide flexible and high-speed interfacing and on-chip programming resources. Table 1-1 provides a list and description of the peripherals that are available.

Table 1-1. Device Peripherals

Type	Name	Number	Description
Serial Communication	Multi-channel Buffered Serial Ports (McBSPs)	5	The McBSPs provide a full-duplex direct serial interface between the device and other devices in a system such as audio and voice codecs and other application chips. McBSP1, McBSP2, and McBSP3 serve as general purpose serial ports while McBSP2 and McBSP3 include additional audio-loopback capability.
	Multi-channel Serial Port Interface (McSPI)	4	The McSPIs provide a master/slave interface to SPI devices.
	High-speed Multi-port USB Host Controller	1	High-speed USB2.0 host controller with three host ports each offering high-speed data transactions (up to 480 Mbps) or full-speed/low-speed data transactions (12 and 1.5 Mbps, respectively). In high-speed mode, the USB host controller ports interface to external USB PHYs using a 12-pin or 8-pin UTMI low pin interface (ULPI). In full-speed and low-speed mode, the ports interface to external USB PHYs using a 6-/4-/3-pin serial interface.
	High-speed USB OTG Controller	1	High-speed USB2.0 OTG controller that offers high-speed data transactions (up to 480 Mbps) on a USB port with embedded DMA controller. The high speed USB OTG controller includes integrated PHY, thus eliminating need of external PHY.
	HDQ/1-Wire®	1	The HDQ/1-Wire interface supports the Benchmark HDQ protocol and the Dallas Semiconductor 1-Wire protocol.
	Universal Asynchronous Receiver/Transmitter (UART)	4	Serial communication interfaces compatible to the industry standard TL16C550 asynchronous communications element. UART1 and UART 2 are general serial communication interfaces. UART3 provides additional support for infrared data association (IrDA) and consumer infrared (CIR) communications.
	High-speed (HS) Inter-integrated Circuit (I2C) Controllers	3	Master/slave I2C high-speed standard interfaces with support for standard mode (up to 100K bits/s), fast mode (up to 400K bits/s), and high-speed mode (up to 3.4M bits/s).
	HECC	1	Interface used to connect to vehicle bus in automotive.
Removable Media	Multimedia Card/Secure Digital/Secure Digital I/O (MMC/SDIO) Card Interface	3	MMC memory card, SD memory card, or SDIO cards interface.
Miscellaneous	GP timers	12	Twelve general-purpose timers
	Watchdog timers (WDTs)	1	Three watchdog timers
	32-kHz synchronization timer	1	32-kHz clock timer
	General-purpose input/output (GPIO)	Package-specific	General-purpose input/output pins controlled by six GPIO controllers.
	EMAC	1	10/100 Mbit Ethernet MAC operation with CPPI4.0 compliant & RMII interface. Interface used to connect to vehicle
	Control module	1	I/O multiplexing and chip-configuration control.
Security Modules (High-security Devices Only)			RNG, Fast PKA, 2xDES/3DES, SHA1/MD5, SHA2/MD5, 2xAES, and Secure Watchdog Timer.

1.4 Device Family

1.4.1 Device Features

Devices are configured with different sets of features on different devices. This technical reference manual details all available features. Some features may not be available or supported in your particular device. Features supported across different devices are shown in Table 1-2 (ZCN package). For more information on the ZCN package, refer to your device-specific data manual.

Table 1-2. Subsystem, Co-Processor, and Peripheral Support Matrixon the (ZCN Package)

Subsystem/Co-Processor/Peripheral	Chapter	AM3517	AM3505
POWERVR SGX™ 3D Graphics Accelerator	11	x	
Cortex-A8 Neon Co-Processor	3	x	x
SDRAM Controller	9	x	x
General-Purpose Memory Controller	9	x	x
VPFE	10	x	x
Display Subsystem	12	x	x
LCD DPI, LCD RFBI and TV Output Interface		x	x
LCD DSI and LCD SDI			
McBSP1/2/3/4/5	17	x	x
McSPI1/2/3/4	16	x	x
High-Speed USB OTG Controller	20	x	x
High-Speed USB Host Controller	20	x	x
HDQ/1-Wire	18	x	x
UART1/2	14	x	x
UART3/IrDA/CIR	14	x	x
I2C1/2/3	15	x	x
EMAC	22	x	x
HECC	23	x	x
MMC/SD/SDIO1/2/3	19	x	x
GP Timer (x12)	13	x	x
Watchdog Timer	13	x	x
32-kHz Sync Timer	13	x	x
GPIO	21	x	x
Secure ROM	1		
RNG	1		
DES/3DES	1		
SHA1/MD5	1		
SHA2/MD5	1		
AES	1		
Fast PKA	1		
Secure Watchdog Timer	1		
High-security Device	1		
General-purpose Device	1	x	x

1.4.2 Device Identification

The identification registers include the CONTROL_IDCODE and CONTROL_DIE_ID data registers. These registers are accessible through the L4 interconnect port starting at physical address 0x4830 A204 and 0x4830 A218, respectively. See the Memory Mapping chapter for more information about the L4 memory space mapping. Table 1-3 describes the identification registers.

The silicon type can be read in the HAWKEYE bit field value of the CONTROL.CONTROL_IDCODE register. The silicon revision can be read in the VERSION bit field value of the CONTROL.CONTROL_IDCODE register.

Table 1-3. Device Identification Registers

Register Name	Address	Size
CONTROL.CONTROL_IDCODE[31:0]	0x4830 A204	32
CONTROL.CONTROL_DIE_ID[127:0]	0x4830 A218	128

To retrieve chip identification, see Table 1-4. This register helps software identify the chip derivative.

Table 1-4. Chip Identification

Scalable Resource	Name	Bit	Value	Description	AM3517	AM3505
SGX	SGX_scalable_control	14:13	00	Full use.	00	
			01	Core clock restricted in HW to /6 from L3.		
			10	HW not present.		10
			11	Reserved.		
Reserved	Reserved	12	0	Reserved. Not available for use.	0	0
MPU L2 Cache Size	MPU_L2_cache_size	11:10	00	0 KB.		
			01	64 KB.		
			10	128 KB.		
			11	Full use (256 KB).	11	11
MPU Frequency	ARM_MHz	9:8	00	500 MHz.	00	00
			01	400 MHz.		
			10	266 MHz.		
			11	Reserved.		
Reserved	Reserved	7:5	1	Reserved. Not available for use.	0	0
NEON & VFP	NEON_VFPLite	4	0	Full use.	0	0
			1	Not available for use.		
Reserved	Reserved	3:2	1	Reserved. Not available for use.	0	0
MMC1 Width	4_8_bit_mmc	1	0	Full use (8-bit width at 3.0v IO)	0	0
			1	Restricted use (4-bit width at 3.0v IO)		
TV Out	TO_OUT	0	0	Full use.	0	0
			1	Not available for use.		
Control Device Status Register 15:0 (Address 0x4800 244C)					0x0C00	0x4C00

Table 1-5. CONTROL_IDCODE Register Definition

Field	Bits	Value	Comment
CONTROL.CONTROL_IDCODE [31:28]	VERSION	See Table 1-6	Revision number
CONTROL.CONTROL_IDCODE [27:12]	HAWKEYE	See Table 1-7	Hawkeye number
CONTROL.CONTROL_IDCODE [11:1]	TI_IDM	0x13	Manufacturer identity (TI)
CONTROL.CONTROL_IDCODE [0]	--	0x1	Always set to 1.

The Hawkeye number is hardcoded in the design. Table 1-7 lists the Hawkeye number values, and Table 1-6 lists the silicon revision values.

Table 1-6. Revision Number Value

Silicon Type	Field	Value
ES1.0	CONTROL.CONTROL_IDCODE[31:28]	0000
ES1.1	CONTROL.CONTROL_IDCODE[31:28]	0001

Table 1-7. Hawkeye Number Value

Silicon Type	Field	Value
ES1.0	CONTROL.CONTROL_IDCODE[27:12]	0xB868
ES1.1	CONTROL.CONTROL_IDCODE[27:12]	0xB868

- The CONTROL.CONTROL_DIE_ID register is the 128 bits single identifier of the device.

Table 1-8. CONTROL_DIE_ID

Field	Bits	Value
DIE_ID[127:0]	RESERVED	Single identifier

1.4.3 General Recommendations Relative to Unavailable Features/Modules

As explained in the previous section, some features are not available in all devices. For unavailable features, use the following recommendations:

- Memory mapping: Memory area of unavailable modules and features are RESERVED, read is undefined, and write can lead to unpredictable behavior.
- Interrupt controllers: Ensure that interrupts of unavailable modules and features are masked in the MPU subsystem.
- DMA: Ensure that DMA requests of unavailable modules and features are masked in DMA subsystems.
- System Control Module (SCM): Unavailable modules and feature pins are not functional and should not be used.
- Power, Reset, and Clock Management Module (PRCM): For power management and power-saving consideration, ensure that power domains of unavailable features/modules are switched off and clocks are cut off.
- Interconnect: To flag potential interconnect outstanding commands, the time-out of target agents attached to unavailable modules can be enabled with the lowest setting.

Memory Mapping

This chapter describes memory mapping.

NOTE: This chapter gives information about all modules and features in the high-tier device. To check availability of modules and features, see Chapter 1, *Device Family*. The memory area of unavailable modules and features is RESERVED, read is undefined, and write can lead to unpredictable behavior.

Topic **Page**

2.1 Introduction

The microprocessor unit (MPU) has a 32-bit address port, allowing it to handle a 4Gbytes space divided into several regions, depending on the target type.

The memory map is composed of a memory space (general-purpose memory controller [GPMC], external memory interface [EMIF4], etc.), register space (L3 and L4 interconnects), and dedicated spaces (SGX, etc.), all of which are shared among the initiators (for example, the MPU subsystem or the display subsystem).

The GPMC and EMIF4 are dedicated to memory connection. The GPMC is used for NOR/NAND flash and PSRAM memories. The EMIF4 is used for DDR2. For more information, see the *Memory Subsystem* chapter.

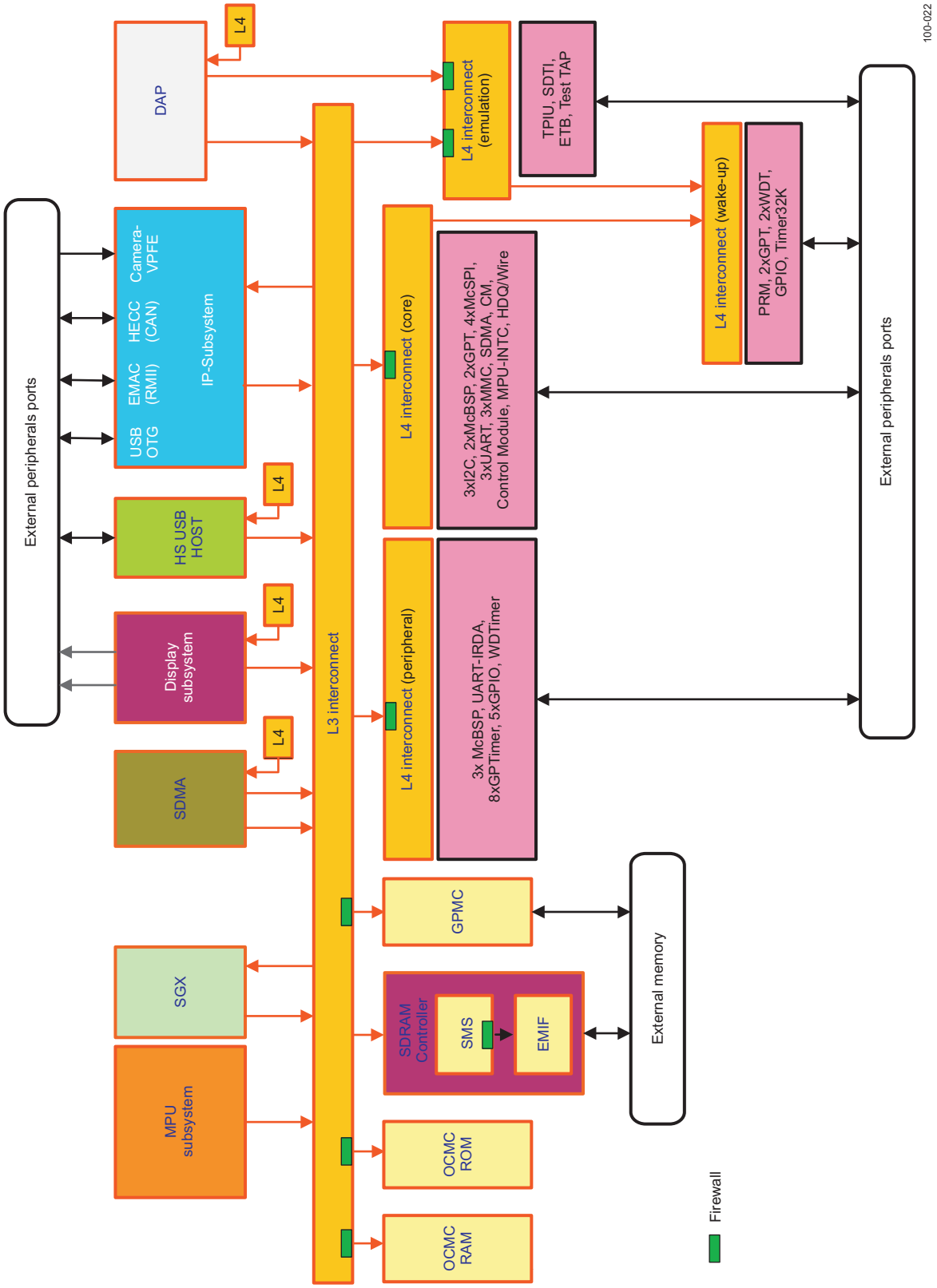
The L3 interconnect allows the sharing of resources, such as peripherals and external or on-chip memories, between all the initiators of the platform. The L4 interconnects control access to the peripherals.

Transfers between initiators and targets across the platform are physically conditioned by the chip interconnect and can be logically conditioned by firewalls. For more information about the intercommunication (L3 and L4 interconnects) and protection mechanisms implemented in the device, see the *Interconnect*.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, *Device Family*, and your device-specific data manual.

Figure 2-1 shows the interconnect of the device and the main modules and subsystems in the platform.

Figure 2-1. Interconnect Overview



2.2 Global Memory Space Mapping

This section provides a global view of the memory mapping and details the boot, GPMC, EMIF4, and virtual rotated frame buffer (VRFB) memory spaces.

The system memory mapping is flexible, with two levels of granularity for target address space allocation:

- Level 1: Four quarters are labeled Q0, Q1, Q2, and Q3. Each quarter corresponds to a 1Gbyte address space (total address space is 4Gbytes).
- Level 2: Each quarter is divided into eight blocks of 128Mbytes, with target spaces mapped inside the blocks.

This organization allows all target spaces to be decoded based on the five most significant bits (MSBs) of the 32-bit address ([31:27]).

- **Boot space**

  The system has a 1Mbyte boot space either in the on-chip boot ROM or on the GPMC memory space.

  When booting from the on-chip ROM with the appropriate external sys_boot5 pin configuration, the 1Mbyte memory space is redirected to the on-chip boot ROM memory address space [0x4000 0000 – 0x400F FFFF].

  When booting from the GPMC with the appropriate external sys_boot5 pin configuration, the memory space is part of the GPMC memory space.

  For more information on sys_boot5 pin configuration, see the *Memory Subsystem* and *Initialization* chapters.

- **GPMC space**

  Eight independent GPMC chip-selects (gpmc_ncs0 to gpmc_ncs7) are available in the first quarter (Q0) of the addressing space to access NOR/NAND flash and PSRAM memories. The chip-selects have a programmable start address and programmable size (16Mbytes, 32Mbytes, 64Mbytes, or 128Mbytes) in a total memory space of 1Gbyte.

- **EMIF4 space**

  Two EMIF4 chip-selects (emif4_ncs0 and emif4_ncs1) are available on the third quarter (Q2) of the addressing space to access SDRAM memories. The base address of EMIF4 space starts at 0x8000 0000. For more information on EMIF please refer to MemorySubsystem Chapter.

- **VRFB space**

  The EMIF4-SMS virtual memory space is a different memory space used to access a subset of the EMIF4 memory space through the rotation engine. The virtual address space size is 768Mbytes split into two parts: The first 256M-byte part is in the second quarter (Q1) of the memory; the second 512Mbytes part is in the fourth quarter (Q3) of the memory.

For more information on GPMC, EMIF4, and VRFB, see the *Memory Subsystem* chapter.

This section gives information about all modules and features in the high-tier device. To check availability of modules and features, see Chapter 1, *Device Family*. The memory area of unavailable modules and features is RESERVED, read is undefined, and write can lead to unpredictable behavior.

Table 2-1 describes the global memory space mapping.

Table 2-1. Global Memory Space Mapping

QUARTER	Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
Q0 (1GB)	**Boot space[1] GPMC**			**1MB 1GB or 1GB-1MB**	
	GPMC	0x0000 0000	0x3FFF FFFF	1GB	8/16 Ex[2]/R/W
Q1 (1GB)	**On-Chip Memory**			**128MB**	**ROM/SRAM address space**
	Boot ROM internal[1]	0x4000 0000	0x4001 3FFF	80KB	32-bit Ex[2]/R – Secure
		0x4001 4000	0x4001 BFFF	32KB	32-bit Ex[2]/R – Public
	Reserved	0x4001 C000	0x400F FFFF	912KB	Reserved
	Reserved	0x4010 0000	0x401F FFFF	1MB	Reserved
	SRAM internal	0x4020 0000	0x4020 FFFF	64KB	32-bit Ex[2]/R/W – Secure/public[3]
	Reserved	0x4021 0000	0x4024 FFFF	256KB	Reserved
	Reserved	0x4025 0000	0x47FF FFFF	128,704KB	Reserved
	L4 interconnects			**128MB**	**All system peripherals**
	L4-Core	0x4800 0000	0x48FF FFFF	16MB	See Table 2-3.
	(L4-Wakeup)[4]	(0x4830 0000)	(0x4833 FFFF)	(256KB)	(See Table 2-4.)
	L4-Per	0x4900 0000	0x490F FFFF	1MB	See Table 2-5.
	Reserved	0x4910 0000	0x4FFF FFFF	111MB	Reserved
	SGX			**64MB**	**Graphic accelerator slave port**
	SGX	0x5000 0000	0x5000 FFFF	64KB	Graphic accelerator slave port
	Reserved	0x5001 0000	0x53FF FFFF	65,472KB	Reserved
	L4 Emulation			**64MB**	**Emulation**
	L4-Emu	0x5400 0000	0x547F FFFF	8MB	See Table 2-6.
	Reserved	0x5480 0000	0x57FF FFFF	56MB	Reserved
	Reserved			**64MB**	**Reserved**
	Reserved	0x5800 0000	0x5BFF 0FFF	64MB	Reserved
	IPSS			**64MB**	**IPSS**
	IPSS	0x5C00 0000	0x5EFF FFFF	48MB	IPSS. See Table 2-8.
	Reserved	0x5F00 0000	0x5FFF FFFF	16MB	Reserved
	Reserved			**128MB**	**Reserved**
	Reserved	0x6000 0000	0x67FF FFFF	128MB	Reserved
	L3 Interconnect			**128MB**	**Control Registers**
	L3 Control Registers	0x6800 0000	0x68FF FFFF	16MB	See Table 2-2.
	Reserved	0x6900 0000	0x6BFF FFFF	48MB	Reserved
	SMS registers	0x6C00 0000	0x6CFF FFFF	16MB	Configuration registers SMS address space 2
	EMIF4 registers	0x6D00 0000	0x6DFF FFFF	16MB	Configuration registers SMS address space 3
	GPMC registers	0x6E00 0000	0x6EFF FFFF	16MB	Configuration registers GPMC address space 1
	Reserved	0x6F00 0000	0x6FFF FFFF	16MB	Reserved

(1) Boot space location depends on the external sys_boot5 pin configuration.
(2) Executable
(3) Default public/secure settings after reset only
(4) Peripherals connected to the L4-Wakeup interconnect are accessed through the L4-Core interconnect.

Table 2-1. Global Memory Space Mapping (continued)

QUARTER	Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
	EMIF4/SMS			**256MB**	**EMIF4/SMS**
	EMIF4/SMS virtual Address space 0	0x7000 0000	0x7FFF FFFF	256MB	EMIF4-SMS virtual address space 0
Q2 (1GB)	**EMIF4/SMS**			**1GB**	**SDRAM main address space (SMS)**
	SDRAM	0x8000 0000	0xBFFF FFFF	1GB	EMIF4/SMS
Q3 (1GB)	**Reserved**			**512MB**	**Reserved**
	Reserved	0xC000 0000	0xDFFF FFFF	512MB	Reserved for future use.
	EMIF4/SMS			**512MB**	**EMIF4/SMS**
	EMIF4/SMS virtual Address space 1	0xE000 0000	0xFFFF FFFF	512MB	EMIF4-SMS virtual address space 1

2.3 L3 and L4 Memory Space Mapping

The memory space system is defined from a hierarchical view: L1, L2, L3, and L4.

L1 memory includes the MPU subsystem.

The chip-level interconnect, which is made of one L3 and four L4s, enables communication between all modules and subsystems.

L3 handles many types of data transfers and, in particular, the data exchange with system on-chip/external memories.

The four L4s that handle transfers with peripherals are: the L4-Core, L4-Wakeup, L4-Per, and L4-Emu interconnects.

For more information about the interconnect, see the *Interconnect* chapter.

The following sections describe the register mapping of the L3 and L4 interconnects. The software configures these registers.

2.3.1 L3 Memory Space Mapping

The L3 interconnect control registers are mapped in a 16Mbytes space and allow the configuration of the L3 interconnect parameters.

The L3 default settings are fully functional and enable all possible functional data paths. However, the interconnect parameters can be changed to accommodate expectations.

Accesses to the L3 interconnect can be configured on a per-module basis using the internal L3 registers, which are grouped into five register block types:

- IA: initiator agent configuration registers
- TA: target agent configuration registers
- RT: register target (global configuration registers)
- PM: protection mechanism (firewalls) configuration registers
- SI: global sideband signal configuration registers

For more information, see the *Interconnect* chapter.

This section gives information about all modules and features in the high-tier device. to check availability of modules and features, see Chapter 1, *Device Family*. The memory area of unavailable modules and features is RESERVED, read is undefined, and write can lead to unpredictable behavior.

Table 2-2 describes the mapping of the L3 interconnect control registers.

Table 2-2. L3 Control Register Mapping

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
L3 RT	0x6800 0000	0x6800 03FF	1KB	L3 configuration registers
L3 SI	0x6800 0400	0x6800 07FF	1KB	Sideband signals configuration
Reserved	0x6800 0800	0x6800 13FF	3KB	Reserved
MPU SS IA	0x6800 1400	0x6800 17FF	1KB	MPU subsystem instruction port agent configuration
Unused	0x6800 1800	0x6800 1BFF	1KB	Unused
SGX SS IA	0x6800 1C00	0x6800 1FFF	1KB	SGX subsystem initiator port agent configuration
SMS TA	0x6800 2000	0x6800 23FF	1KB	SMS target port agent configuration
GPMC TA	0x6800 2400	0x6800 27FF	1KB	GPMC target port agent configuration
OCM RAM TA	0x6800 2800	0x6800 2BFF	1KB	OCM RAM target port agent configuration
OCM ROM TA	0x6800 2C00	0x6800 2FFF	1KB	OCM ROM target port agent configuration
Reserved	0x6800 3000	0x6800 3FFF	4KB	Reserved

Table 2-2. L3 Control Register Mapping (continued)

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
HS USB HOST IA	0x6800 4000	0x6800 43FF	1KB	HS USB HOST initiator port agent configuration
IPSS IA	0x6800 4400	0x6800 47FF	1KB	IPSS
Reserved	0x6800 4800	0x6800 4BFF	1KB	Reserved
sDMA RD IA	0x6800 4C00	0x6800 4FFF	1KB	sDMA RD initiator port agent configuration
sDMA WR IA	0x6800 5000	0x6800 53FF	1KB	sDMA WR initiator port agent configuration
Display SS IA	0x6800 5400	0x6800 57FF	1KB	Display subsystem initiator port agent configuration
Unused	0x6800 5800	0x6800 5BFF	1KB	Unused
DAP IA	0x6800 5C00	0x6800 5FFF	1KB	Debug access port initiator port agent configuration
IPSS TA	0x6800 6000	0x6800 63FF	1KB	IPSS Target Port agent configuration
SGX SS TA	0x6800 6400	0x6800 67FF	1KB	SGX subsystem target port agent configuration
L4-Core TA	0x6800 6800	0x6800 6BFF	1KB	L4-Core target port agent configuration
L4-Per TA	0x6800 6C00	0x6800 6FFF	1KB	L4-Per target port agent configuration
Reserved	0x6800 7000	0x6800 FFFF	36KB	Reserved
RT PM	0x6801 0000	0x6801 03FF	1KB	Register target port protection
Reserved	0x6801 0400	0x6801 23FF	8KB	Reserved
GPMC PM	0x6801 2400	0x6801 27FF	1KB	GPMC target port protection
OCM RAM PM	0x6801 2800	0x6801 2BFF	1KB	OCM RAM target port protection
OCM ROM PM	0x6801 2C00	0x6801 2FFF	1KB	OCM ROM target port protection
Reserved	0x6801 3000	0x6801 3FFF	4KB	Reserved
IPSS PM	0x6801 4000	0x6801 43FF	1KB	IPSS
Reserved	0x6801 4400	0x68FF FFFF	16,303KB	Reserved

2.3.2 L4 Memory Space Mapping

The device contains four L4 interconnects: the L4-Core, L4-Wakeup, L4-Per, and L4-Emu interconnects.

As with the L3 interconnect, the L4 interconnects can be configured to tune the access depending on the characteristics of each module.

For more information on the L4 interconnect, see the *Interconnect* chapter.

2.3.2.1 L4-Core Memory Space Mapping

The L4-Core interconnect is a 16Mbytes space composed of the L4-Core interconnect configuration registers and the module registers.

Table 2-3 describes the mapping of the registers for the L4-Core interconnect.

NOTE: All memory spaces described as modules provide direct access to module registers outside the L4-Core interconnect. All other accesses are internal to the L4-Core interconnect.

This section gives information about all modules and features in the high-tier device. To check availability of modules and features, see Chapter 1, *Device Family*. The memory area of unavailable modules and features is RESERVED, read is undefined, and write can lead to unpredictable behavior.

Table 2-3. L4-Core Memory Space Mapping [1]

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
L4-Core	0x4800 0000	0x48FF FFFF	16MB	
Reserved	0x4800 0000	0x4800 1FFF	8KB	Reserved
System control module	0x4800 2000	0x4800 2FFF	4KB	Module
	0x4800 3000	0x4800 3FFF	4KB	L4 interconnect
Clock manager • DPLL • Clock manager	0x4800 4000	0x4800 5FFF	8KB	Module region A
	0x4800 6000	0x4800 67FF	2KB	Module region B
	0x4800 6800	0x4800 6FFF	2KB	Reserved
	0x4800 7000	0x4800 7FFF	4KB	L4 interconnect
Reserved	0x4800 8000	0x4802 3FFF	112KB	Reserved
Reserved	0x4802 4000	0x4802 4FFF	4KB	Reserved
	0x4802 5000	0x4802 5FFF	4KB	Reserved
Reserved	0x4802 6000	0x4803 FFFF	104KB	Reserved
L4-Core configuration	0x4804 0000	0x4804 07FF	2KB	Address/protection (AP)
	0x4804 0800	0x4804 0FFF	2KB	Initiator port (IP)
	0x4804 1000	0x4804 1FFF	4KB	Link agent (LA)
Reserved	0x4804 2000	0x4804 FBFF	55KB	Reserved
Reserved	0x4804 FC00	0x4804 FFFF	1KB	Reserved
Display subsystem • Display subsystem top • Display controller • RFBI • Video encoder	0x4805 0000	0x4805 03FF	1KB	Display subsystem top
	0x4805 0400	0x4805 07FF	1KB	Display controller
	0x4805 0800	0x4805 0BFF	1KB	RFBI
	0x4805 0C00	0x4805 0FFF	1KB	Video encoder
	0x4805 1000	0x4805 1FFF	4KB	L4 interconnect
Reserved	0x4805 2000	0x4805 5FFF	16KB	Reserved
sDMA	0x4805 6000	0x4805 6FFF	4KB	Module
	0x4805 7000	0x4805 7FFF	4KB	L4 interconnect
Reserved	0x4805 8000	0x4805 FFFF	32KB	Reserved
I2C3	0x4806 0000	0x4806 0FFF	4KB	Module
	0x4806 1000	0x4806 1FFF	4KB	L4 interconnect
USBTLL module	0x4806 2000	0x4806 2FFF	4KB	Module
	0x4806 3000	0x4806 3FFF	4KB	L4 interconnect
HS USB HOST	0x4806 4000	0x4806 4FFF	4KB	Module
	0x4806 5000	0x4806 5FFF	4KB	L4 interconnect
Reserved	0x4806 6000	0x4806 9FFF	16KB	Reserved
UART1	0x4806 A000	0x4806 AFFF	4KB	Module
	0x4806 B000	0x4806 BFFF	4KB	L4 interconnect
UART2	0x4806 C000	0x4806 CFFF	4KB	Module
	0x4806 D000	0x4806 DFFF	4KB	L4 interconnect
Reserved	0x4806 E000	0x4806 FFFF	8KB	Reserved
I2C1	0x4807 0000	0x4807 0FFF	4KB	Module
	0x4807 1000	0x4807 1FFF	4KB	L4 interconnect
I2C2	0x4807 2000	0x4807 2FFF	4KB	Module
	0x4807 3000	0x4807 3FFF	4KB	L4 interconnect
McBSP1 (Digital baseband data)	0x4807 4000	0x4807 4FFF	4KB	Module
	0x4807 5000	0x4807 5FFF	4KB	L4 interconnect
Reserved	0x4807 6000	0x4808 5FFF	64KB	Reserved

[1] The registers mapped in this range are shadow registers of the first 2Kbytes region A [0x4800 4000 - 0x4800 47FF]. Region A and region B share the same port.

Table 2-3. L4-Core Memory Space Mapping [2] (continued)

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
GPTIMER10	0x4808 6000	0x4808 6FFF	4KB	Module
	0x4808 7000	0x4808 7FFF	4KB	L4 interconnect
GPTIMER11	0x4808 8000	0x4808 8FFF	4KB	Module
	0x4808 9000	0x4808 9FFF	4KB	L4 interconnect
Reserved	0x4808 A000	0x4809 3FFF	40KB	Reserved
Unused	0x4809 4000	0x4809 5FFF	8KB	Unused
McBSP5 (MIDI data)	0x4809 6000	0x4809 6FFF	4KB	Module
	0x4809 7000	0x4809 7FFF	4KB	L4 interconnect
McSPI1	0x4809 8000	0x4809 8FFF	4KB	Module
	0x4809 9000	0x4809 9FFF	4KB	L4 interconnect
McSPI2	0x4809 A000	0x4809 AFFF	4KB	Module
	0x4809 B000	0x4809 BFFF	4KB	L4 interconnect
MMC/SD/SDIO1	0x4809 C000	0x4809 CFFF	4KB	Module
	0x4809 D000	0x4809 DFFF	4KB	L4 interconnect
UART4	0x4809 E000	0x4809 EFFF	4KB	Module
	0x4809 F000	0x4809 FFFF	4KB	L4 interconnect
RNG	0x480A 0000	0x480A FFFF	4KB	Module
	0x480A 1000	0x480A 1FFF	4KB	L4 interconnect
DES3DES1	0x480A 2000	0x480A 2FFF	4KB	Module
	0x480A 3000	0x480A 3FFF	4KB	L4 interconnect
SHA2MD5	0x480A 4000	0x480A 4FFF	4KB	Module
	0x480A 5000	0x480A 5FFF	4KB	L4 interconnect
AES1	0x480A 6000	0x480A 6FFF	4KB	Module
	0x480A 7000	0x480A 7FFF	4KB	L4 interconnect
Fast PKA	0x480A 8000	0x480A 8FFF	4KB	Module
	0x480A 9000	0x480A 9FFF	4KB	L4 interconnect
USB High Speed 2.0	0x480A B000	0x480A BFFF	4KB	Module
	0x480A C000	0x480A CFFF	4KB	L4 interconnect
MMC/SD/SDIO3	0x480A D000	0x480A DFFF	4KB	Module
	0x480A E000	0x480A EFFF	4KB	L4 interconnect
Reserved	0x480A F000	0x480A FFFF	4KB	Reserved
Reserved	0x480B 0000	0x480B 0FFF	4KB	Module
	0x480B 1000	0x480B 1FFF	4KB	Reserved
HDQ/1-wire	0x480B 2000	0x480B 2FFF	4KB	Reserved
	0x480B 3000	0x480B 3FFF	4KB	L4 interconnect
MMC/SD/SDIO2	0x480B 4000	0x480B 4FFF	4KB	Module
	0x480B 5000	0x480B 5FFF	4KB	L4 interconnect
Unused	0x480B 6000	0x480B 7FFF	8KB	Unused
McSPI3	0x480B 8000	0x480B 8FFF	4KB	Module
	0x480B 9000	0x480B 9FFF	4KB	L4 interconnect
McSPI4	0x480B A000	0x480B AFFF	4KB	Module
	0x480B B000	0x480B BFFF	4KB	L4 interconnect
Unused	0x480B C000	0x480C 0FFF	20KB	Unused
DES3DES2	0x480C 1000	0x480C 1FFF	4KB	Module
	0x480C 2000	0x480C 2FFF	4KB	L4 interconnect
SHA1MD5 2	0x480C 3000	0x480C 3FFF	4KB	Module
	0x480C 4000	0x480C 4FFF	4KB	L4 interconnect

Table 2-3. L4-Core Memory Space Mapping [2] (continued)

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
AES2	0x480C 5000	0x480C 5FFF	4KB	Module
	0x480C 6000	0x480C 6FFF	4KB	L4 interconnect
Unused	0x480C 7000	0x480C AFFF	16KB	Unused
SR2	0x480C B000	0x480C BFFF	4KB	Module
	0x480C C000	0x480C CFFF	4KB	L4 interconnect
Unused	0x480C D000	0x480C EFFF	8KB	Unused
CPFROM	0x480CF000	0x480C FFFF	4KB	Module
	0x480D 0000	0x480D 0FFF	4KB	L4 interconnect
Reserved	0x480D 1000	0x481F FFFF	1212KB	Reserved
Interrupt controller 1	0x4820 0000	0x4820 0FFF	4KB	Non-shared device mapping
Reserved	0x4201 0000	0x482F FFFF	508KB	Non-shared device mapping
S.M.	0x4828 0000	0x4828 0FFF	4KB	Non-shared device mapping
Reserved	0x4828 1000	0x482F FFFF	506KB	Non-shared device mapping
L4-Wakeup interconnect (region A)	0x4830 0000	0x4830 9FFF	40KB	Non-shared device mapping
Control module ID code	0x4830 A000	0x4830 AFFF	4KB	See Table 2-4
	0x4830 B000	0x4830 BFFF	4KB	L4 interconnect
L4-Wakeup interconnect (Region B)	0x4830 C000	0x4833 FFFF	208KB	See Table 2-4
	0x4834 0000	0x4834 0FFF	4KB	L4 interconnect
Reserved	0x4834 1000	0x48FF EFFF	13,052KB	Reserved

2.3.2.2 L4-Wakeup Memory Space Mapping

The L4-Wakeup interconnect is a 256Kbytes space composed of the L4-Wakeup interconnect configuration registers and the module registers.

Table 2-4 describes the mapping of the registers for the L4-Wakeup interconnect.

NOTE: All memory spaces described as modules provide direct access to module registers outside the L4-Wakeup interconnect. All other accesses are internal to the L4-Wakeup interconnect.

Table 2-4. L4-Wakeup Memory Space Mapping

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
L4-Wakeup	0x4830 0000	0x4833 FFFF	256KB	
Reserved	0x4830 0000	0x4830 3FFF	16KB	Reserved
GPTIMER12	0x4830 4000	0x4830 4FFF	4KB	Module
	0x4830 5000	0x4830 5FFF	4KB	L4 interconnect
Power and reset manager • Power manager • Reset manager	0x4830 6000	0x4830 7FFF	8KB	Module region A
	0x4830 8000	0x4830 87FF	2KB	Module region B[1]
	0x4830 8800	0x4830 8FFF	2KB	Reserved
	0x4830 9000	0x4830 9FFF	4KB	L4 interconnect
Reserved	0x4830 A000	0x4830 BFFF	8KB	Reserved

[1] The registers mapped in this range are shadow registers of the first 2Kbytes region A [0x4830 6000 - 0x4830 67FF]. Region A and region B share the same port.

Table 2-4. L4-Wakeup Memory Space Mapping (continued)

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
WDTIMER1	0x4830 C000	0x4830 CFFF	4KB	Module
	0x4830 D000	0x4830 DFFF	4KB	L4 interconnect
GPIO1	0x4831 0000	0x4831 0FFF	4KB	Module
	0x4831 1000	0x4831 1FFF	4KB	L4 interconnect
Reserved	0x4831 2000	0x4831 3FFF	8KB	Reserved
WDTIMER2	0x4831 4000	0x4831 4FFF	4KB	Module
	0x4831 5000	0x4831 5FFF	4KB	L4 interconnect
Reserved	0x4831 6000	0x4831 7FFF	8KB	Reserved
GPTIMER1	0x4831 8000	0x4831 8FFF	4KB	Module
	0x4831 9000	0x4831 9FFF	4KB	L4 interconnect
Reserved	0x4831 A000	0x4831 FFFF	24KB	Reserved
32KTIMER	0x4832 0000	0x4832 0FFF	4KB	Module
	0x4832 1000	0x4832 1FFF	4KB	L4 interconnect
Reserved	0x4832 2000	0x4832 7FFF	24KB	Reserved
L4-Wakeup configuration	0x4832 8000	0x4832 87FF	2KB	Address/protection (AP)
	0x4832 8800	0x4832 8FFF	2KB	Initiator port (IP) L4-Core
	0x4832 9000	0x4832 9FFF	4KB	Link agent (LA)
	0x4832 A000	0x4832 A7FF	2KB	Initiator port (IP) L4-Emu
Reserved	0x4832 A800	0x4833 FFFF	86KB	Reserved

2.3.2.3 L4-Peripheral Memory Space Mapping

The L4-Per interconnect is a 1Mbyte space composed of the L4-Per interconnect configuration registers and the module registers.

Table 2-5 describes the mapping of the registers for the L4-Per interconnect.

NOTE: All memory spaces described as modules provide direct access to the module registers outside the L4-Per interconnect. All other accesses are internal to the L4-Per interconnect.

Table 2-5. L4-Peripheral Memory Space Mapping

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
L4-Per	0x4900 0000	0x490F FFFF	1MB	
L4-Per configuration	0x4900 0000	0x4900 07FF	2KB	Address/protection (AP)
	0x4900 0800	0x4900 0FFF	2KB	Initiator port (IP)
	0x4900 1000	0x4900 1FFF	4KB	Link agent (LA)
Reserved	0x4900 2000	0x4901 FFFF	120KB	Reserved
UART3 (Infrared)	0x4902 0000	0x4902 0FFF	4KB	Module
	0x4902 1000	0x4902 1FFF	4KB	L4 interconnect
McBSP2 (Audio for codec)	0x4902 2000	0x4902 2FFF	4KB	Module
	0x4902 3000	0x4902 3FFF	4KB	L4 interconnect
McBSP3 (Bluetooth voice data)	0x4902 4000	0x4902 4FFF	4KB	Module
	0x4902 5000	0x4902 5FFF	4KB	L4 interconnect
McBSP4 (Digital baseband voice data)	0x4902 6000	0x4902 6FFF	4KB	Module
	0x4902 7000	0x4902 7FFF	4KB	L4 interconnect

Table 2-5. L4-Peripheral Memory Space Mapping (continued)

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
McBSP2 (Sidetone)	0x4902 8000	0x4902 8FFF	4KB	Module
	0x4902 9000	0x4902 9FFF	4KB	L4 interconnect
McBSP3 (Sidetone)	0x4902 A000	0x4902 AFFF	4KB	Module
	0x4902 B000	0x4902 BFFF	4KB	L4 interconnect
Reserved	0x4902 C000	0x4902 FFFF	16KB	Reserved
WDTIMER3	0x4903 0000	0x4903 0FFF	4KB	Module
	0x4903 1000	0x4903 1FFF	4KB	L4 interconnect
GPTIMER2	0x4903 2000	0x4903 2FFF	4KB	Module
	0x4903 3000	0x4903 3FFF	4KB	L4 interconnect
GPTIMER3	0x4903 4000	0x4903 4FFF	4KB	Module
	0x4903 5000	0x4903 5FFF	4KB	L4 interconnect
GPTIMER4	0x4903 6000	0x4903 6FFF	4KB	Module
	0x4903 7000	0x4903 7FFF	4KB	L4 interconnect
GPTIMER5	0x4903 8000	0x4903 8FFF	4KB	Module
	0x4903 9000	0x4903 9FFF	4KB	L4 interconnect
GPTIMER6	0x4903 A000	0x4903 AFFF	4KB	Module
	0x4903 B000	0x4903 BFFF	4KB	L4 interconnect
GPTIMER7	0x4903 C000	0x4903 CFFF	4KB	Module
	0x4903 D000	0x4903 DFFF	4KB	L4 interconnect
GPTIMER8	0x4903 E000	0x4903 EFFF	4KB	Module
	0x4903 F000	0x4903 FFFF	4KB	L4 interconnect
GPTIMER9	0x4904 0000	0x4904 0FFF	4KB	Module
	0x4904 1000	0x4904 1FFF	4KB	L4 interconnect
Reserved	0x4904 2000	0x4904 FFFF	56KB	Reserved
GPIO2	0x4905 0000	0x4905 0FFF	4KB	Module
	0x4905 1000	0x4905 1FFF	4KB	L4 interconnect
GPIO3	0x4905 2000	0x4905 2FFF	4KB	Module
	0x4905 3000	0x4905 3FFF	4KB	L4 interconnect
GPIO4	0x4905 4000	0x4905 4FFF	4KB	Module
	0x4905 5000	0x4905 5FFF	4KB	L4 interconnect
GPIO5	0x4905 6000	0x4905 6FFF	4KB	Module
	0x4905 7000	0x4905 7FFF	4KB	L4 interconnect
GPIO6	0x4905 8000	0x4905 8FFF	4KB	Module
	0x4905 9000	0x4905 9FFF	4KB	L4 interconnect
Reserved	0x4905 A000	0x490F FFFF	664KB	Reserved

2.3.2.4 L4-Emulation Memory Space Mapping

The L4-Emu interconnect is an 8Mbytes space composed of the L4-Emu interconnect configuration registers and module registers.

Table 2-6 describes the mapping of the registers for the L4-Emu interconnect.

NOTE: All memory spaces described as modules provide direct access to the module registers outside the L4-Emu interconnect. All other accesses are internal to the L4-Emu interconnect

Table 2-6. L4-Emulation Memory Space Mapping

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
L4-Emu	0x5400 0000	0x547F FFFF	8MB	
Reserved	0x5400 0000	0x5400 3FFF	16KB	Reserved
TEST chip-level TAP	0x5400 4000	0x5400 4FFF	4KB	Module
	0x5400 5000	0x5400 5FFF	4KB	L4 interconnect
L4-Emu configuration	0x5400 6000	0x5400 67FF	2KB	Address/protection (AP)
	0x5400 6800	0x5400 6FFF	2KB	Initiator port (IP) L4-Core
	0x5400 7000	0x5400 7FFF	4KB	Link agent (LA)
	0x5400 8000	0x5400 87FF	2KB	Initiator port (IP) DAP
Reserved	0x5400 8800	0x5400 FFFF	30KB	Reserved
MPU emulation	0x5401 0000	0x5401 7FFF	16KB	Module
	0x5401 8000	0x5401 7FFF	4KB	L4 interconnect
TPIU	0x5401 9000	0x5401 9FFF	4KB	Module
	0x5401 A000	0x5401 AFFF	4KB	L4 interconnect
ETB	0x5401 B000	0x5401 BFFF	4KB	Module
	0x5401 C000	0x5401 CFFF	4KB	L4 interconnect
DAPCTL	0x5401 D000	0x5401 DFFF	4KB	Module
	0x5401 E000	0x5401 EFFF	4KB	L4 interconnect
SDTI	0x5401 F000	0x5401 FFFF	4KB	L4 interconnect
	0x5402 0000	0x544F FFFF	4992KB	Reserved
	0x5450 0000	0x5450 FFFF	4KB	SDTI module (configuration)
	0x5451 0000	0x545F FFFF	1984KB	Reserved
	0x5460 0000	0x546F FFFF	1MB	SDTI module (window)
Reserved	0x5470 0000	0x5470 3FFF	16KB	Reserved
GPTIMER12	0x5470 4000	0x5470 4FFF	4KB	Module
	0x5470 5000	0x5470 5FFF	4KB	L4 interconnect
Power and reset manager • Power manager • Reset manager (WAKEUP domain[1])	0x5470 6000	0x5470 7FFF	8KB	Module region A
	0x5470 8000	0x5470 87FF	2KB	Module region B[2]
	0x5470 8800	0x5470 8FFF	2KB	Reserved
	0x5470 9000	0x5470 9FFF	4KB	L4 interconnect
Reserved	0x5470 A000	0x5470 BFFF	8KB	Reserved
WDTIMER1	0x5470 C000	0x5470 CFFF	4KB	Module
	0x5470 D000	0x5470 DFFF	4KB	L4 interconnect
Reserved	0x5470 E000	0x5470 FFFF	8KB	Reserved
GPIO1 (WAKEUP domain[1])	0x5471 0000	0x5471 0FFF	4KB	Module
	0x5471 1000	0x5471 1FFF	4KB	L4 interconnect
Reserved	0x5471 2000	0x5471 3FFF	8KB	Reserved
WDTIMER2 (WAKEUP domain[1])	0x5471 4000	0x5471 4FFF	4KB	Module
	0x5471 5000	0x5471 5FFF	4KB	L4 interconnect
Reserved	0x5471 6000	0x5471 7FFF	8KB	Reserved
GPTIMER1 (WAKEUP domain[1])	0x5471 8000	0x5471 8FFF	4KB	Module
	0x5471 9000	0x5471 9FFF	4KB	L4 interconnect
Reserved	0x5471 A000	0x5471 FFFF	24KB	Reserved
32KTIMER (WAKEUP domain[1])	0x5472 0000	0x5472 0FFF	4KB	Module
	0x5472 1000	0x5472 1FFF	4KB	L4 interconnect

(1) These modules are accessed through the L4-Wakeup interconnect (for emulation purpose only).
(2) The registers mapped in this range are shadow registers of the first 2Kbytes region A [0x5470 6000 - 0x5470 67FF]. Region A and region B share the same port.

Table 2-6. L4-Emulation Memory Space Mapping (continued)

Device Name	Start Address (HEX)	End Address (HEX)	Size	Description
Reserved	0x5472 2000	0x5472 7FFF	24KB	Reserved
L4-Wakeup configuration (WAKEUP domain[1])	0x5472 8000	0x5472 87FF	2KB	Address/protection (AP)
	0x5472 8800	0x5472 8FFF	2KB	Initiator port (IP) L4-Core
	0x5472 9000	0x5472 9FFF	4KB	Link agent (LA)
	0x5472 A000	0x5472 A7FF	2KB	Initiator port (IP) L4-Emu
Reserved	0x5472 A800	0x5472 FFFF	22KB	Reserved
L4 WKUP	0x5473 0000	0x5473 0FFF	4KB	Module
Reserved	0x5473 1000	0x547F FFFF	828KB	Reserved

2.3.3 Register Access Restrictions

This section gives information about all modules and features in the high-tier device. To check availability of modules and features, see Chapter 1, *Device Family*. The memory area of unavailable modules and features is RESERVED, read is undefined, and write can lead to unpredictable behavior.

Table 2-7 gives the supported data access widths per module.

Table 2-7. Register Access Restrictions

Module	Allowed Access
MPU subsystem	8-bit/16-bit/32-bit
IPSS	32-bit
SGX	32-bit
Display subsystem	32-bit
GPMC	8-bit/16-bit/32-bit
SMS	8-bit/16-bit/32-bit
EMIF4	8-bit/16-bit/32-bit
sDMA	8-bit/16-bit/32-bit
HS USB HOST	32-bit
USBTLL	32-bit
USB - ULPI and UTMI registers	8-bit
L3 interconnect	8-bit/16-bit/32-bit
L4-Wakeup interconnect	8-bit/16-bit/32-bit
L4-Core interconnect	8-bit/16-bit/32-bit
Clock manager	32-bit
Power and reset manager	32-bit
System control module	8-bit/16-bit/32-bit
32KTIMER	16-bit/32-bit
GPIO	8-bit/16-bit/32-bit
GPTIMER	16-bit/32-bit
WDTIMER	16-bit/32-bit
I2C	8-bit/16-bit
HDQ/1-wire	32-bit
McBSP	32-bit
Sidetone	8-bit/16-bit/32-bit
McSPI	8-bit/16-bit/32-bit
UART	8-bit/16-bit/32-bit
MMC/SD/SDIO	32-bit
MPU INTC	16-bit/32-bit

Table 2-7. Register Access Restrictions (continued)

Module	Allowed Access
SR	8-bit/16-bit/32-bit

2.4 IPSS Memory Space Mapping

2.4.1 L3 Interconnect View of the IPSS Memory Space

Table 2-8 lists the IPSS memory space mapping from the perspective of the MPU subsystem through the L3 interconnect.

Table 2-8. L3 Interconnect View of the IPSS Memory Space

Region Name	Start Address (HEX)	End Address (HEX)	Size	Description
CPGMAC-TOP	0x5C00 0000	0x5C00 00FF	256B	
Reserved	0x5C00 0100	0x5C00 FFFF	63.75KB	Reserved
CPGMAC-CPGMAC	0x5C01 0000	0x5C01 07FF	2KB	
Reserved	0x5C01 0800	0x5C01 FFFF	62KB	Reserved
CPGMAC-CPPI	0x5C02 0000	0x5C02 1FFF	8KB	
Reserved	0x5C02 2000	0x5C02 FFFF	56KB	Reserved
CPGMAC-MDIO	0x5C03 0000	0x5C03 00FF	256B	
Reserved	0x5C03 0100	0x5C03 FFFF	63.75KB	Reserved
USBOTGSS	0x5C04 0000	0x5C04 7FFF	32KB	
Reserved	0x5C04 8000	0x5C04 FFFF	32KB	Reserved
HECC	0x5c05 0000	0x5C05 3FFF	16KB	
Reserved	0x5C05 4000	0x5C05 FFFF	48KB	Reserved
VPFE	0x5c06 0000	0x5C06 FFFF	64KB	
Reserved (valid)	0x5C07 0000	0x5EFF FFFF	47.56MB	Reserved. Null response is provided by SCR

MPU Subsystem

This chapter describes the microprocessor unit (MPU) subsystem.

Topic **Page**

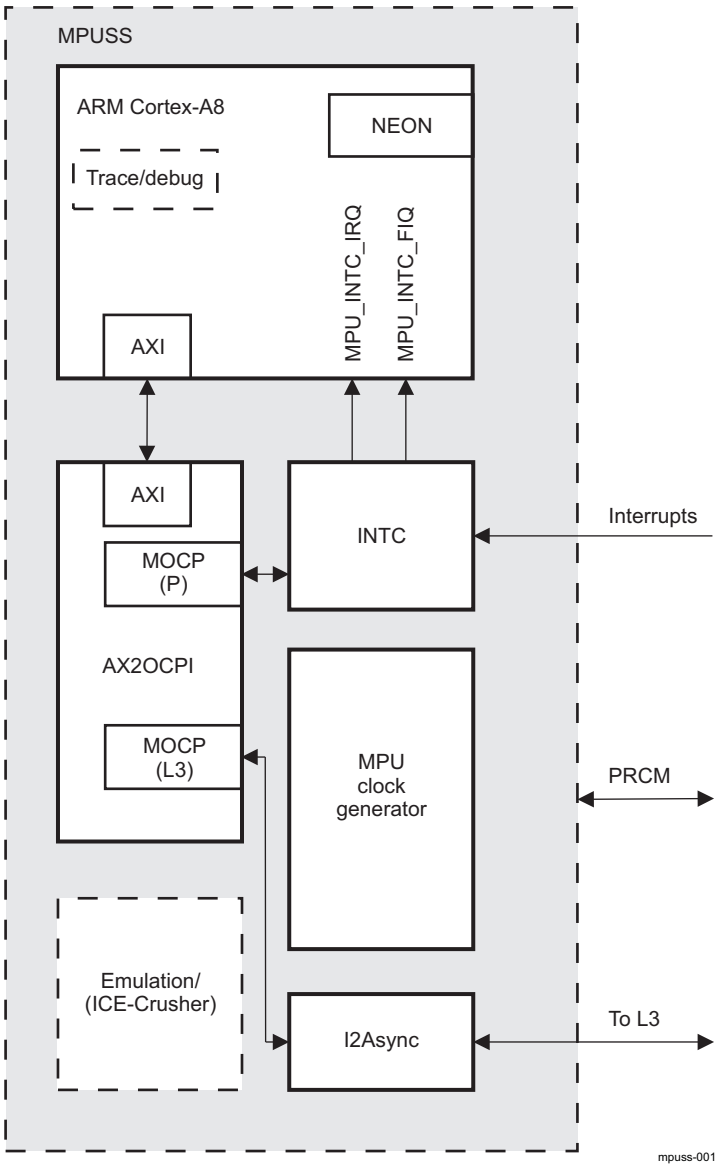
3.1 MPU Subsystem Overview

3.1.1 Introduction

The MPU subsystem of the device handles transactions among the ARM core, the L3 interconnect, and the interrupt controller (INTC).

The MPU subsystem is a hard macro that integrates the ARM subchip with additional logic for protocol conversion, emulation, interrupt handling, and debug enhancements. Figure 3-1 is a high-level block diagram of the MPU subsystem.

Figure 3-1. MPU Subsystem Overview



3.1.2 Features

The MPU subsystem integrates the following:

- ARM subchip
  - Public ARM Cortex™-A8 core
  - ARM version 7 ISA: Standard ARM instruction set + Thumb-2™, Jazelle® RCT Java accelerator, and media extensions
  - Neon single instruction, multiple data (SIMD) coprocessor (VFP light + media streaming instructions)
  - Cache memories
    - Level 1: 16-KB instruction and 16-KB data caches – 4-way associative, 64 bytes/line
    - Level 2: See Section 1.4, .
  - Emulation/debug
- INTC of 96 synchronous level-sensitive interrupt lines (For details, see Chapter 8, *Interrupt Controller*.)
- AXI2OCP bridge between ARM AXI bus, L3 master open-core protocol (OCP) bus, and INTC master OCP bus
- MPU clock generator: Clock generation module that generate clocks, power modes, and idle and active acknowledge signals
- Debug, trace, and emulation features: ICECrusher™, embedded trace macrocell (ETM), advanced peripheral bus (APB) modules. Cortex-A8 MPU implements an APB slave interface that allows access to ETM, ICECrusher, and debug registers.

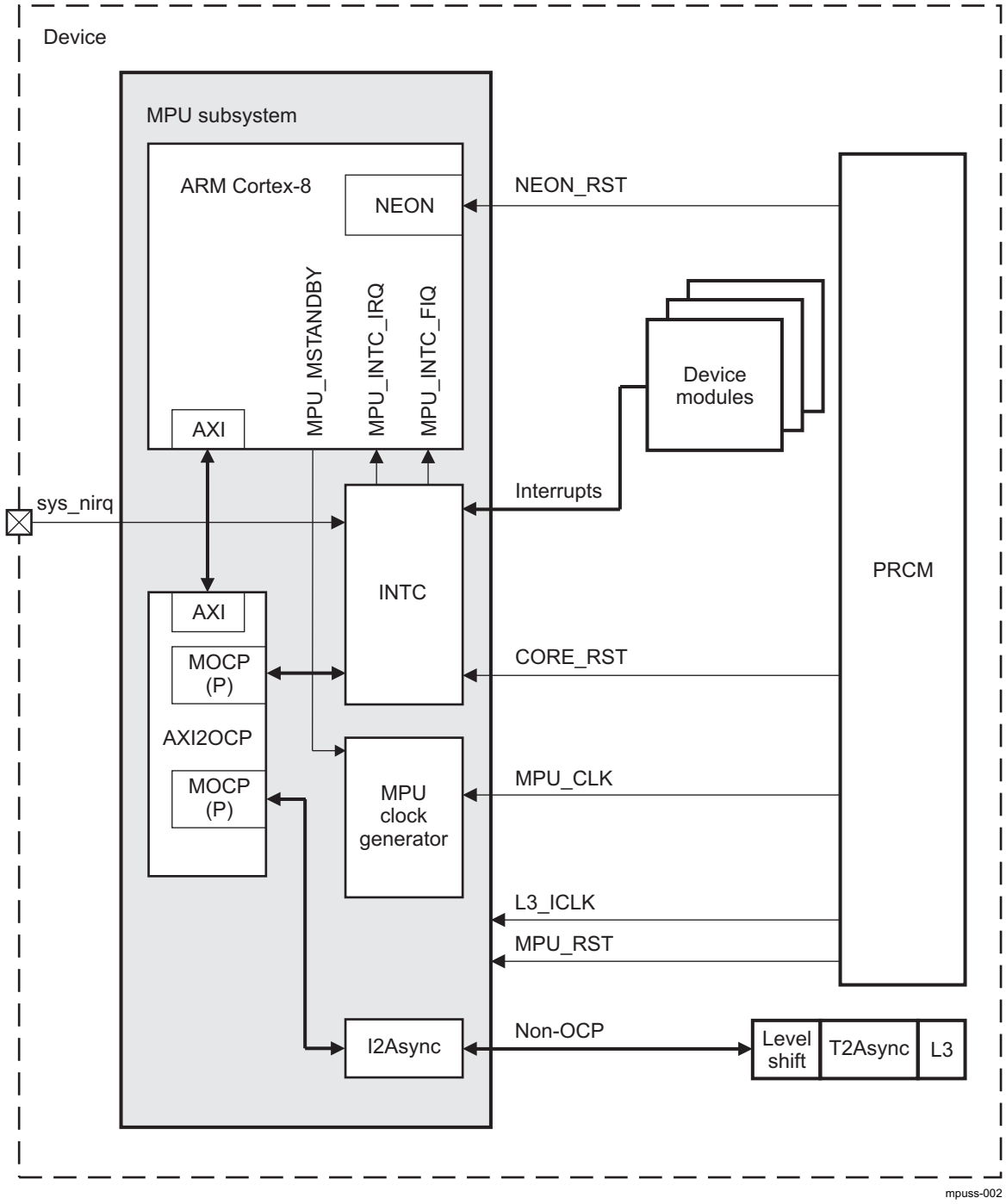
3.2 MPU Subsystem Integration

The MPU subsystem integrates the following group of submodules:

- Public ARM Cortex-A8 MPU: Provides a high processing capability, including the Neon technology for mobile multimedia acceleration. The ARM communicates through an AXI bus with the AXI2OCP bridge and receives interrupts from the MPU INTC.
- INTC: Handles module interrupts (for details, see Chapter 8, *Interrupt Controller*).
- AXI2OCP bridge: Allows communication among the ARM (AXI), the INTC (OCP), and the modules (OCP L3).
- I2Async bridge: An asynchronous bridge interface providing an asynchronous OCP-to-OCP interface. This interface is between the AXI2OCP bridge in the MPU subsystem and the T2Async bridge external to the MPU subsystem. The T2Async bridge connects to OCP L3.
- MPU clock generator: Provides clocks to internal modules of the MPU subsystem; fed by the MPU digital phase-locked loop (DPLL) of the power, reset, and clock management (PRCM) module of the device. The MPU DPLL generates the clock for the ARM Cortex-A8 CPU and the Cortex-A8 MPU subsystem logic. The power, reset, and MPU DPLL source clock are generated from the device PRCM module.

Figure 3-2 shows the signals that interface with the external modules.

Figure 3-2. MPU Subsystem Integration Overview



NOTE: Some debug, trace, and emulation features are implemented in the MPU subsystem. This chapter includes only clock/reset inputs and power-management aspects for these features.

3.2.1 MPU Subsystem Clock and Reset Distribution

3.2.1.1 Clock Distribution

The MPU subsystem includes a clock generator block that supplies clocks for the modules in the MPU subsystem. It is fed by the MPU_CLK clock from the PRCM module.

All major modules in the MPU subsystem are clocked at half the frequency of the ARM core. The divider of the output clock can be programmed with the PRCM.CM_CLKSEL2_PLL_MPU[4:0] MPU_DPLL_CLKOUT_DIV bit field; the frequency is relative to the ARM core. For details see , *Power, Reset, and Clock Management (PRCM) module* . The clock generator generates the following functional clocks:

- **ARM (ARM_FCLK):** This is the core clock. It is the base fast clock that is routed internally to the ARM logic and internal RAMs, including Neon, L2 cache, the ETM core (emulation), and the ARM core. It runs at one half the frequency of the MPU_CLK when DPLL1 is locked, and runs at the same frequency as the MPU_CLK when DPLL1 is bypassed.
- **AXI2OCP clock (AXI_FCLK):** This clock is half the frequency of the ARM clock (ARM_FCLK). The OCP interface thus performs at one half the frequency of ARM.
- **INTC functional clock (MPU_INTC_FCLK):** This clock, which is part of the INTC module, is half the frequency of the ARM clock (ARM_FCLK).
- **ICE-Crusher functional clock (ICECRUSHER_FCLK):** ICECrusher clocking operates on the APB interface, using the ARM core clocking.
- **I2Async clock (I2ASYNC_FCLK):** This clock is half the frequency of the ARM clock (ARM_FCLK). It matches the OCP interface of the AXI2OCP bridge.

NOTE: The second half of the asynchronous bridge (T2ASYNC) is clocked directly by the PRCM module with the core clock. T2ASYNC is not part of the MPU subsystem.

Emulation clocking: Except for the ICECrusher functional clock, which is provided by the MPU DLL, the emulation modules in the MPU subsystem are not generated by the MPU subsystem DPLL, but by an EMU DPLL. These clocks (EMU_CLOCKS) are distributed by the PRCM module, are asynchronous with the ARM core clock (ARM_FCLK) and can run at a maximum of 1/3 the ARM core clock.

Figure 3-3 shows the MPU subsystem clocking scheme.

Figure 3-3. MPU Subsystem Clocking Scheme

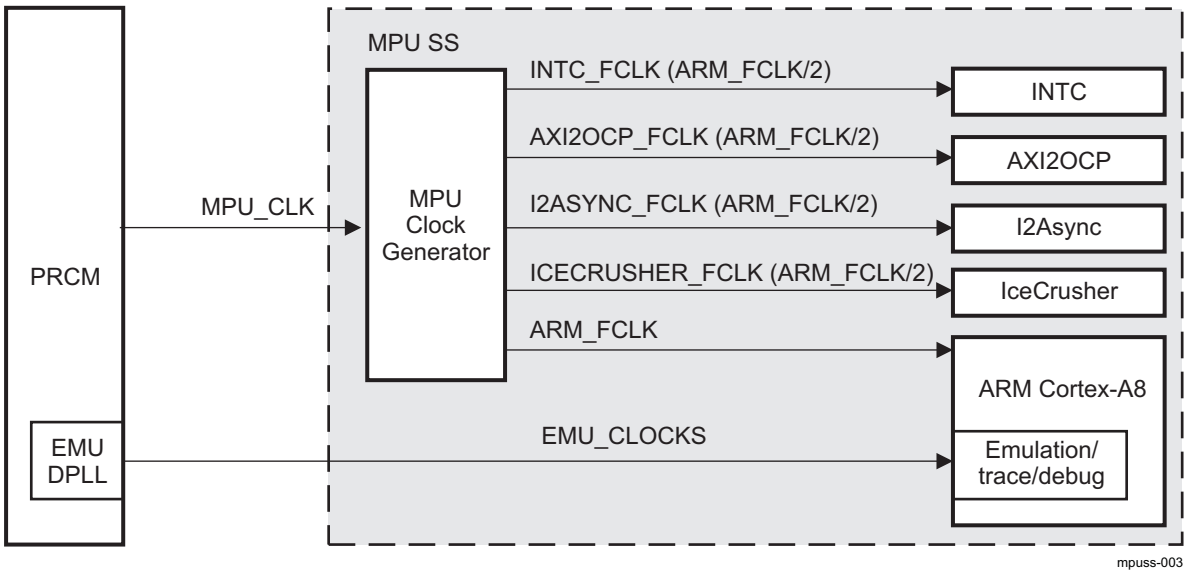


Table 3-1 lists the clocks generated in the MPU subsystem by the MPU clock generator.

Table 3-1. MPU Clock Generator Clock Signals

Signal Name	I/O	Interface	Comments
MPU_CLK	I	PRCM	MPU DPLL clock out
ARM_FCLK	O	ARM	ARM functional clock
MPU_INTC_FCLK	O	MPU INTC	MPU INTC functional clock
I2ASYNC_FCLK	O	I2Async	I2Async functional clock
AXI_FCLK	O	AXI2OCP	AXI2OCP functional clock
ICECRUSHER_FCLK	O	ICE-Crusher	ICECrusher functional clock

3.2.1.2 Reset Distribution

Resets to the MPU subsystem are provided by the PRCM module and controlled by the clock generator module. There are as many reset signals as power domains. For details about power domains, see Section 3.3.2.1. Figure 3-4 shows the reset scheme of the MPU subsystem.

Figure 3-4. MPU Subsystem Reset Scheme

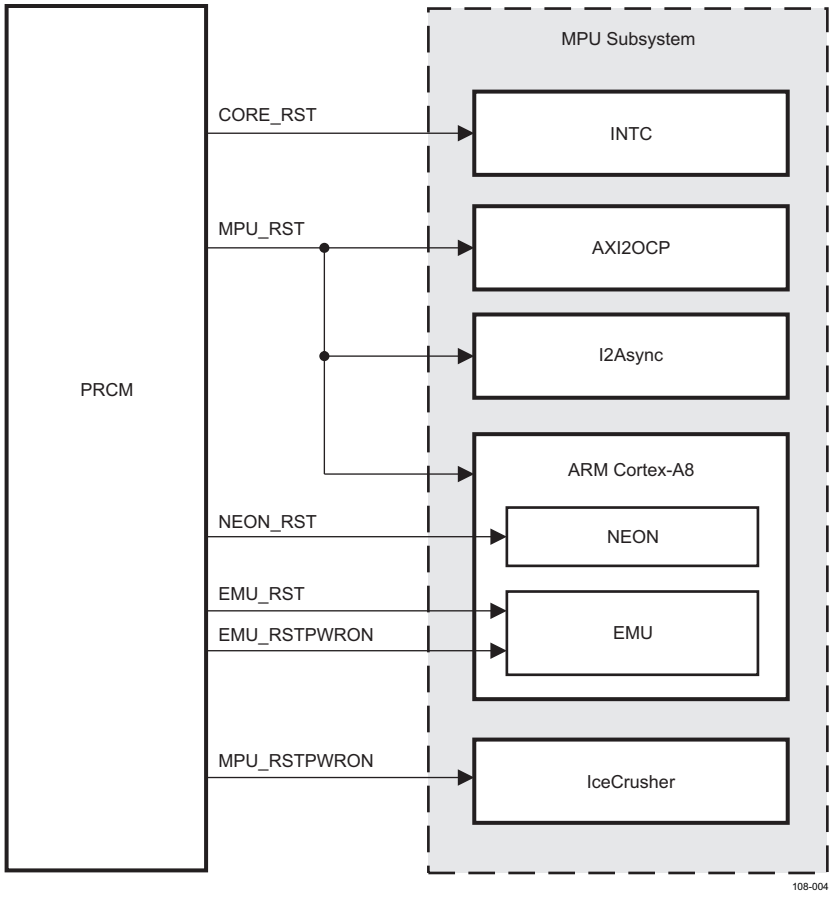


Table 3-2. MPU Subsystem Reset Signals

Signal Name	I/O	Interface	Comments
MPU_RST	I	PRCM	MPU power domain reset
Neon_RST	I	PRCM	Neon power domain reset
CORE_RST	I	PRCM	CORE power domain reset
MPU_RSTPWRON	I	PRCM	ICECrusher reset. It is active only on cold reset.
EMU_RST	I	PRCM	Emulation interconnect reset
EMU_RSTPWRON	I	PRCM	Emulation modules reset

For details about clocks, resets, and power domains, see , *Power, Reset, and Clock Management*.

3.2.2 ARM Subchip

3.2.2.1 ARM Overview

The public ARM Cortex-A8 processor incorporates the technologies available in the ARM7 architecture. These technologies include Neon for media and signal processing and Jazelle RCT for acceleration of realtime compilers, Thumb-2 technology for code density and the VFPv3 floating-point architecture.

For details, see the public ARM Cortex-A8 Technical Reference Manual.

3.2.2.2 ARM Description

3.2.2.2.1 Public ARM Cortex-A8 Instruction, Data, and Private Peripheral Port

The AXI bus interface is the main interface to the ARM system bus. It performs L2 cache fills and noncacheable accesses for instructions and data. The AXI interface supports 64-bit wide input and output data buses. It supports multiple outstanding requests on the AXI bus and a wide range of bus clock-to-core clock ratios. The bus clock is synchronous with the core clock.

See the public ARM Cortex-A8 Technical Reference Manual for a complete programming model of the transaction rules (ordering, posting, and pipeline synchronization) that are applied depending on the memory region attribute associated with the transaction destination address.

3.2.2.2.2 MPU Subsystem Features

Table 3-3 is a list of main functionalities of the ARM core supported in the MPU subsystem for the device. The MPU subsystem implements the ARM7™ instruction set architecture.

Table 3-3. ARM Core Key Features

Feature	Comment
ARM version 7 ISA	Standard ARM instruction set + Thumb-2, Jazelle RCT Java accelerator, and media extensions. Backward-compatible with previous ARM ISA versions.
L1 Icache and Dcache	16KB, 4-way, 64-byte cache line, and 128-bit interface. Note: L1 memories cannot be put in retention mode.
L2 Cache	The L2 cache and cache controller are embedded in the ARM core. For size, see Section 1.4, .
TLB	Fully associative and separate ITLB with 32 entries and DTLB with 32 entries
CoreSight ETM	The CoreSight ETM is embedded in the ARM core. The 4KB buffer (ETB) exists at the device level.
Branch target address cache	512 entries
Enhanced Memory Management Unit	Mapping sizes are 4KB, 64KB, 1MB, and 16MB. ARM MMU adds extended physical address ranges.
Neon	Enhances throughput for media workloads and VFP-Lite support
Tightly coupled memory	Not present
AXI Bus	Separate 64-bit input and 64-bit output data buses. The public ARM Cortex-A8 has a single AXI bus interface. The AXI interface is shared by instruction fetches, data accesses, peripheral accesses and PLE (on-chip preload engine, previously known as DMA) accesses. DMA is available.
Low interrupt latency	Interrupt request (IRQ) and fast interrupt request (FIQ) are directly connected to ARM and INTC is before L3.
Vectored Interrupt Controller Port	Not used
JTAG based debug	Supported through debug access port (DAP)
Trace support	Supported through trace port interface unit (TPIU)
External coprocessor	Not supported

For more information, see the public *ARM Cortex-A8 Technical Reference Manual*.

3.2.2.3 Clock, Reset, and Power Management

3.2.2.3.1 Clocks

Table 3-4 shows the ARM functional clock. For configuration, see , *Power, Reset, and Clock Management*.

Table 3-4. MPU Subsystem Clock Signal

Signal Name	I/O	Interface	Comments
ARM_FCLK	I	MPU Clock Generator	Functional clock

3.2.2.3.2 Reset

Table 3-5 lists the resets for the ARM. They include the power domain reset, Neon_RST, which resets the Neon module of the ARM subchip only, and MPU_RST, which resets the rest of the ARM subchip and the AXI2OCP and I2Async bridges.

Table 3-5. ARM Reset Signals

Signal Name	I/O	Interface	Comments
MPU_RST	I	PRCM	MPU power domain reset
Neon_RST	I	PRCM	Reset Neon only
EMU_RST	I	PRCM	Emulation interconnect reset
EMU_RSTPWRON	I	PRCM	Emulation modules reset

3.2.2.3.3 Power Management

For details, see Section 3.3.2.

3.2.3 AXI2OCP and I2Async Bridges

3.2.3.1 Bridges Overview

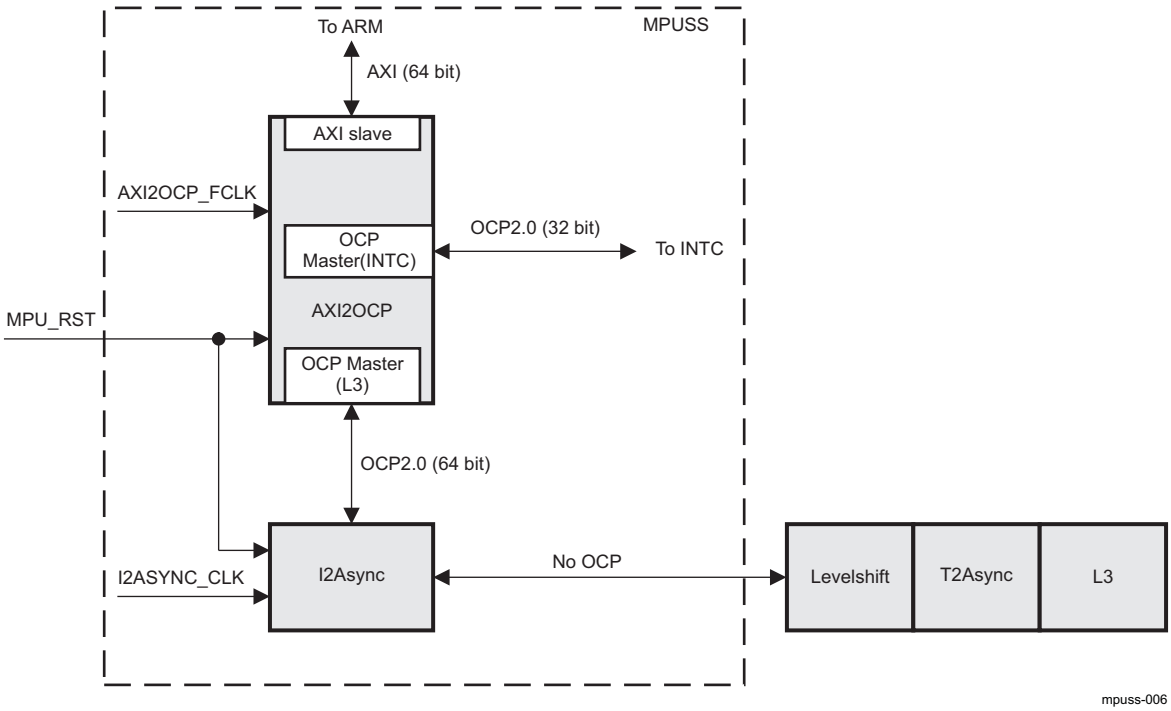
The AXI2OCP bridge connects the AXI bus on the ARM MPU to the OCP native L3 interconnect and INTC. It converts between AXI and OCP protocols and maintains a mapping of AXI tags to the OCP thread ID. This bridge is responsible for some minimal address decoding to determine where to forward requests between L3 and INTC.

Bridging to the L3 is accomplished through an asynchronous interface involving the I2Async and T2Async modules. The I2Async module in the MPU subsystem has an OCP port that is asynchronously transferred to the T2Async module and routed to the L3. T2Async is outside the MPU subsystem.

NOTE: The interface between I2Async and T2Async is not an OCP protocol.

Figure 3-5 is an overview of the AXI2OCP and the L3 bridges.

Figure 3-5. Bridges Overview



3.2.3.2 AXI2OCP Description

The AXI2OCP bridge key features are:

- Connection to the L3 interconnect through a 64-bit OCP port. Address bus is 32-bit.
- Connection to the INTC through a 32-bit OCP port (only single transactions are supported)
- Support of single-request-multiple-data (data handshaking) burst mode to pipeline requests to L3. Bursts with width less than 64 bits are converted as single requests on L3. This can affect system performance. The INTC does not support burst transfers
- Support of multiple outstanding requests
- Remapping of the AXI tags to five OCP threads:
  - Instruction fetch (Thread_IR)
  - Cacheable data read (Thread_CR)
  - Cacheable data write (Thread_CW)
  - Noncacheable data read (Thread_DR)
  - Noncacheable data write (Thread_DW)
- Interface to the L3 through the asynchronous bridge (I2Async)
- Emulation and support of boot-mode translation
- Translation of exclusive accesses to nonexclusive read/write in the bridge
- Boot mode address translation
- Force write nonposted support
- Debug related: Ready fail, force ready
- Interconnect attach neutralizer
- Narrow burst support
- Time-out counter for nonresponsive slaves on OCP

3.2.3.3 Clocks, Reset, and Power Management

3.2.3.3.1 Clocks

Table 3-6 lists the bridge functional clocks.

Table 3-6. Bridge Clock Signals

Signal Name	I/O	Interface	Comments
AXI2OCP_FCLK	I	MPU Clock Generator	AXI2OCP functional clock
I2ASYNC_FCLK	I	MPU Clock Generator	I2Async functional clock

3.2.3.3.2 Reset

Table 3-7 lists the bridge reset. It is a power domain reset that also resets the ARM.

Table 3-7. MPU Subsystem Reset Signal

Signal Name	I/O	Interface	Comments
MPU_RST	I	PRCM	MPU power domain reset

3.2.3.3.3 Power Management

For details, see Section 3.3.2.

3.2.4 Interrupt Controller

For information, see Chapter 8, *Interrupt Controllers*.

3.2.4.1 Clocks

Table 3-8 lists the INTC clocks.

Table 3-8. Bridge Clock Signals

Signal Name	I/O	Interface	Comments
MPU_INTC_FCLK	I	MPU Clock Generator	INTC functional clock
MPU_INTC_ICLK	I	OCP Clock	INTC interface clock

3.2.4.2 Reset

Table 3-9 lists the reset of the INTC. It is a power domain reset that resets the entire CORE power domain. For details, see Chapter 8, *Interrupt Controller* and , *Power, Reset, and Clock Management*.

Table 3-9. MPU Subsystem Reset Signal

Signal Name	I/O	Interface	Comments
CORE_RST	I	PRCM	CORE power domain reset

3.2.4.3 Power Management

See Chapter 8, *Interrupt Controller*, and , *Power, Reset, and Clock Management*.

3.3 MPU Subsystem Functional Description

3.3.1 Interrupts

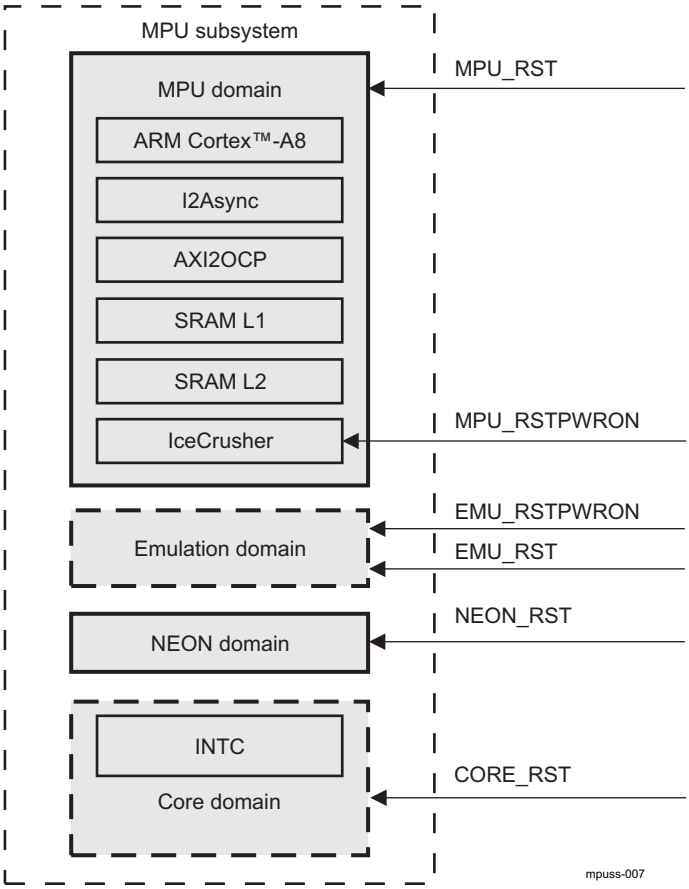
The MPU INTC is connected to the MPU through the AXI2OCP bridge. It runs at half-processor speed. The INTC prioritizes all service requests from the system peripherals and generates an IRQ or an FIQ to the MPU, depending on the INTC programming. The INTC handles only interrupts directed to the MPU subsystem. A maximum of 96 requests can be steered/prioritized as MPU FIQ or IRQ interrupt requests. For details, see Chapter 8, *Interrupt Controller*.

3.3.2 Power Management

3.3.2.1 Power Domains

The MPU subsystem is divided into five power domains controlled by the PRCM module, as shown in Figure 3-6. The EMU and CORE power domains are not fully embedded in the MPU subsystem.

Figure 3-6. MPU Subsystem Power Domain Overview



Power-management requirements at the device level govern power domains for the MPU subsystem.

The device-level power domains are directly aligned with voltage domains and thus can be represented as a cross reference to the different voltage domains. Table 3-10 shows the power domains of the MPU subsystem and the modules inside it.

Table 3-10. Overview of the MPU Subsystem Power Domain

Functional Power Domain	Physical Power Domain per System/Module
MPU subsystem domain	ARM, AXI2OCP, I2Asynch Bridge, ARM L1 and L2 periphery logic and array, ICECrusher, ETM, APB modules

Table 3-10. Overview of the MPU Subsystem Power Domain (continued)

Functional Power Domain	Physical Power Domain per System/Module
MPU Neon domain	ARM Neon accelerator
CORE domain	MPU INTC
EMU domain	EMU (ETB, DAP)

NOTE: L1 and L2 array memories have separate control signals to the MPU subsystem; thus are directly controlled by the PRCM module.

For information about the physical power domains and the voltage domains, see , *Power, Reset, and Clock Management.*

3.3.2.2 Power States

Each power domain can be driven by the PRCM module in four power states, depending on the functional mode required by the user.

Table 3-11. MPU Power States

Power State	Logic Power	Memory Power	Clocks	Memory State Retention
ACTIVE	On	On or Off	On (at least one clock)	All
INACTIVE	On	On or Off	Off	All
RETENTION	On or Off	On or Off	Off (all clocks)	All or part
OFF	Off	Off	Off (all clocks)	None

For each power domain, the PRCM module manages all transitions by controlling domain clocks, domain resets, domain logic power switches, memory power switches, and memory retention. The MPU DPLL internally synchronizes the internal clocks, resets, and switches.

3.3.2.3 Power Modes

MPU DPLL power modes:

The PRCM.CM_AUTOIDLE_PLL_MPU[2:0] AUTO_MPU_DPLL bit field lets the MPU DPLL be put in an auto-idle mode when set to 0x1. In this mode, the MPU DPLL is automatically put in low power stop mode when the MPU clock is not required. It is also restarted automatically. Table 3-12 describes the power modes of the MPU DPLL in auto-idle or manual mode. The manual modes (locked and low-power bypass) can be configured by the PRCM.CM_CLKEN_PLL_MPU[2:0] EN_MPU_DPLL bit field. The status of the MPU DPLL clock activity can be checked with the PRCM.CM_IDLEST_PLL_MPU[0] ST_MPU_CLK bit.

Table 3-12. MPU DPLL Power Modes

Mode	System Input Clock	Clock Output	DPLL Power State	Condition
Locked	ON	ON	ON	Software request (manual) or MPU wakes up (auto)
Low-power bypass	ON	ON	ON	Software request (manual) or on global reset release (auto)
Stop low power	OFF	OFF	ON	MPU is RET or OFF (auto).
OFF	OFF	OFF	OFF	Device is OFF (auto).

The MPU DPLL power domain is switched off automatically by the PRCM module only when the device enters OFF mode.

MPU subsytem power modes:

The major part of the MPU subsystem belongs to the MPU power domain. Modules in this power domain can be off when the ARM processor is in OFF or standby mode. IDLE/WAKEUP control is managed by the clock generator block, but initiated by the PRCM module. The MPU standby status can be checked with the PRCM.CM_IDLEST_MPU[0] ST_MPU bit.

For the MPU to be on, the core (referred to here as the device core) power must be on.

Device power management does not allow INTC to go to OFF state when the MPU domain is on (active or one of the retention modes).

The Neon core is in independent power off mode when not in use. Enabling and disabling of Neon can be controlled by software.

The MPU retention modes are described in Table 3-13.

Table 3-13. MPU Retention Modes

Name	ARM Logic	L1	L2
Dormant	OFF	OFF	RET
RET	RET	RET	RET

Table 3-14 lists the supported operational power modes. All other combinations are illegal. The ARM L2, Neon, and ETM/debug can be powered up/down independently. The APB/ATB ETM/Debug column refers to all three features: ARM emulation, trace, and debug.

Table 3-14. MPU Subsystem Operation Power Modes

Mode	MPU and ARM Core Logic	ARM L2 RAM	Neon	OMAP Core INTC	APB/ATB ETM/Debug	Comments
1	Active	Active	Active	Active	Disabled or enabled	Functional active run mode (ETM enabled mode when emulation/debug required. Production devices should have ETM disabled).
2	Active	Active	OFF	Active	Disabled or enabled	Functional active run mode. Neon disabled through software; Neon is internally clock-gated.
3	Active	RET	Active	Active	Disabled or enabled	Do not use; see (1).
4	Active	RET	OFF	Active	Disabled or enabled	Do not use; see (1).
5	Active	OFF	Active	Active	Disabled or enabled	Active mode; L2 is off. Controlled through software to the PRCM module. L2 context save and restore required or L2 flush.
6	Active	OFF	OFF	Active	Disabled or enabled	Active mode; L2 is off. Controlled through software to the PRCM module. L2 context save and restore required or L2 flush.

(1) The L2 can be put in retention mode regardless of other voltage domain states. The combination of Cortex Logic active and L2 in retention mode (modes 3 and 4) is legal, but results in incorrect execution of instructions with referencing data from L2. This combination must not be used.

Table 3-14. MPU Subsystem Operation Power Modes (continued)

Mode	MPU and ARM Core Logic	ARM L2 RAM	Neon	OMAP Core INTC	APB/ATB ETM/Debug	Comments
7	OFF	RET	OFF	OFF	Disabled or enabled	Lowest power sleep mode (dormant mode); L2 is in retention. Controlled through software to the PRCM module. ARM core and L1 context save and restore required or L1 flush.
8	Standby	Active	Standby	Active	Disabled or enabled	Standby mode. StandbyWFI (wait for interrupt) controlled to put into standby and wakeup through interrupt.
9	Standby	Active	OFF	Active	Disabled or enabled	Standby mode. StandbyWFI controlled to put into standby and wakeup through interrupt when Neon is off.
10	Standby	RET	Standby	Active	Disabled or enabled	Standby mode (retention mode). StandbyWFI controlled to put into standby and wakeup through interrupt when L2 is in retention.
11	Standby	RET	OFF	Active	Disabled or enabled	Standby mode (retention mode). StandbyWFI controlled to put into standby and wakeup through interrupt when L2 is in retention and Neon is off.
12	Standby	OFF	Standby	Active	Disabled or enabled	Standby mode. StandbyWFI controlled to put into standby and wakeup through interrupt when L2 is off.
13	Standby	OFF	OFF	Active	Disabled or enabled	Standby mode. StandbyWFI controlled to put into standby and wakeup through interrupt when L2 and Neon are off.
14	OFF	OFF	OFF	OFF	Disabled or enabled	Power-down mode

In any mode where the MPU or Neon power domains are active, the MPU DPLL clocks must be active (modes 1, 3, and 5 require active clocks from the DPLL, while modes 7 and 8 do not).

Thus, the MPU DPLL must be in one of the following states:

- Locked
- Low-power bypass: inclk = on, clkout = on, power = on

When the MPU DPLL is not providing clocks, the MPU subsystem must be in a power mode where the MPU power domain, Neon power domain, debug power domain, and INTC power domain are in standby, RETENTION, or OFF state. The states of the MPU DPLL can be:

- Locked
- STOP low power
- OFF

CAUTION

The L2 can be put into retention mode regardless of other voltage domain states. The combination of ARM Logic active and L2 in retention mode (modes 3 and 4) is legal, but results in incorrect execution of instructions with referencing data from L2. This combination must not be used.

3.3.2.4 Transitions

Table 3-15 describes allowable transitions from power modes described in Table 3-14. For example, a transition from mode 13 to mode 4 is allowed, but the reverse is not true, because the L2 RET to OFF is illegal.

Any mode change that requires state saving or flush must be serialized. For example, L2 RET does not require state saving, so it can happen at the same time as power-down Neon. L2 off and Neon off cannot happen at the same time, because L2 flush and Neon state saving must be serialized. Standby mode can enter from active mode only by executing the WFI instruction.

Table 3-15. Power Mode Allowable Transitions

	To Power Mode													
From	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	Y	Y	Y	Y	Y			Y						Y
2	Y	Y	Y	Y		Y			Y		Y			Y
3	Y	Y	Y	Y			Y			Y				
4	Y	Y	Y	Y			Y				Y			
5	Y		Y	Y	Y	Y	Y			Y		Y		Y
6		Y	Y	Y	Y	Y	Y				Y		Y	Y
7	Y	Y	Y	Y			Y							
8	Y		Y					Y		Y				
9	Y	Y	Y	Y					Y		Y			
10	Y		Y					Y		Y				
11	Y	Y	Y	Y					Y		Y			
12	Y		Y		Y							Y		
13	Y	Y	Y	**Y**	Y	Y			Y		Y		Y	
14	Y	Y	Y	Y	Y	Y	Y							Y

For more information about clocks, reset, power management, and wake-up events for the MPU subsystem, see , *Power, Reset, and Clock Management*.

3.4 MPU Subsystem Basic Programming Model

For detailed descriptions of registers used for MPU configuration, see , *Power, Reset, and Clock Management*, and Chapter 8, *Interrupt Controller*.

3.4.1 Clock Control

For clock configuration settings, see , *Power, Reset, and Clock Management*.

3.4.2 MPU Power Mode Transitions

The following subsections describe transitions of different power modes for the MPU power domain:

- Basic power on reset
- MPU to standby mode
- MPU out of standby mode
- MPU power on from a powered-off state

3.4.2.1 Basic Power-On Reset

The following power-on reset (POR) sequence applies to initial power-up and wakeup from device off mode:

1. Reset DPLL, supply reference clock, program the MPU DPLL in applicable DPLL mode to generate clocks for MPU subsystem modules. This is controlled solely by the PRCM module.
2. Reset the INTC (CORE_RST) and the MPU subsystem modules (MPU_RST). The clocks must be active during MPU reset and CORE reset.

3.4.2.2 MPU to Standby Mode

The following MPU to standby mode sequence applies to initial power-up and wakeup from device off mode:

1. The ARM core initiates entering standby through software only (CP15 - WFI).
2. MPU modules requested internally by MPU subsystem to enter idle, after ARM core standby detected.
3. MPU is in standby output asserted for the PRCM module (all outputs ensured to be at reset values).
4. The PRCM module requests the INTC to enter idle mode. Acknowledge from INTC goes to the PRCM module.
5. The PRCM module starts to shut down clocks through DPLL programming.

NOTE: The INTC SWAKEUP output is a pure hardware signal to the PRCM module for the status of its IDLE request and IDLE acknowledge handshake.

NOTE: In debug mode, ICECrusher can prevent the MPU subsystem from entering IDLE mode.

3.4.2.3 MPU Out of Standby Mode

The following MPU out of standby mode sequence of operation applies to initial power-up and wakeup from device off mode:

1. PRCM must start clocks through DPLL programming.
2. Detect active clocking through status output of DPLL.
3. Initiate an interrupt through the INTC to wake up the ARM core from STANDBYWFI mode.

3.4.2.4 MPU Power-On from a Powered-Off State

1. To minimize the peaking of current during power up, DPLL power on, MPU power on, Neon power on, and Core power on (INTC) should follow the ordered sequence per power switch daisy chain.

NOTE: Before the MPU can be reset, the CORE power domain must be on, and reset, with the DPLL clocks on.

2. Follow the reset sequence as described in Section 3.4.2.1, *Basic Power-On Reset*.

3.4.3 Neon Power Mode Transition

Because of the hardware sleep dependency between Neon and the MPU domain, when the Neon power domain transition is configured to automatic hardware-supervised mode (the CM_CLKSTCTRL_Neon[1:0] CLKTRCTRL_Neon bit field is set to 0x3), it cannot transition to idle mode unless the MPU goes to standby mode. The MPU domain must also be configured in automatic hardware-supervised mode (the CM_CLKSTCTRL_MPU[1:0] CLKTRCTRL_MPU bit field must be set to 0x3) for the Neon power domain transition to occur.

3.4.4 ARM Programming Model

For the complete programming model, see the public *ARM Cortex-A8 Technical Reference Manual*.

Power, Reset, and Clock Management

This chapter describes power, reset, and clock management.

Topic **Page**

4.1 PRCM Introduction to Power Management

This introduction contains the following information:

- Requirement and goal of power management in mobile devices
- State-of-the-art power-management techniques for maximizing battery life for mobile devices
- Essential architectural building blocks for power management
- Overview of the device power-management architecture

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

4.1.1 Goal of Power Management

Power management (efficient use of the available limited battery resources of a mobile device) is one of the most important design aspects of any mobile system. It imposes strong control over limited available power resources to ensure they function for the longest possible amount of time.

The device architecture ensures maximum performance for user satisfaction (audio/video support) while offering versatile power-management techniques for maximum design flexibility, depending on application requirements.

4.1.2 Architectural Blocks for Power Management

The processor does not support Voltage and Power domains. As a result, there is one voltage supply to the device processors and peripherals. There is also no support for switching power on or off to individual power domains. Although the power domain terminology is retained, the state of power to all domains is either ON when the device is supplied with external power, or OFF when the external power source to the device is off.

The device does support dynamic clock gating for power management through clock domains. A clock domain is a group of modules or subsections of the device that share a common clock.

4.1.2.1 Clock Domain

A clock domain is a group of modules fed with the same gated clock signal (see Figure 4-1). By gating the clock to each domain, it is possible to cut a clock to a group of inactive modules to lower their active power consumption. Thus, a clock domain allows control of dynamic power consumption by the device.

Figure 4-1. Generic Clock Domain

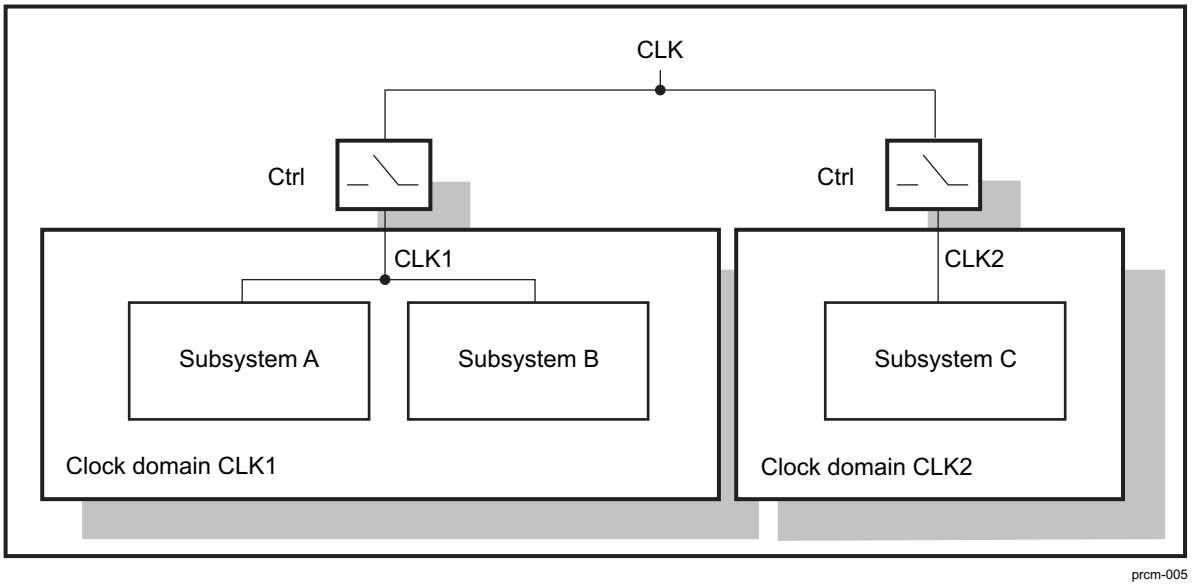


Table 4-1 lists the two possible states of the clock domain.

Table 4-1. States of a Clock Domain

State	Description
Active	The domain clock is running.
Idle	The domain clock is stopped or gated.

4.1.2.2 Power Domain

Some members of the device family support independent control of power to subsections of the device (called Power Domains) via independent power switches. While retaining some of the Power Domain terminology, this device does not support independent Power Domain control. The state of all Power Domains in the device is either ON when the device is externally supplied with power or OFF when the device is completely powered down. In effect, the entire device is a single power domain.

4.1.2.3 Voltage Domain

Whereas some members of the device family have subsystems grouped into voltage domains that can be supplied by independently scalable voltage regulators, the device has its core subsystems all in a single voltage domain. Therefore, the voltage domain concept does not apply to this device.

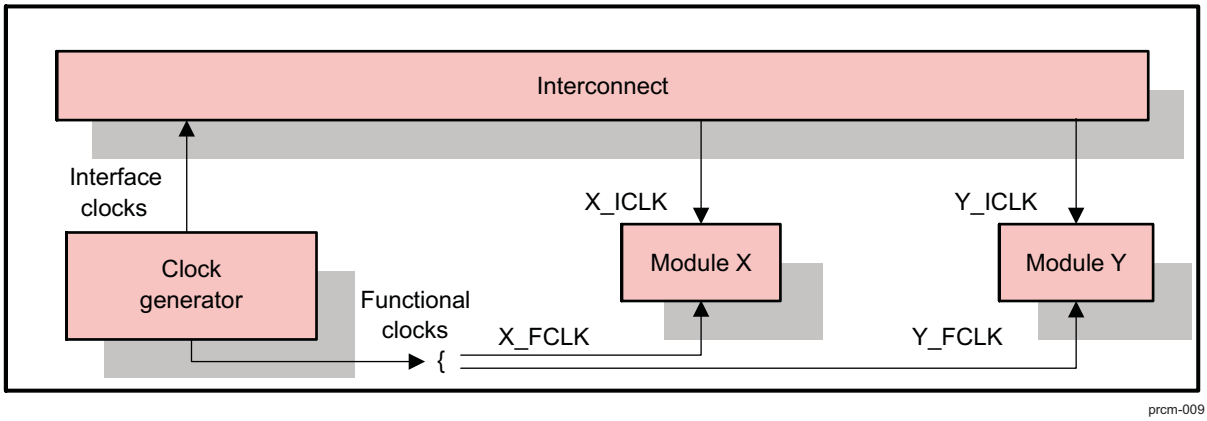
4.1.3 Device Power-Management Architecture

The device supports a clock distribution and control architecture, which is described in the following sections.

4.1.3.1 Module Interface and Functional Clocks

The clocks delivered to the modules in the device are divided into two categories: interface clocks and functional clocks (see Figure 4-2).

Figure 4-2. Functional and Interface Clocks



Interface clocks have the following characteristics:

- They ensure proper communication between any module/subsystem and the interconnect.
- In most cases, they supply the module interface and registers.
- A typical module has one interface clock, but it can have several.
- They are synchronous across the entire device, because the interconnect fabric is itself fully synchronous.
- Interface clock management is done at the device level.
- From the standpoint of the power, reset, and clock management (PRCM) module, an interface clock is distributed through the device interconnect and is identified with an _ICLK suffix.

Functional clocks have the following characteristics:

- They supply the functional part of a module or a subsystem.
- Each module can have several functional clocks, or none at all. A module may or may not require its functional clocks to be active (nonidle).
- Several modules can share the same functional clock signals, but the branches of the clock tree are not balanced between the modules.
- From the PRCM standpoint, a functional clock is directly distributed to the related modules through a dedicated clock tree. It is identified with an _FCLK suffix.

NOTE: At the module level, the interface clocks are always fed by the interface clock outputs of the PRCM. The functional clocks are fed either by a PRCM functional clock output or a PRCM interface clock output. In the latter case, the functional and interface clocks of the module inherit capabilities (autoidle features) from the PRCM interface clock (see Section 4.7, *Clock Manager Functional Description*).

4.1.3.2 Autoidle Clock Control

The device supports an autoidle clock control scheme for the module interface clocks. With this control scheme, PRCM can automatically activate and deactivate the interface clock of any device module, depending on its operating mode. This scheme executes under hardware control and is transparent to the software. This scheme identifies two module types in the device: the target and the initiator modules, or subsystems.

NOTE: The functional clocks do not have the autoidle clock scheme, and the software must gate the functional clock of each module when it is not needed.

Initiator

An initiator can generate bus transactions (read, write, etc.) toward targets. It is considered to be active when it generates transactions. If it enters standby mode, it stops generating transactions. Because most initiators also support a target port for configuration purposes, they are both targets and initiators. Some examples of initiators are processors, direct memory access (DMA), and memory management unit (MMU).

Target

A target is a passive module that can process bus transactions (that is, it reads/writes to memory-mapped registers). It is considered to be active when its interface clocks and some or all of its functional clocks are available so it can accept incoming transactions. A target can be put in idle mode by the PRCM, and in this mode its interface clock can be gated at any time. An idle module can still receive its functional clocks and generate interrupts and DMA requests. It can also generate asynchronous wake-up requests, if it is wakeup-capable.

Active, Idle, and Standby Modes

The PRCM module handles automatic clock control differently for initiator and target modules.

For the initiator module, the following hardware handshake scheme is employed:

1. The initiator, when switching from active to idle mode, signals its status to the PRCM.
2. The PRCM cuts off the interface clock to the initiator module.
3. When the initiator module must reactivate, it signals the PRCM, which reactivates its functional and interface clocks.

For the target module, the following hardware handshake scheme is employed:

1. When the PRCM confirms that the target module satisfies the idle conditions, it signals the target module.
2. The target module acknowledges the idle request of the PRCM, depending on its idle mode internal settings (for details about idle mode settings, see the chapter in the technical reference manual for the corresponding device module):

- If the module is set to smart-idle mode, it terminates its current operations, and then acknowledges the idle request to the PRCM.
- If the module is set to force-idle mode, it acknowledges immediately, regardless of its current state. Because pending transfers, interrupts, and DMA requests can potentially be lost, special software care must be taken.
- If the module is set to no-idle mode, it does not acknowledge the idle request. This forces the PRCM to maintain the clock active.

3. The PRCM cuts off the module clocks.
4. The target module can be wakened by the PRCM when its wake-up conditions are satisfied (wake-up event). It activates the module clocks, and then signals the wakeup.
5. The target module acknowledges the wake-up request.
6. Some target modules can support wake-up capability. They can explicitly request the PRCM to activate their clock.

This automatic clock control ensures reduced dynamic power consumption of the device without any associated software overhead.

4.2 PRCM Overview

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power saving considerations, ensure that clocks for unavailable or unused modules are gated.

4.2.1 Introduction

The power-management framework of the device significantly reduces dynamic power consumption to extend the life of the battery in the end-product. This framework incorporates support for state-of-the-art power-management techniques. It ensures optimal device operation with significantly reduced power consumption. The PRCM module, which is the enhanced power-management subsystem of the device, is the central control module for the clock, reset, and power-management signals in the device.

The device has the following features to support the different power-management techniques:

- Clock tree with selective clock-gating conditions
- Hardware-controlled reset sequencing management
- Support for hardware-controlled autogating of module clocks
- Support for low-power device standby mode

The PRCM module is the centralized management module for the power, reset, and clock control signals. It interfaces with all the components on the device for power, clock, and reset management through power-control signals. It integrates enhanced features to allow the device to adapt energy consumption dynamically according to changing application and performance requirements.

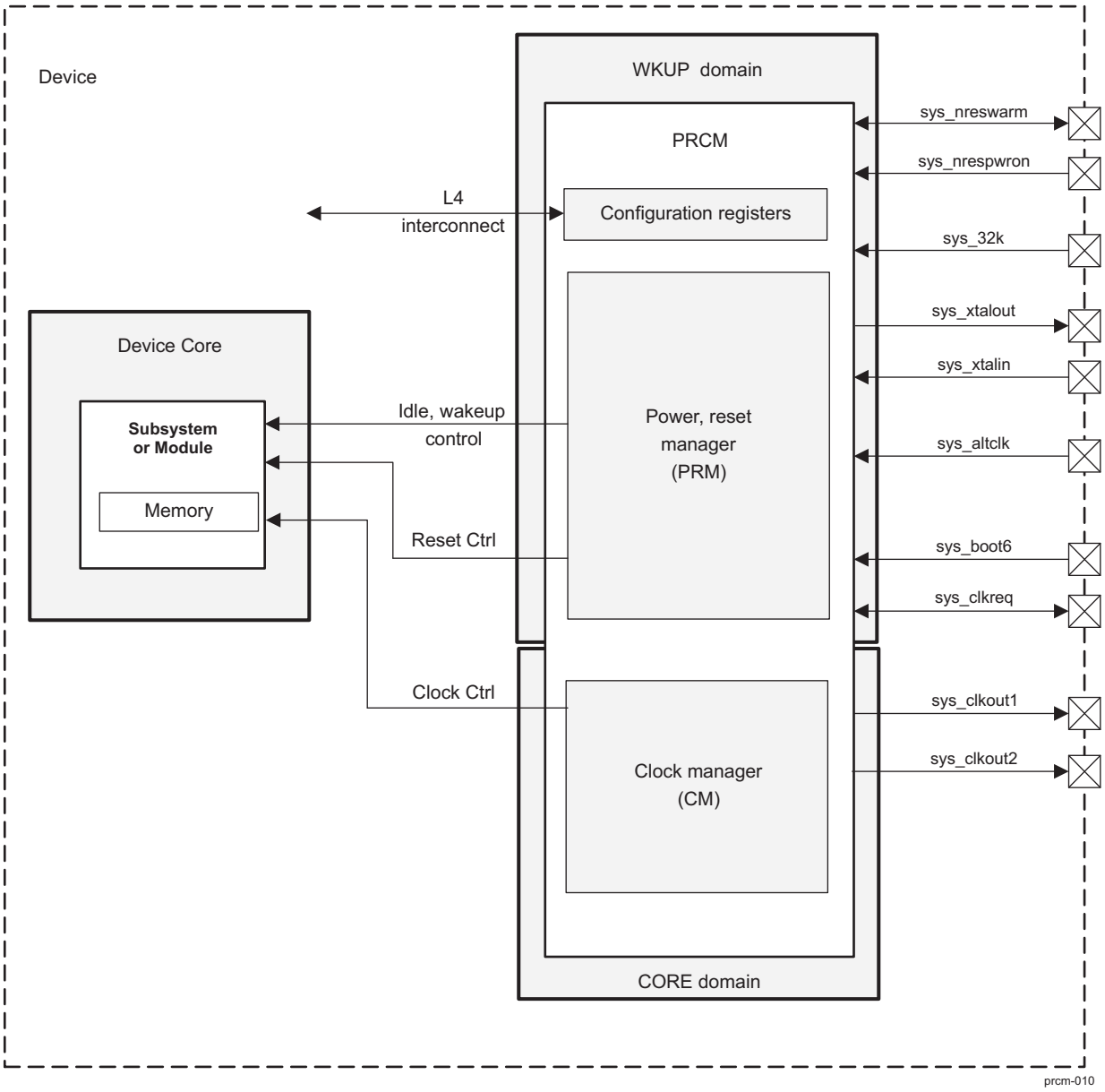
The PRCM module is composed of two main entities:

- Power reset manager (PRM): Handles the power, reset, wake-up management, and system clock source control (oscillator)
- Clock manager (CM): Handles the clock generation, distribution, and management

The PRCM is fully configurable through its L4 interface port.

Figure 4-3 is an overview of the PRCM module and its internal connections with a generic power domain (a group of modules with related functionality).

Figure 4-3. PRCM Overview



4.2.2 PRCM Features

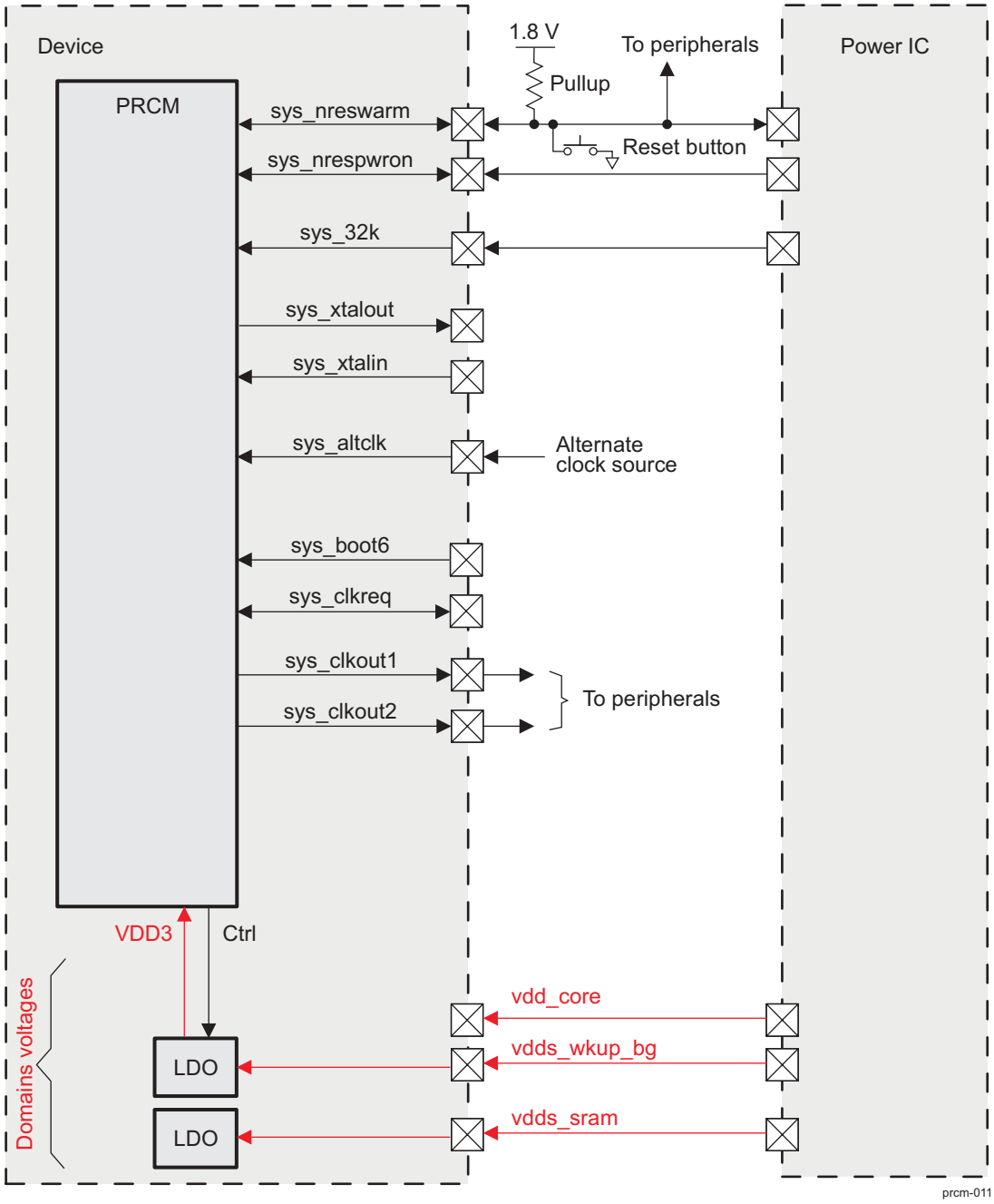
The PRCM module includes the following features:

- Handling proper idle/wake-up procedures
- Allowing both software and partial hardware control
- Monitoring and handling wake-up events
- Controlling system clock/reset input sources
- Managing and distributing clocks and resets with high granularity
- Handling power-up sequences
- Debug and emulation features

4.3 PRCM Environment

The PRCM module receives the external reset, clock, and power signals. Figure 4-4 shows the interface of the PRCM with external reset, clock, and power sources.

Figure 4-4. PRCM Functional External Interface (Detailed View)



NOTE: In the remainder of this chapter, the term "power IC" refers to a peripheral power source IC that is interfaced with the device.

The following sections describe the interfaces for the external clock, reset, and power sources.

4.3.1 External Clock Signals

The device has three external clock inputs: low frequency (sys_32k), high frequency (sys_xtalin), and an optional clock (sys_altclk). The device has two configurable clock outputs: sys_clkout1 and sys_clkout2. Figure 4-5 shows the external clock signals of the PRCM module. Table 4-2 lists the external clock signals, I/Os, and module reset values.

Figure 4-5. External Clock Interface

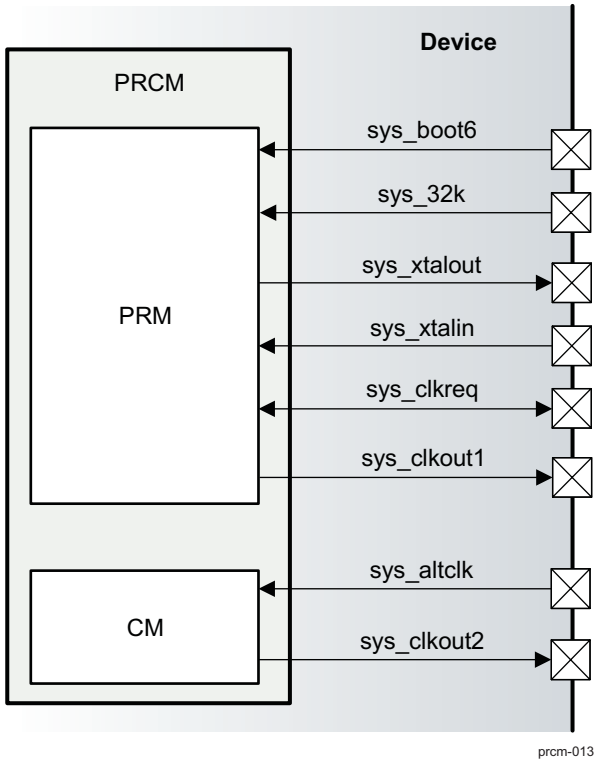


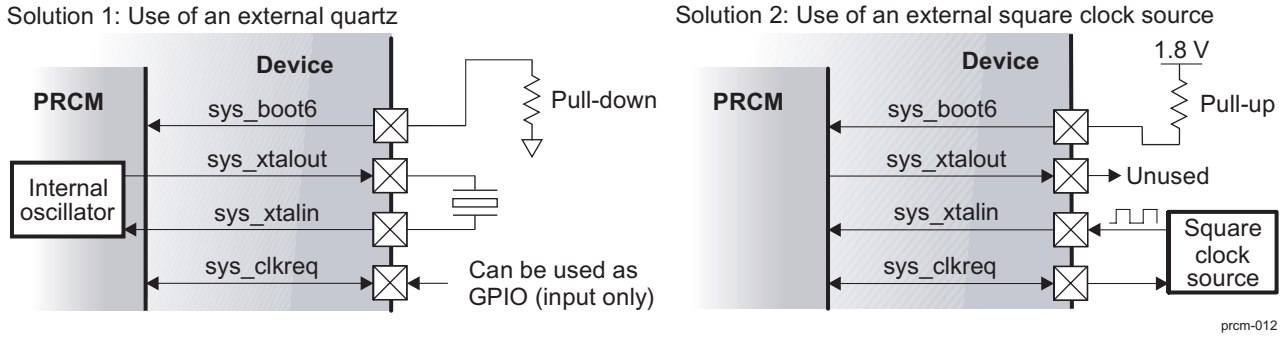
Table 4-2. External Clock Signal Descriptions

Signal Name	I/O[1]	Description	Module Reset Value
sys_boot6	I	Boot oscillator mode control	Unknown
sys_32k	I	32-kHz clock input	Unknown
sys_xtalout	O	Output of oscillator	0
sys_xtalin	I	Main input clock. Crystal oscillator clock (only at 26 MHz) or CMOS digital clock (at 26 MHz).	Unknown
sys_clkreq	I/O	Clock request to/ from device for system clock	1
sys_clkout1	O	Configurable output clock 1	0
sys_altclk	I	Alternate clock source selectable for USB (48 MHz) or NTSC/PAL (54 MHz)	Unknown
sys_clkout2	O	Configurable output clock 2	0

(1) I = Input, O = Output

Figure 4-6 shows the PRCM external clock sources.

Figure 4-6. PRCM External Clock Sources



The system clock source can be either of the following:

- Internal oscillator with crystal connected between sys_xtalin and sys_xtalout
- A CMOS digital clock that arrives at the sys_xtalin pin

In the first option, the sys_clkreq signal is used in input mode to control sys_clkout1 and the internal clock oscillator. In the second option, the signal is used in output mode to request the external system clock.

An external pulldown or pullup tied on sys_boot6 determines whether the internal oscillator is used or an external clock source is supplied, respectively.

CAUTION

Only one clock source option can be used at a time.

An alternate clock input (sys_altclk) provides a precise clock source for 54 MHz, 48 MHz, or other frequencies (for example, 59 MHz or 49.04 MHz for VDAC).

For more information about external clock signals, see Section 4.7.5 , *External Clock Controls*.

4.3.2 External Reset Signals

The device supports two reset signals: power-on (sys_nrespwron) and warm reset (sys_nreswarm).

sys_nrespwron is asserted at power up to reset the full logic in the device. sys_nreswarm can be activated at any time by an external device or an external reset push-button action (see Figure 4-4) to cause a global warm reset event.

Because sys_nreswarm is bidirectional, it can also be used to drive a reset of external devices. Any global warm reset source (for example, a push-button) causes sys_nreswarm to be driven out and maintained for a limited length of time at the boundary of the device. In this way, the device and its related peripherals are properly reset together.

The sys_nrespwron assertion also causes the sys_nreswarm assertion.

Figure 4-7 shows the external reset signals. Table 4-3 lists the external reset signals, I/Os, and module reset values.

Figure 4-7. External Reset Signals

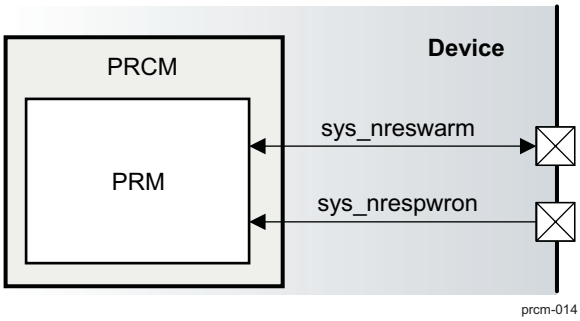


Table 4-3. External Reset Signals Description

Signal Name	I/O[1]	Description	Module Reset Value
sys_nreswarm	I/O	Warm-boot reset	1
sys_nrespwron	I	Power-on reset	Unknown

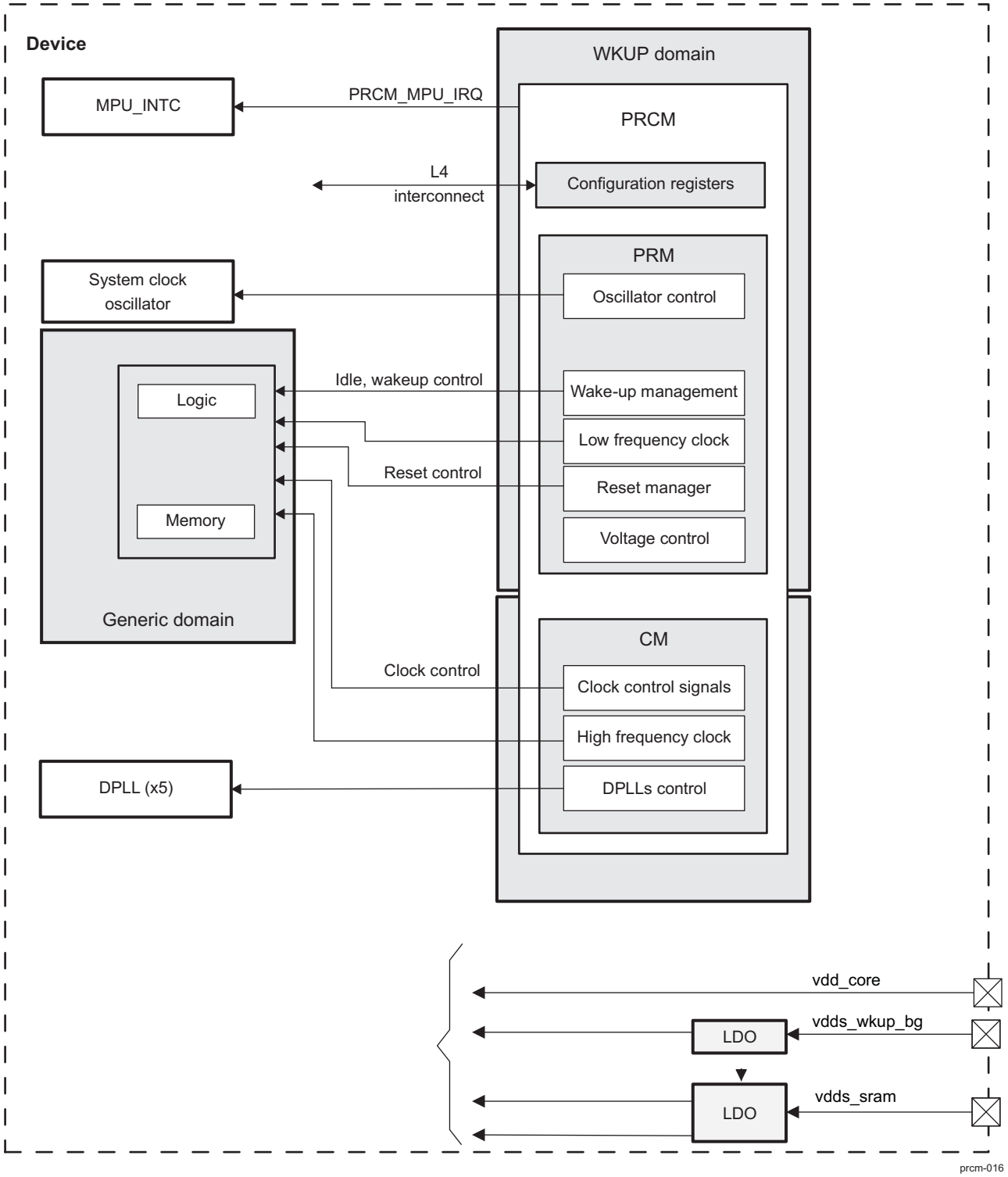
(1) I = Input, O = Output

4.4 PRCM Integration

The PRCM internal registers can be accessed for configuration and controlled through the WKUP L4 interconnect. In addition to the L4 interconnect, the PRCM internal module interface contains the following:

- A set of signals for idle/wake-up control for each module
- Clocks and reset signals
- Interrupt to the MPU subsystem interrupt controller (INTC)
- Digital phase-locked loop (DPLL) control commands for recalibration and bypass
- System clock oscillator control for device-level sleep/wake-up transitions

Figure 4-8. PRCM Integration



4.4.1 Power-Management Scheme, Reset, and Interrupt Requests

4.4.1.1 Resets

The PRM and CM modules are reset by independent reset signals (see Table 4-4).

Table 4-4. PRCM Reset Signals

PRCM Subsystem	Reset Signal
PRM	PRM_RSTPWRON
CM	CM_RSTPWRON_RET

The PRM module is reset by the cold reset signal PRM_RSTPWRON. The CM module is reset by assertion of the CM_RSTPWRON_RET signal.

The CM logic is reset on:

- Any global cold reset

The PRM logic is reset on:

- Any global cold reset

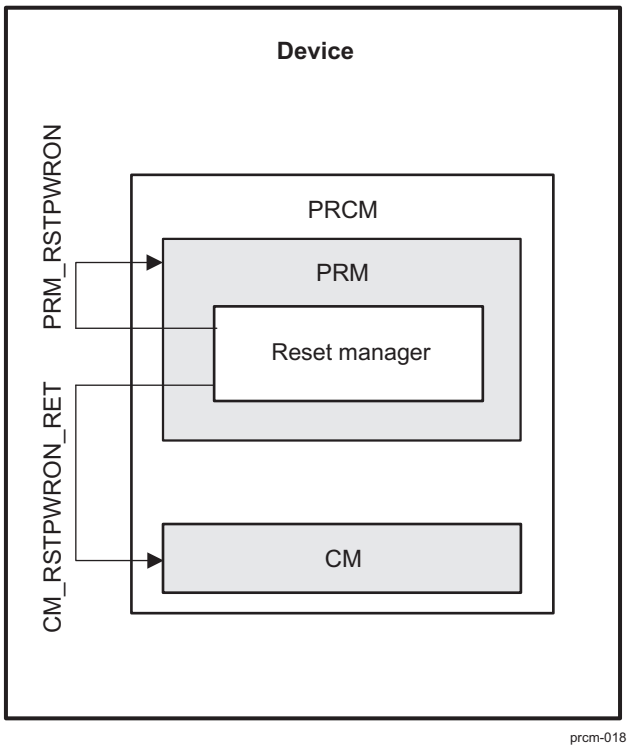
CM and PRM registers that are sensitive to a warm reset are also reset when a global warm reset occurs. However, the CM and PRM logic is not reset.

NOTE: At global cold reset:

- Only the device finite state-machine (FSM) in the PRM is operating on the 32-kHz clock, and it is released from reset on the release of the global reset.
- PRM logic operates on the system clock and is released from reset on release of the reset PRM_RSTPWRON.

Figure 4-9 shows the PRCM reset signals.

Figure 4-9. PRCM Reset Signals



4.4.1.2 Interrupt Requests

The PRCM module can generate an interrupt to the MPU interrupt controller module:

- PRCM_MPU_IRQ: Mapped to the MPU INTC module (M_IRQ_11 interrupt line)

4.5 PRCM Reset Manager Functional Description

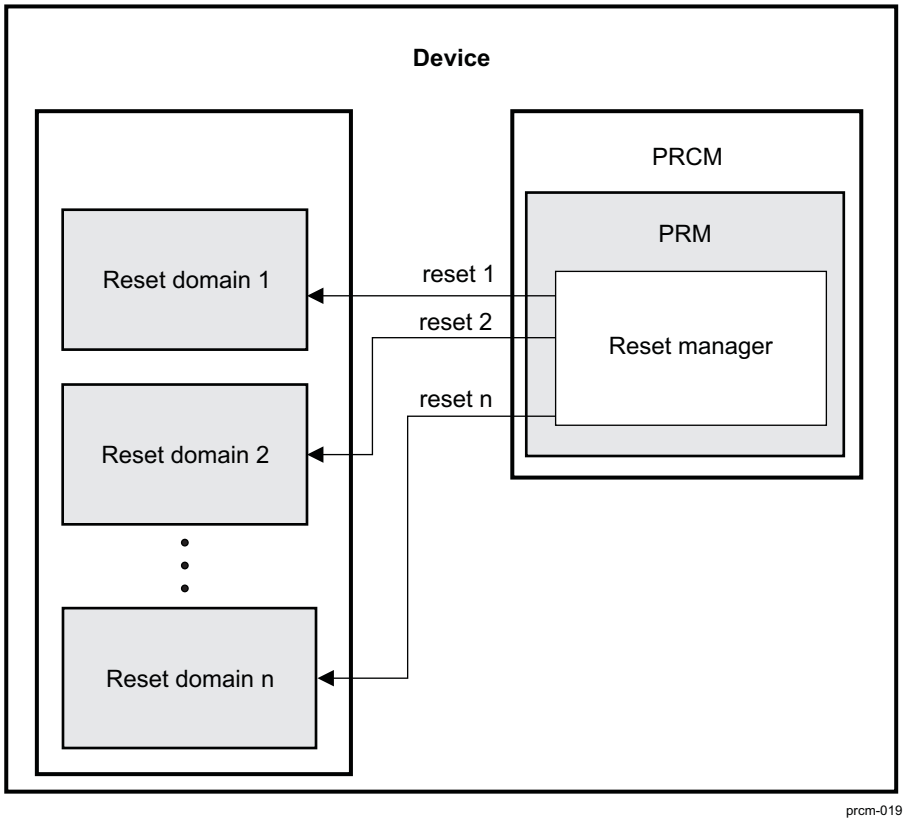
4.5.1 Overview

The PRCM manages the reset of all the modules in the device, including DPLLs.

Several reset domains are defined. A reset domain is defined by a unique reset signal that originates from the reset manager and is connected to one or multiple modules of the device. All the connected modules of the reset domain are reset simultaneously when the reset signal is asserted.

Figure 4-10 is an overview of the reset manager interface with a generic domain in the device.

Figure 4-10. Reset Manager Interface



Resets can be generated either by hardware sources or software control. For modules that can be reset by software control, a software-reset bit is implemented in their <module name>_SYSCONFIG configuration register. Software reset has the same effect on the module logic as a hardware reset.

Special reset control is available when the device operates under emulation control to reset specific reset domains.

NOTE: All reset assertion is asynchronous, and all reset signals are active low, except for the DPLL reset signals.

4.5.2 General Characteristics of Reset Signals

Reset signals can be categorized based on three criteria:

- Scope
- Occurrence
- Source type

4.5.2.1 Scope

A reset signal can be categorized according to its scope (the area of the device affected by that reset):

- Global reset: Affects the entire device; all modules are reset. Generally, when the device powers up or an abnormal operation is detected (secure watchdog timer overflows, the eFuse bad device is detected, etc.).
- Local reset: Affects one reset domain. When a software-reset control bit for a domain is set, only the group of modules within that domain is affected.

4.5.2.2 Occurrence

A reset signal can also be categorized depending on when the reset occurs:

- Cold reset: Occurs only on device power up or in certain emulation modes. The cold reset is a global reset that affects every module on the device. It usually corresponds to the initial power-on reset.
- Warm reset: Occurs when the device is in normal operating state. The warm reset is also a global reset, but it does not affect all the modules on the device. Usually, the device does not require a complete reboot on a warm reset. Several reset sources are types of warm resets, such as the global software reset and the watchdog reset.

Modules that behave differently in cold reset and warm reset have two reset signals: RST and PWRON_RST. These reset signals reconstruct warm reset and cold reset in modules that require them.

For a global warm reset, the PRCM performs the following sequence:

1. It applies a warm reset on all the modules.
2. It drives the sys_nreswarm reset output low and holds it for a specified length of time (programmed in the PRCM.PRM_RSTTIME[7:0] RSTTIME1 bit field).

A global warm reset does not apply to the following modules of the device:

- SDRC
- System control module (SCM) (I/O control)
- Part of PRM and CM registers (see note below)
- 32-kHz synchronization timer
- DPLL3 (refer to)
- Emulation modules
- eFuse farm

NOTE: For information on the PRCM registers affected by the global warm reset see the Registers Mapping Summary tables in Section 4.12, *PRCM Register Manual*.

4.5.2.3 Source Type

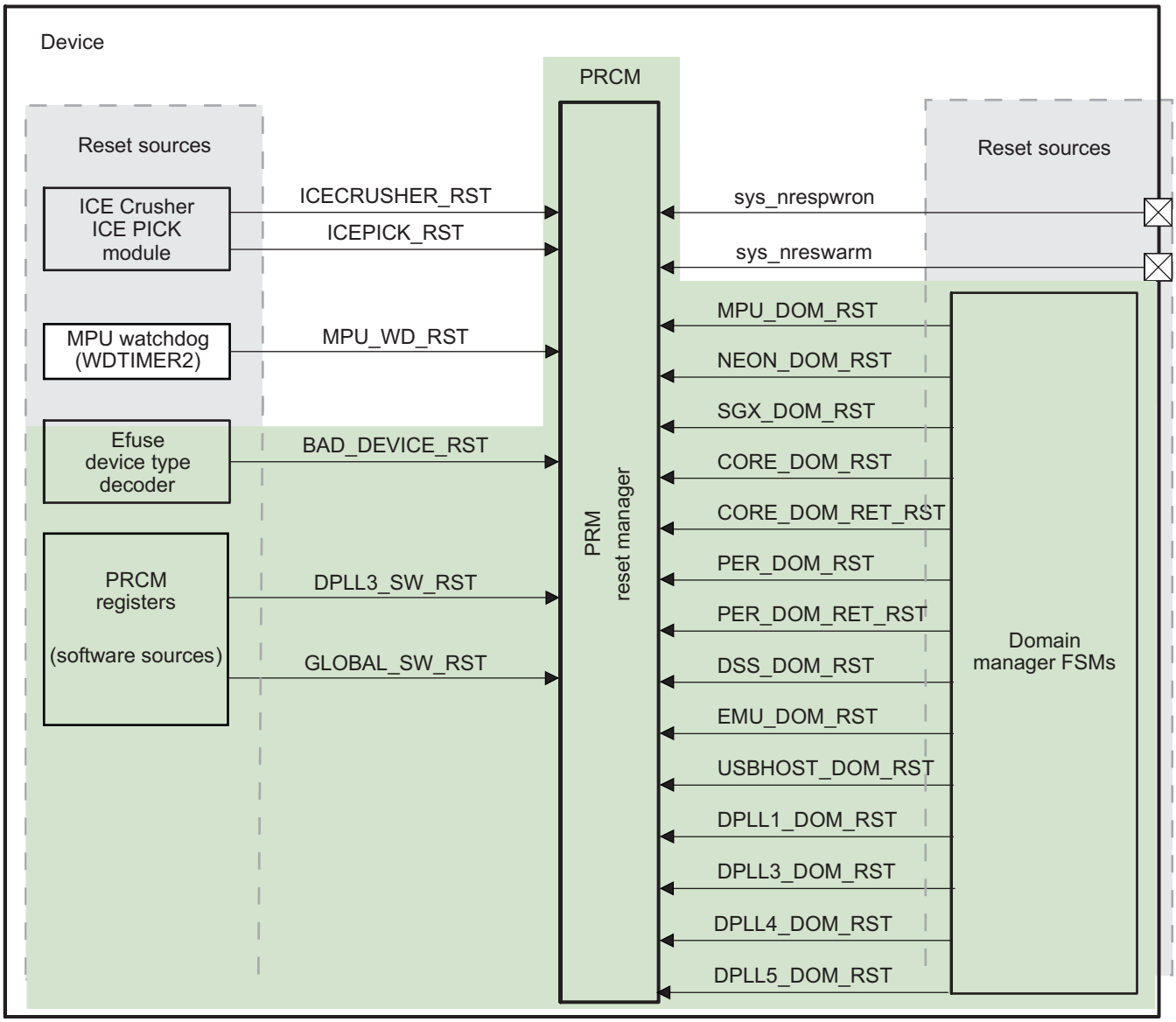
A reset can also be categorized depending on whether it is software-controlled or hardware-triggered:

- Software reset: Triggered by setting a bit in a configuration register of the PRCM module
- Hardware reset: Triggered by a signal from a hardware module inside or outside the PRCM module

4.5.3 Reset Sources

Figure 4-11 is an overview of the reset sources.

Figure 4-11. Reset Sources Overview



*The green region in the figure represents the boundary of the PRCM.

prcm-020

4.5.3.1 Global Reset Sources

Table 4-5 lists the global reset sources of the device. The global reset source signals received by the reset manager trigger the reset of all the device modules. For all hardware reset signals, the source of the reset is identified; for the software reset signals, the reset triggering bit is identified.

Table 4-5. Global Reset Sources

Type[1]	Name	Source/Control	Description
H/C	sys_nrespwron	Input pin	The entire device is reset on power up.
H/C	BAD_DEVICE_RST	PRCM	Asserted when during the power-up sequence the device is identified as bad, after reading eFuses.
H/W	sys_nreswarm	Bidirectional pin	External hardware warm reset

(1) H = Hardware reset, S = Software reset, C = Cold reset, W = Warm reset

Table 4-5. Global Reset Sources (continued)

Type[1]	Name	Source/Control	Description
H/W	SECURE_WD_RST	WDTIMER1	Security watchdog timer overflow reset
H/W	MPU_WD_RST	WDTIMER2	MPU watchdog timer overflow reset
H/W	MPU_SEC_VIOL_RST	Processor security FSM	Security violation reset request by the processor security FSM
S/W	GLOBAL_SW_RST	PRCM.PRM_RSTCTRL[1] RST_GS	Global software reset
S/W	DPLL3_SW_RST	PRCM.PRM_RSTCTRL[2] RST_DPLL3	Local cold reset for DPLL3 and a global cold reset to the device.
H/W	ICEPICK_RST	ICEPick module	Global warm reset from ICEPick emulation module.

4.5.3.2 Local Reset Sources

Table 4-6 lists the local reset sources of the device. A local reset source signal received by the reset manager resets only some of the device modules.

Table 4-6. Local Reset Sources

Type[1]	Name	Source/Control	Description
H/C	CORE_DOM_RET_RST	PRCM	Asserted only for a domain transition from OFF to ACTIVE state as in the case of the device power-up. See note below for further clarification.
H/C	USB_DOM_RET_RST	PRCM	
H/C	PER_DOM_RET_RST	PRCM	
H/C	MPU_DOM_RST	PRCM	Asserted only for a domain transition from OFF to ACTIVE state as in the case of the device power-up. See note below for further clarification.
H/C	NEON_DOM_RST	PRCM	
H/C	SGX_DOM_RST	PRCM	
H/C	CORE_DOM_RST	PRCM	
H/C	PER_DOM_RST	PRCM	
H/C	DSS_DOM_RST	PRCM	
H/C	DPLL1_DOM_RST	PRCM	
H/C	DPLL3_DOM_RST	PRCM	
H/C	DPLL4_DOM_RST	PRCM	
H/C	DPLL5_DOM_RST	PRCM	

(1) H = Hardware reset, S = Software reset, C = Cold reset, W = Warm reset

NOTE: For domains with <domain name>_DOM_RST and <domain name>_DOM_RET_RST, the reset sources are asserted together when the device transitions from OFF to ON power state, whereas only <domain name>_DOM_RET_RST is asserted on a global or local warm reset.

4.5.4 Reset Distribution

Each domain can contain one power-on reset (RSTPWRON) and one or more reset (RST) signals. These signals behave as follows:

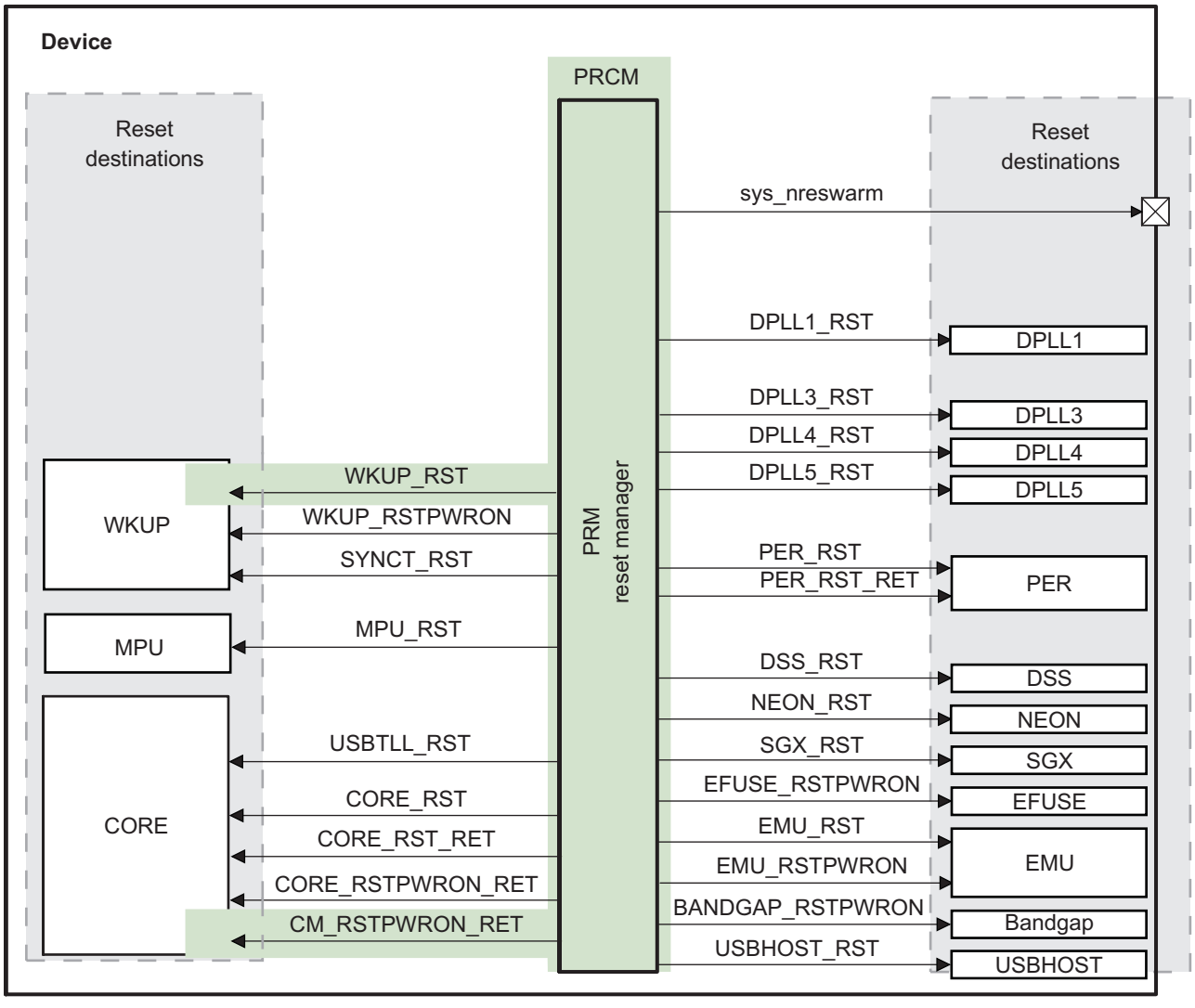
- On any global or local cold reset, both RST and RSTPWRON are asserted.
- On any global or local warm reset, only RST is asserted.

The CORE domain receives two additional legacy reset signals: retention reset (RST_RET) and power-on retention reset (RSTPWRON_RET). These signals behave as follows:

- On any global cold reset, both RST_RET and RSTPWRON_RET are asserted.
- On any global warm reset, only RST_RET is asserted.

Figure 4-12 shows the reset distribution among the domains.

Figure 4-12. Reset Destinations Overview



*The green region in the figure represents the boundary of the PRCM

prcm-021

4.5.5 Domain Reset Descriptions

4.5.5.1 MPU Domain

The MPU domain has one input and one output reset signal (see Table 4-7).

Table 4-7. MPU Domain Reset Signals

Name	I/O[1]	Source/Destination[2]	Reset Domain
MPU_RST	I	PRM	Resets the MPU processor core and the asynchronous bridge in the MPU domain.
MPU_SEC_VIOL_RST	O	PRM	Global cold reset source to reset manager. Generated by the security FSM in the MPU subsystem.

(1) I = Input, O = Output
(2) Source for an input signal and destination for an output signal.

NOTE: The MPU_CLK is divided by 2 inside the MPU subsystem to generate the ARM_FCLK This divider is only active when the DPLL is locked. (Refer to MPU Subsystem for information on ARM_FCLK).

4.5.5.2 NEON Domain

The NEON domain has one reset input signal (see Table 4-8).

Table 4-8. NEON Domain Reset Signal

Name	I/O[1]	Source	Reset Domain
NEON_RST	I	PRM	Resets the NEON coprocessor

(1) I = Input, O = Output

4.5.5.3 CORE Domain

The CORE domain has eight input reset signals. (See Table 4-9).

Table 4-9. CORE Domain Reset Signals

Name	I/O[1]	Source/Destination[2]	Reset Domain
CORE_RST	I	PRM	Resets parts of the three asynchronous bridges, MPU INTC, interconnects, GPMC, OCM, UART[1,2], HDQ and HS USB, I2C[1..3], McBSP 1 and 5, McSPI [1..3], MMC[1..3], GPTIMER[10,11], D3D[1,2], SHAM1, RNG, AES[1..2], PKA, and SHAM2
CORE_RST_RET	I	PRM	Resets part of the SDRC, SDMA, SMS, and MPU INTC
CORE_RSTPWRON_RET	I	PRM	Resets part of the SDRC and SCM
CM_RSTPWRON_RET	I	PRM	Resets the clock manager
CPEFUSE_RST	I	PRM	Reset the Customer Programmable EFUSE controller. Asserted under the same condition as that of CORE_RST. The CPEFUSE functional clock must be active to release the reset. This clock is enabled by default following a power-on reset. (For the reset sequence, see Section 4.5.9.2.)
USBTLL_RST	I	PRM	Resets the USB TLL asynchronously

(1) I = Input, O = Output
(2) Source for an input signal and destination for an output signal

The CM logic is reset on:

- Any global cold reset

Because the CM logic is not reset in this case, the CM registers that are sensitive to a warm reset must also be reset synchronously with the L4 clock when a global warm reset occurs.

4.5.5.4 DSS Domain

The DSS domain has one reset input signal (see Table 4-10).

Table 4-10. DSS Domain Reset Signal

Name	I/O[1]	Source	Reset Domain
DSS_RST	I	PRM	Resets the entire display subsystem

(1) I = Input, O = Output

4.5.5.5 USBHOST Domain

The USBHOST domain has one reset input signal (see Table 4-11).

Table 4-11. USBHOST Domain Reset Signal

Name	I/O[1]	Source	Reset Domain
USBHOST_RST	I	PRM	Resets the entire HS USB Host subsystem

(1) I = Input, O = Output

4.5.5.6 SGX Domain

The SGX domain has one reset input signal (see Table 4-12).

Table 4-12. SGX Domain Reset Signal

Name	I/O[1]	Source	Reset Domain
SGX_RST	I	PRM	Resets the entire SGX subsystem

(1) I = Input, O = Output

4.5.5.7 WKUP Domain

The WKUP domain has three reset input signals and two reset output signals (see Table 4-13).

Table 4-13. WKUP Domain Reset Signals

Name	I/O[1]	Source/Destination[2]	Reset Domain
WKUP_RST	I	PRM	Resets the GPTIMER[1,12], WDTIMER2, GPIO 1 and USIMOCP
WKUP_RSTPWRON	I	PRM	Reset the wake-up control module and WDTIMER1
SYNCT_RST	I	PRM	Resets the 32-kHz sync timer. This reset is directly connected to the sys_nrespwron global reset source.
SECURE_WD_RST	O	PRM	Global warm reset for PRM. Generated by WDTIMER1.
MPU_WD_RST	O	PRM	Global warm reset for PRM. Generated by WDTIMER2.

(1) I = Input, O = Output
(2) Source for an input signal and destination for an output signal

The PRM logic is reset on any global cold reset. Because the PRM logic is not reset in this case, the PRM registers that are sensitive to a warm reset must also be reset synchronously with the system clock when a global warm reset occurs.

4.5.5.8 PER Domain

The PER domain has two reset input signals (see Table 4-14).

Table 4-14. PER Domain Reset Signal

Name	I/O[1]	Source	Reset Domain
PER_RST	I	PRCM	Resets the UART3, McBSP[2,3,4], GPTIMER[2,..,9], WDTIMER3 modules
PER_RST_RET	I	PRCM	Resets the GPIO [2,..,6] modules

(1) I = Input, O = Output

4.5.5.9 DPLL Domains

The DPLL domains for DPLL1, DPLL3, DPLL4 and DPLL5 each have one reset input signal.

Table 4-15. DPLL Domain Reset Signals

Name	I/O[1]	Source	Reset Domain
DPLL1_RSTPWRON	I	PRCM	Resets the DPLL1 module
DPLL3_RSTPWRON	I	PRCM	Resets the DPLL3 module
DPLL4_RSTPWRON	I	PRCM	Resets the DPLL4 module
DPLL5_RSTPWRON	I	PRCM	Resets the DPLL5 module

(1) I = Input, O = Output

They are asserted for any type of global cold reset.

4.5.5.10 EFUSE Domain

The EFUSE domain has one reset input signal (see Table 4-16).

Table 4-16. EFUSE Domain Reset Signal

Name	I/O[1]	Source	Reset Domain
EFUSE_RSTPWRON	I	PRCM	Resets the eFuse controller

(1) I = Input, O = Output

This signal is asserted for any type of global cold reset.

4.5.5.11 BANDGAP Logic

The BANDGAP logic has one reset input signal (see Table 4-17).

Table 4-17. BANDGAP Logic Reset Signal

Name	I/O[1]	Source	Reset Domain
BANDGAP_RSTPWRON	I	PRCM	Resets the BANDGAP logic

(1) I = Input, O = Output

This signal is asserted for any type of global cold reset.

4.5.5.12 Other Module Resets

Both Figure 4-13 and Figure 4-14 show the reset connectivity of the remaining modules. The CBASS, USBOTG subsystem, CPGMAC subsystem, VPFE, HECC, EMIF subsystem, and VBUS-to-OCP bridges are connected to the Warm Reset output of the PRM as shown in the Figure 4-13.

Additionally, a software reset register is present the system control module to allow software to reset these modules. Note that SW Reset is not self clearing. Software needs to write '1' to put these modules in reset and write '0' to bring it out of reset. Please refer to the *System Control Module* chapter for more details on the software reset register.

Figure 4-13. Other Module Reset Distributions Overview

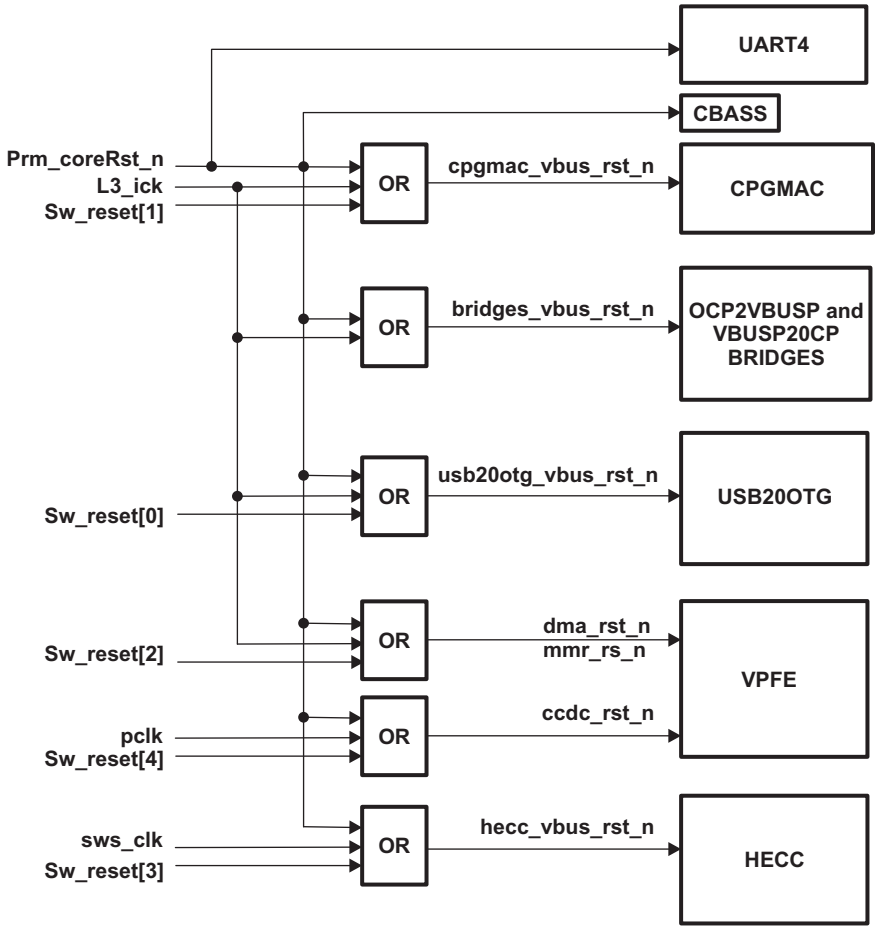
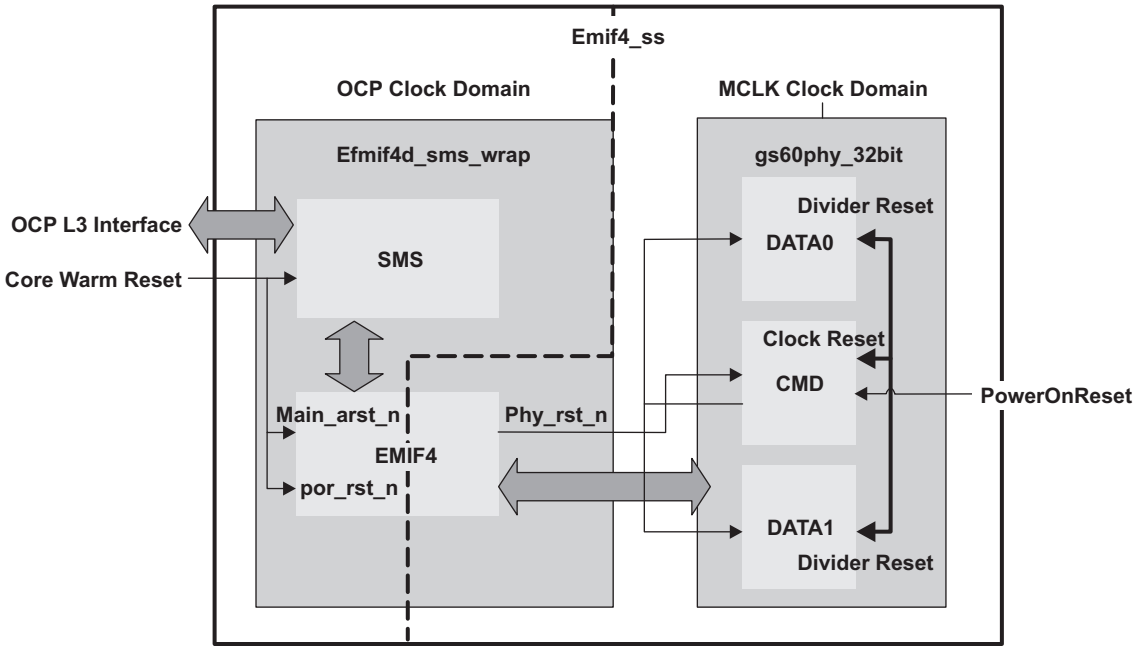


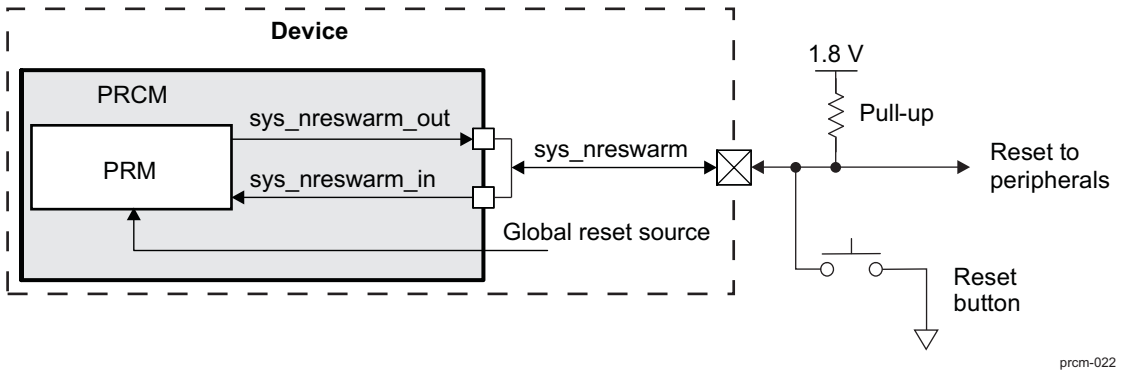
Figure 4-14. EMIF4 Reset Distributions Overview



4.5.5.13 External Warm Reset Assertion

Figure 4-15 shows the external warm reset interface.

Figure 4-15. External Warm Reset Interface



Any global reset source (internal or external) causes sys_nreswarm_out to be driven and maintained at the boundary of the device for at least the amount of time configured in the PRCM.PRM_RSTTIME[7:0] RSTTIME1 bit field. This ensures that the device and its related peripherals are properly reset together.

NOTE: Because the system warm-reset output is implemented on a bidirectional pad, any input pulse on sys_nreswarm causes a global warm reset.

4.5.6 Reset Logging

The reset in the device is logged in two ways. First, dedicated registers in the PRCM (that is, the RM_RSTST_power domain> and PRM_RSTST registers) log the reset source. Second, the SCM also logs the device reset activity in dedicated registers, for security purposes.

4.5.6.1 PRCM Reset Logging Mechanism

The reset status registers (RM_RSTST_power domain> and PRM_RSTST) are reset asynchronously on assertion of a global cold reset. However, a reset status bit is always logged when the reset is released to the domain.

For this reason, after the assertion of a global cold reset, the reset status register is cleared to 0. When the domain reset is released, the register bit to log the global cold reset (that is, the RM_RSTST_<power domain> [0] GLOBALCOLD_RST bit) is updated to 1. For the same reason, the reset status register of domains released from reset by software is updated only when software releases the domain reset.

The assertion of a global cold reset prevents logging any other source of reset until after the release of the domain reset. This is valid in the following situations:

- A source of reset other than global cold reset is asserted before, during, or after the active period of a global cold source of reset and before the release of the domain reset signal.
- A source of reset other than global cold reset was asserted and then released, but a global cold reset source was asserted before the release of the domain reset signal.

4.5.6.2 SCM Reset Logging

The PRM exports reset the activity status signals to the SCM. For security purposes, the SCM uses these signals to log a reset status in the SCM.CONTROL_SEC_STATUS register. The reset activity status signal is asserted high when any source of reset to the domain is active.

It also provides reset status for the following global reset signals:

- Global cold reset
- Global warm reset
- Global warm secure watchdog reset (SECURE_WD_RST)
- Global warm security violation reset (MPU_SEC_VIOL_RST)

There is one reset activity status signal for each of the following domains:

- CORE
- DSS
- EMU
- SGX
- MPU
- NEON
- PER
- USBHOST

These signals are asserted high on assertion of any source of reset on the domain and logged. For information about the SCM, see the *System Control Module* chapter.

4.5.7 Reset Management Overview

The reset management structure in the device can be considered as a 2-layered structure composed of multiple instances of the reset manager. In the first layer, a top-level reset manager, called the device reset manager, handles all the sources of global reset (cold and warm). It provides reset managers for the second layer, called local reset managers, and the global reset and global power-on reset signals.

Each domain has a minimum of one local reset manager. The local reset manager also receives resets, such as the software reset and domain power transition reset, from the local reset source for the domain.

Figure 4-16 through Figure 4-19 provide an overview of reset management in the device. They do not provide reset sequencing between the reset managers.

NOTE: The domain must be ready (that is, the domain clocks must be active) before its reset is released.

Figure 4-16. Device Reset Manager Overview

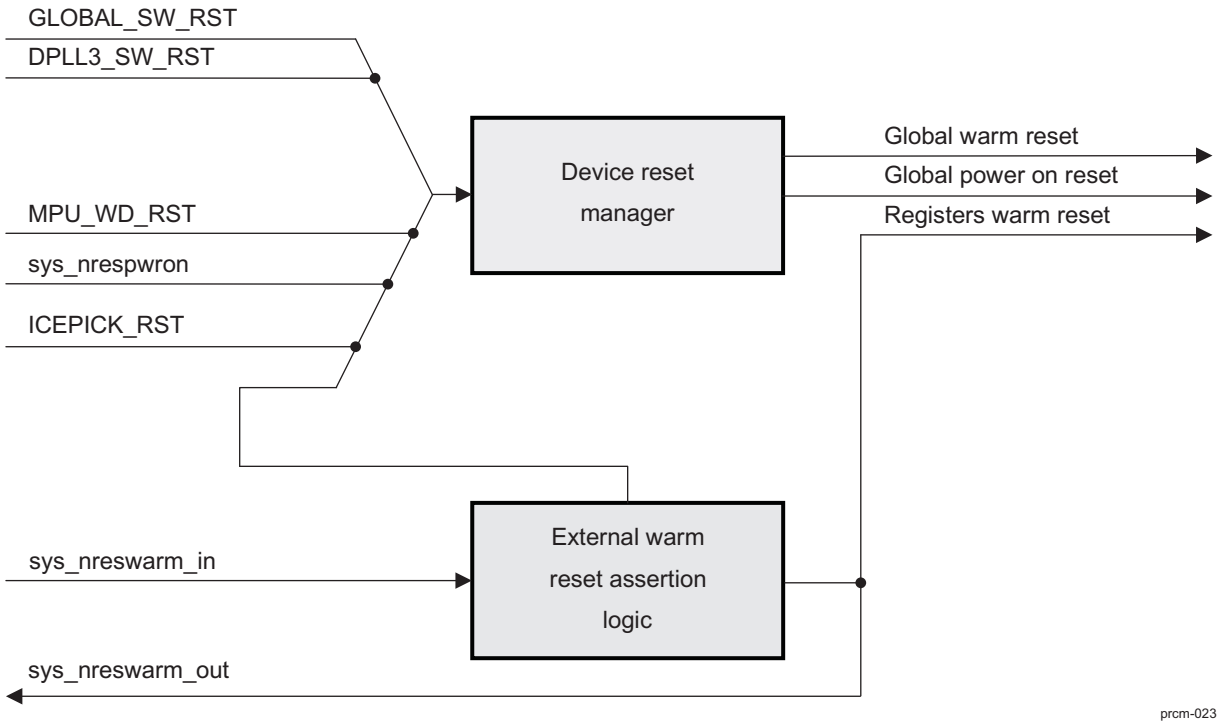


Figure 4-17. Domain Reset Management: Part 1

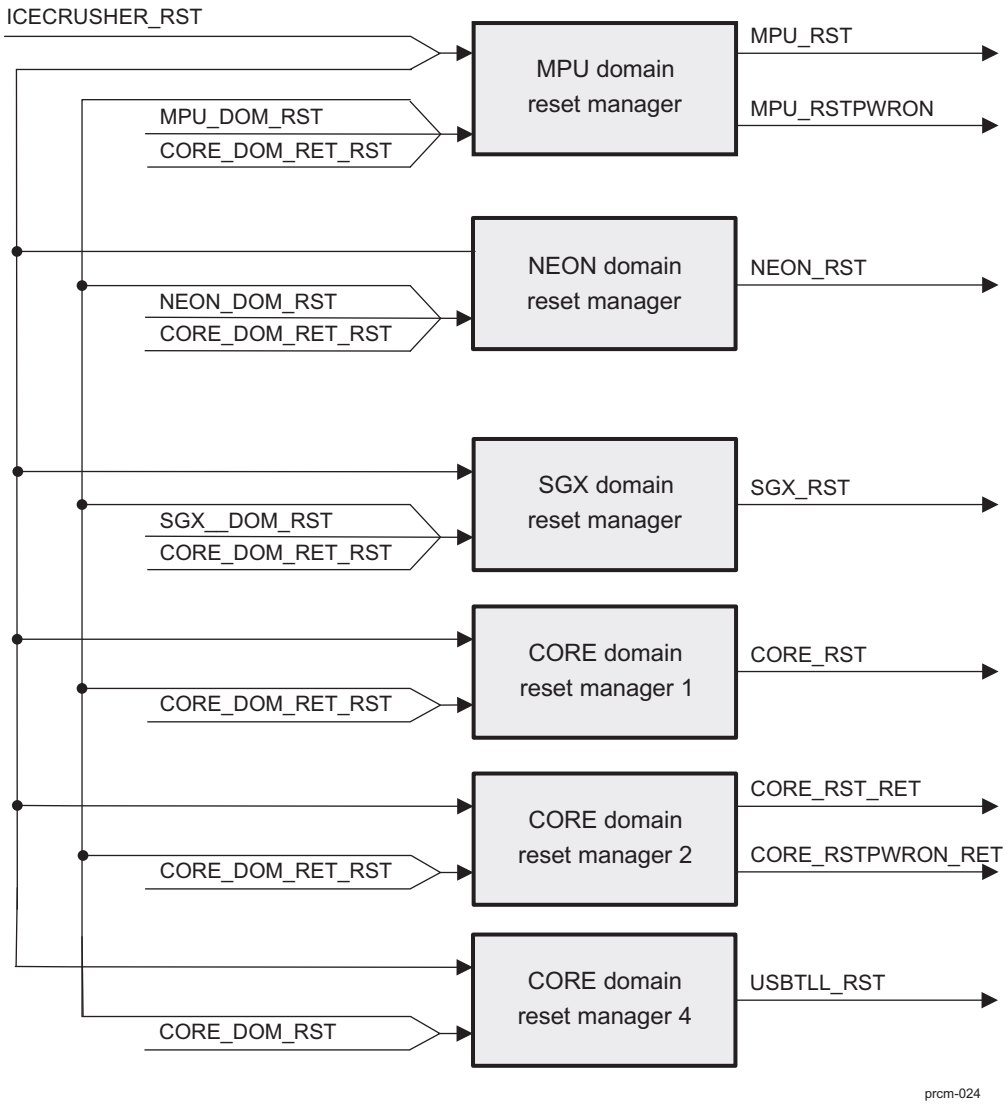


Figure 4-18. Domain Reset Management: Part 2

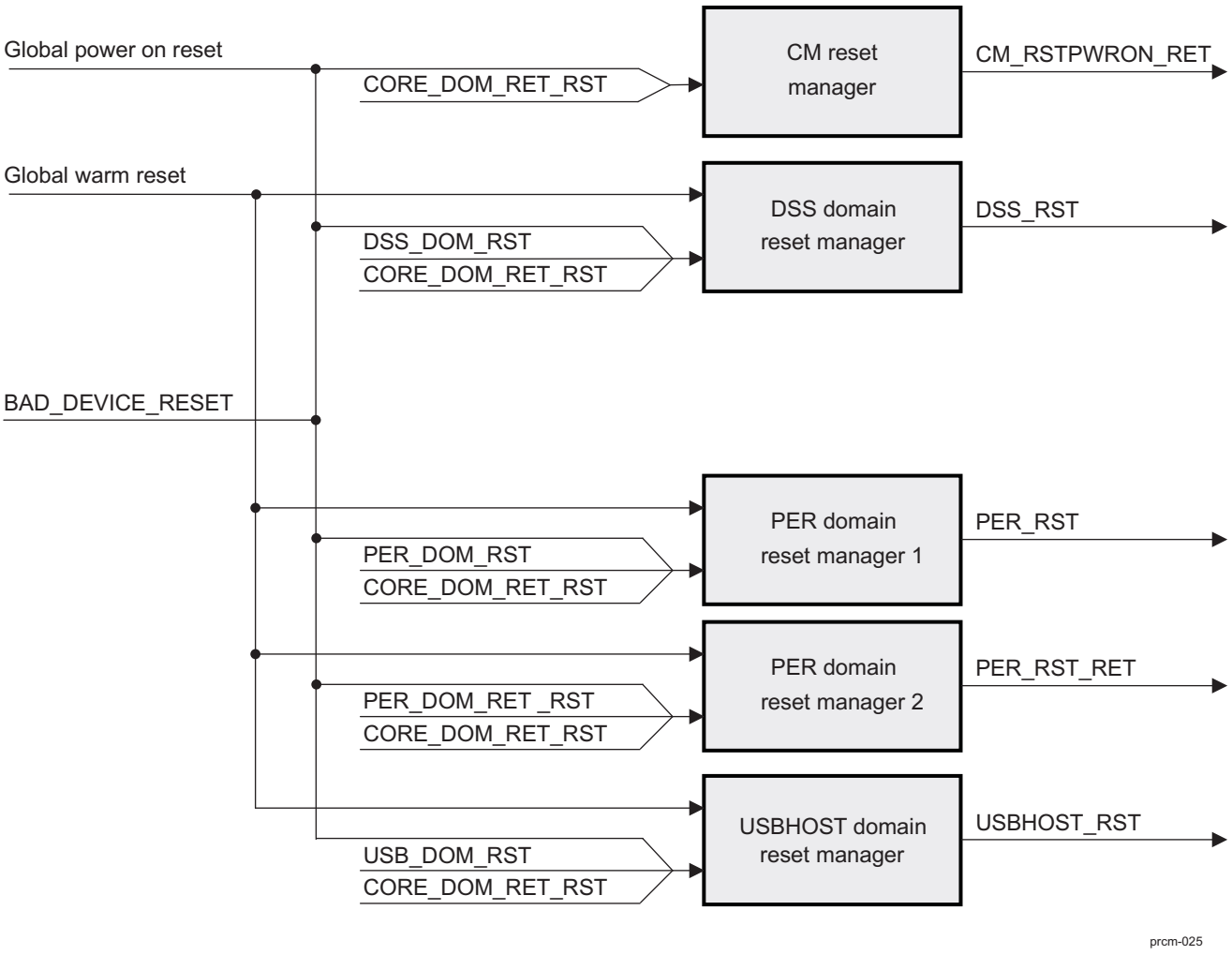
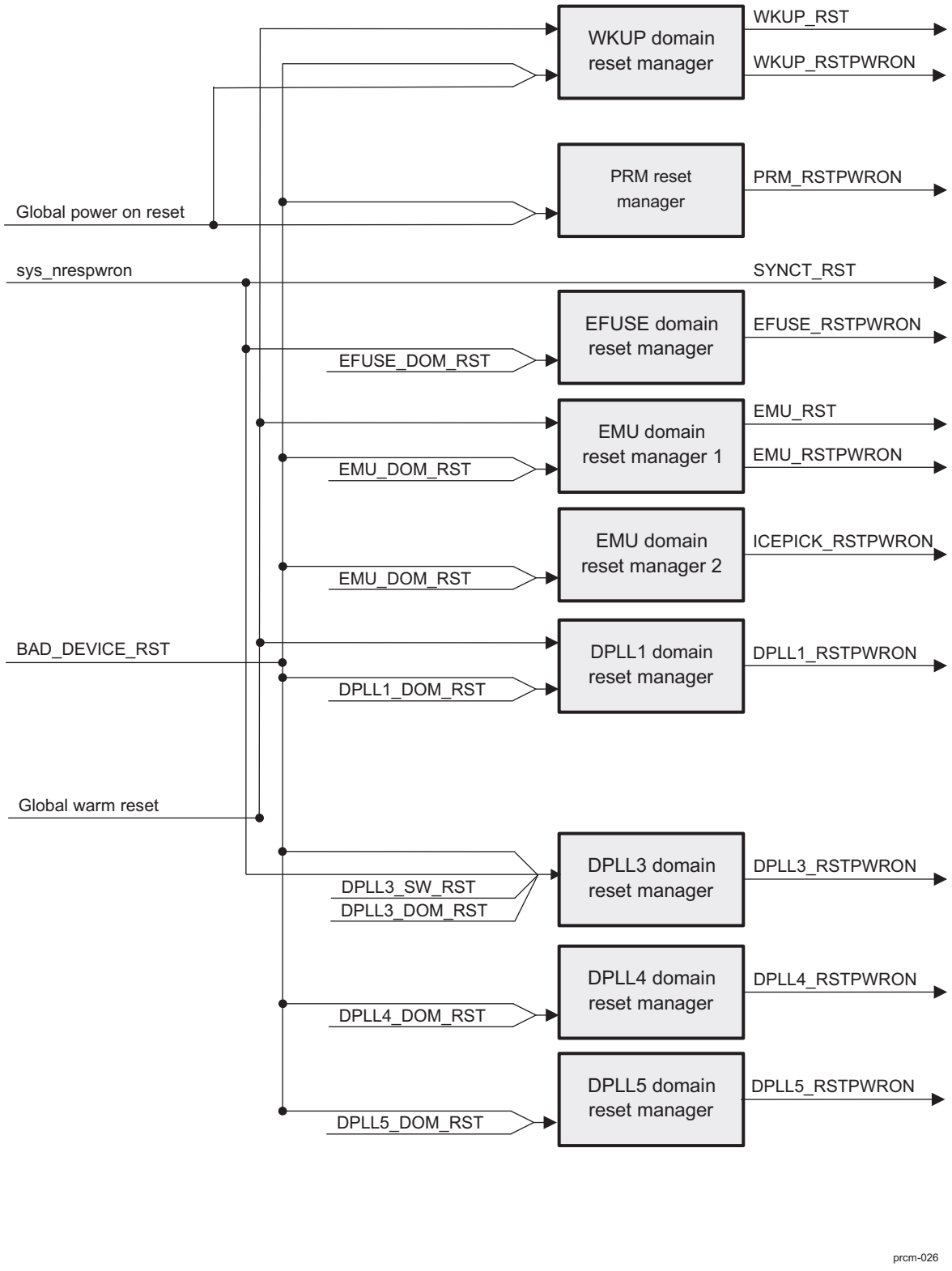


Figure 4-19. Domain Reset Management: Part 3



4.5.8 Reset Summary

Table 4-18 and Table 4-19 summarize the different sources of global and local resets and their actions on the reset signals.

Table 4-18. Global Reset Summary[1]

Domain Resets		Reset Sources				
Power Domain	Signal	Cold Reset		Warm Reset		
		sys_nres pweron	DPLL3_ SW_RST	sys_nres warm_in	MPU_WD_ RST	GLOBAL_ SW_RST
MPU	MPU_RST	■	■	■	■	■
NEON	NEON_RST	■	■	■	■	■
SGX	SGX_RST	■	■	■	■	■
CORE	CORE_RST	■	■	■	■	■
	CORE_RSTPWRON	■	■	■	■	■
	CORE_RST_RET	■				
	CORE_RSTPWRON_RET	■				
	CM_RSTPWRON_RET	■				
	USBTLL_RST	■	■	■	■	■
WKUP	WKUP_RST	■	■	■	■	■
	SYNCT_RST	■				
PER	PER_RST	■	■	■	■	■
	PER_RST_RET	■	■	■	■	■
DSS	DSS_RST	■	■	■	■	■
USBHOST	USBHOST_RST	■	■	■	■	■
EMU	EMU_RST	■	■	■	■	■
	EMU_RSTPWRON	■	■			
	ICEPICK_RSTPWRON	■	■			
DPLL1	DPLL1_RSTPWRON	■				
DPLL3	DPLL3_RSTPWRON	■	■			
DPLL4	DPLL4_RSTPWRON	■				
DPLL5	DPLL5_RSTPWRON	■				
SR	SR_RST	■	■	■	■	■
EFUSE	EFUSE_RSTPWRON	■				
BANDGAP	BANDGAP_RSTPWRON	■				
Device pad (output)	SYS_NRESWARM_OUT	■	■	■	■	■

[1] The shaded blocks identify the domain reset signals triggered as a result of the reset source signal (at the head of the column).

Table 4-19. Local Reset Summary[1]

Domain Resets		Reset Sources				
Power Domain	Signal	Cold Reset				Warm Reset
		CORE_DOM_RET_RST	PER_DOM_RET_RST	DPLL3_SW_RST	BAD_DEVICE_RESET	ICECRUSHER_RST
MPU	MPU_RST	■			■	■
NEON	NEON_RST	■			■	■
SGX	SGX_RST	■			■	■
CORE	CORE_RST	■			■	■
	CORE_RST_RET	■			■	■
	CORE_RSTPWRON_RET	■			■	■
	CM_RSTPWRON_RET				■	■
	USBTLL_RST	■			■	■
WKUP	WKUP_RST				■	■
	SYNCT_RST					
PER	PER_RST	■	■		■	■
	PER_RST_RET	■	■		■	■
DSS	DSS_RST	■			■	■
USBHOST	USBHOST_RST	■			■	■
EMU	EMU_RST				■	
	EMU_RSTPWRON				■	
	ICEPICK_RSTPWRON					
DPLL1	DPLL1_RSTPWRON	■			■	■
DPLL3	DPLL3_RSTPWRON	■		■	■	■
DPLL4	DPLL4_RSTPWRON	■			■	■
DPLL5	DPLL5_RSTPWRON	■			■	■
SR	SR_RST				■	■
EFUSE	EFUSE_RSTPWRON					
BANDGAP	BANDGAP_RSTPWRON					
Device pad (output)	sys_nreswarm_out			■	■	■

[1] The shaded blocks identify the domain reset signals triggered as a result of the reset source signal (at the head of the column).

4.5.9 Reset Sequences

4.5.9.1 Power-Up Sequence

NOTE: Please reference the device-specific data manual for power-up sequencing.

4.5.9.2 CPEFUSE Reset Sequence

In the CPEFUSE reset sequence, Customer Programmable EFUSE is coupled with the SCM. Whenever the SCM is reset, the CPEFUSE cells are sensed. This sequence is initiated by the PRCM module and the SCM (hardware-controlled) after a device cold reset. Under these conditions, the CM part of the PRCM is released from reset, and, according to the configuration of the PRCM.CM_FCLKEN3_CORE[0] EN_CPEFUSE bit, the CPEFUSE functional clock is automatically restarted and the CPEFUSE_RST reset is released (de-asserted). The SCM completes the autoload sequence of the CPEFUSE. Software must poll the autoload completion status bit in the SCM before switching off the CPEFUSE functional clock.

Whenever the software must blow a CPEFUSE, it must ensure that the functional clock is enabled; it starts the autoload sequence by programming the SCM.

For information about the SCM, see the *System Control Module* chapter.

4.6 PRCM Power Manager Functional Description

4.6.1 Overview

4.6.1.1 Device Partitioning

Table 4-20 lists the device modules split over the domains.

Table 4-20. Domain Modules

Power Domain	Modules
MPU	MPU core
	ICE-crusher CS
	MPU async bridge (master)
	SSM
NEON	NEON coprocessor
SGX	SGX subsystem
CORE	GPMC
	GPTIMER[10, 11]
	HDQ/1-Wire
	HS USB
	I2C[1, 2, 3]
	McBSP[1, 5]
	McSPI[1, 2, 3, 4]
	MMC/SD/SDIO[1, 2, 3]
	MPU async bridge (slave)
	MPU INTC
	OCM_RAM
	OCM_ROM
	SCM
	SDRC
	SHAM[1, 2]
	SMS

Table 4-20. Domain Modules (continued)

Power Domain	Modules
	L3 Interconnect
	UART[1, 2]
	SDMA
	Temperature sensor (x2)]
	CPEfuse farm
	L4_Core interconnect
DSS	Display subsystem
	Video DAC
PER	UART3
	WDTIMER3
	McBSP[2..4]
	GPIO[2..6]
	GPTIMER[2..9]
	L4_Per interconnect
WKUP	GPIO1
	GPTIMER[1, 2]
	WDTIMER[1, 2]
	32-kHz sync timer
	L4_Wakeup interconnect
EMU	CWT
	DAP-APB
	ETB
	ICEPICK
	SDTI
	TRACEPORT
	L4_EMU interconnect
EFUSE	eFuse farm
DPLL1	MPU DPLL
DPLL3	CORE DPLL
DPLL4	Peripherals DPLL
DPLL5	Peripherals DPLL2

4.6.1.2 Domain State Transitions

For each domain, the PRM manages state transitions, controlling domain clocks and resets.

Two types of power state transitions are possible:

- Sleep: Moving from a higher consumption power state (ACTIVE) to a lower consumption power state (INACTIVE).
- Wake-up: Moving directly from a lower consumption power state (INACTIVE) to the ACTIVE power state.

4.6.1.3 Device Power Modes

A device power mode is a specific functional combination of the states of all the domains of the device.

Unlike domain states, device power modes are not hardware-defined; they are defined by software as relevant combinations of the domain states.

There are two types of device power modes:

- Active: Any valid combination of domain states in which one or several domains are still active,

regardless of whether any software is running.

- Standby: Any valid combination of domain states in which all the domains are in INACTIVE state.

4.6.2 Domain Implementation

4.6.2.1 Domain Dependencies

Besides the inner conditions to operate a state transition on a single domain, the device offers hardwired and software-programmable dependencies between the domains.

Two kinds of dependencies exist:

- Sleep dependencies: Used to start a sleep transition on a domain only if the related domains are in mute mode (not requesting any service from the linked domain)
- Wake-up dependencies: Used to initiate a wake-up transition on a domain as a consequence of a linked domain wake-up transition

Two dedicated sets of registers (PRCM.CM_SLEEPDEP_<domain> and PRCM.PM_WKDEP_<domain>) allow the setting of programmable dependencies.

Section 4.8.5, *Sleep and Wake-Up Dependencies*, summarizes the possible dependency combinations between domains.

4.6.2.2 Domain Software Controls

If all conditions are met to initiate a domain state transition (that is, all the modules are idle/standby and the related clocks are shut down), the PRCM automatically manages the transition according to the following settings:

- Dependencies setting: The PRCM.CM_SLEEPDEP_<domain> registers and PRCM.PM_WKDEP_<domain> registers are used to set/clear programmable dependencies between domains.

  Not all dependencies are programmable by software; some are hardwired and thus do not allow any control. For more information about the dependencies between domains, see Section 4.8, *Idle and Wake-Up Management*, and Section 4.11, *Basic Programming Model*.
- Event generator: Three registers (PRCM.PM_EVGENCTRL_MPU, PRCM.PM_EVGENONTIM_MPU, and PRCM.PM_EVGENOFFTIM_MPU) allow the MPU domain to be switched between on inactive mode. The PRCM.PM_EVGENONTIM_MPU and PRCM.PM_EVGENOFFTIM_MPU registers are used to set the durations of the on and inactive modes, respectively.

  For details, see Section 4.11, *Basic Programming Model*, and Section 4.12,*PRCM Registers Manual*.

4.7 PRCM Clock Manager Functional Description

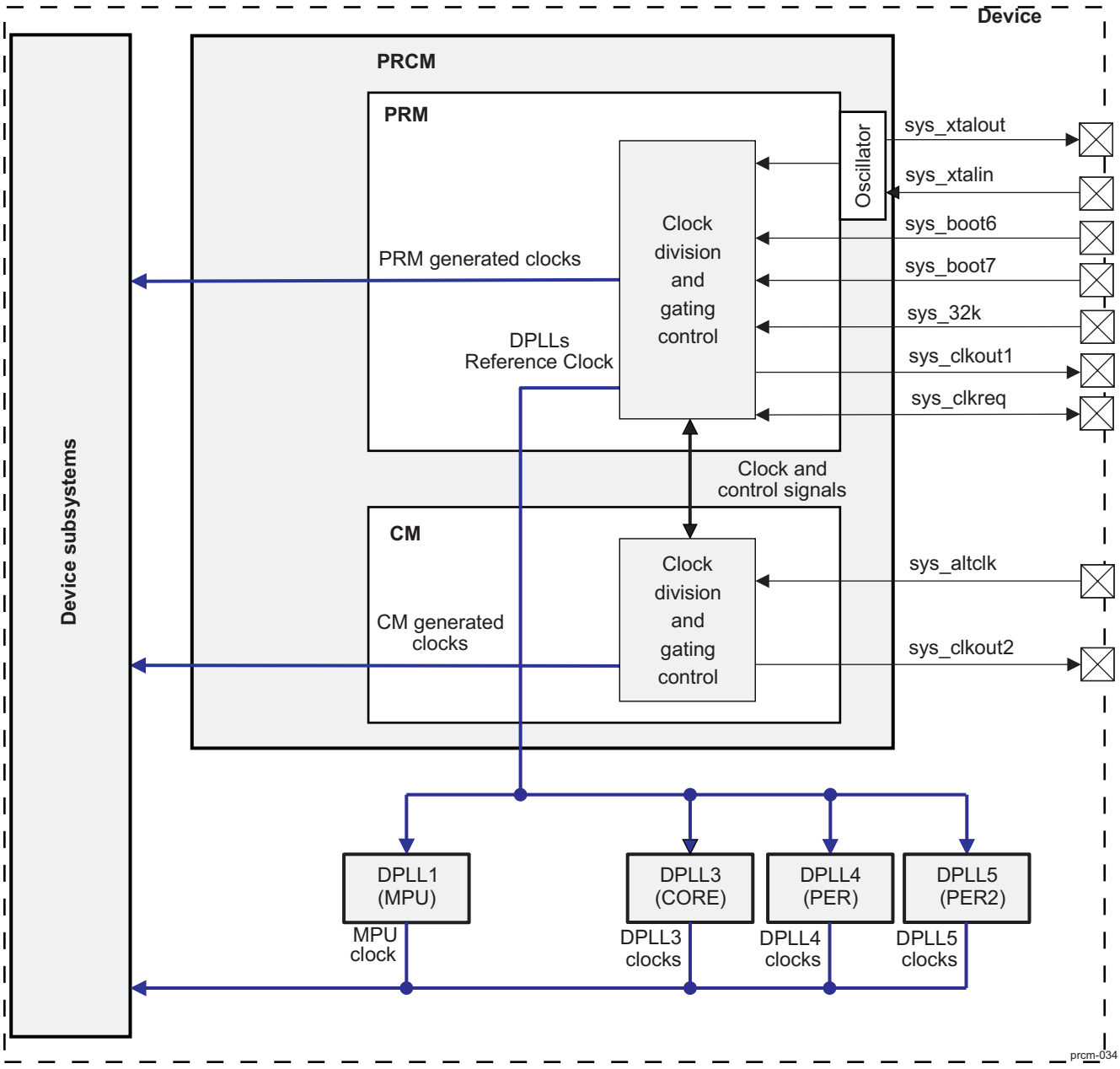
This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power saving considerations, ensure that clocks to unused domains and modules are properly gated.

4.7.1 Overview

The PRCM module provides control for clock generation, division, distribution, synchronization, and gating. It distributes the clock sources to all the modules in the device.

The device-level clock generation is handled by internal oscillators (the system clock oscillator and the 32-kHz oscillator for secure clock) and DPLLs; clock division and gating are handled by the PRM and the CM sections of the PRCM. Figure 4-20 shows the high-level clock-management scheme in the device.

Figure 4-20. PRCM Clock Manager Overview



4.7.1.1 Interface and Functional Clocks

The PRCM propagates two kinds of clocks:

- Interface clock: Ensures proper communication between any module and the system interconnects (L3 or L4). In most cases, the interface clock supplies the interface and registers of the module. For some modules, the interface clock is also used as the functional clock.
- Functional clock: Supplies the functional part of a module or subsystem. In some cases, a module or subsystem may require several functional clocks.

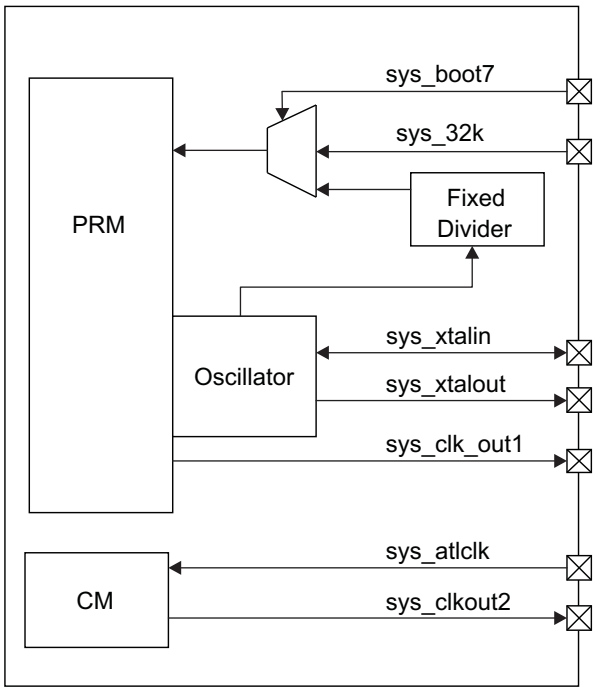
To be operational, a module requires functional clock(s); to communicate with other modules, it requires an interface clock. For example, the functional clock of a general-purpose timer (GPTIMER) must be active for it to run, but its interface clock can be turned off.

A module can use one or more optional functional clocks. Because an optional functional clock is used only for specific features of the module, it can be shut down without stopping the module activity (for example, 54 MHz for the DSS domain).

4.7.2 External Clock I/Os

Figure 4-21 shows the external clock I/Os of the device.

Figure 4-21. External Clock I/O



4.7.2.1 External Clock Inputs

4.7.2.1.1 32-kHz Always-On Clock

The 32-kHz clock is used for low-frequency operation (timers, denouncing, etc.). It also supplies the WKUP domain for operation in the lowest power standby mode. The device supports an external 32kHz clock signal or an internal 32kHz clock divided down from the external 26MHz high-frequency clock. The selection between external and internal 32kHz clock is done by sys_boot7 pin. The sys_32k input pin supplies the 32-kHz always-on clock (32K_FCLK clock) when the sys_boot7 pin is configured for external 32-kHz clock.

4.7.2.1.2 High-Frequency System Clock

The high-frequency system clock (SYS_CLK) is either supplied to the device from an external clock source through the sys_xtalin input pin or is generated internally by a local system clock crystal oscillator. In the latter case, a crystal is connected between the sys_xtalout and sys_xtalin device pins. The sys_boot[6] pin is used to set the oscillator operating mode (see Figure 4-6).

The source system clock can be 26 MHz and is internally divided by 2 to provide the standard frequencies (26 MHz becomes 13 MHz). It supplies the reference to the DPLLs and is also used by several modules. The system clock is activated in the device after the device power-on reset is released by the reset manager in the PRCM module.

The source-clock selection register (PRCM.PRM_CKSEL [2:0] SYS_CLKIN_SEL bit field) is set by the software to identify the input frequency of the system clock.

Table 4-21 provides the system clock input configurations.

Table 4-21. System Clock Input Configurations

Input Source	Device Pin Mapping	Description
Quartz	sys_xtalin and sys_xtalout	Internal oscillator is enabled.
Square clock (1.8-V CMOS signal)	sys_xtalin (sys_xtalout is not connected)	Internal oscillator is bypassed.

NOTE: An external pullup or pulldown tied on the sys_boot6 input pin of the device determines whether the internal oscillator is used (oscillator mode) or an external clock source is supplied to the sys_xtalin input pin (bypass mode).

4.7.2.1.3 Alternate Clock

The sys_altclk pin can be used to provide an alternate 54-MHz, 48-MHz, or any other frequency clock.

4.7.2.2 External Clock Outputs

The device can output two clocks externally:

- sys_clkout1 can output an oscillator clock (12, 13, 16.8, 19.2, 26, or 38.4 MHz) at any time. The output oscillator clock can be controlled either by software or externally using sys_clkreq control.
- sys_clkout2 can output sys_clk (12, 13, 16.8, 19.2, 26, or 38.4 MHz), core_clk (CORE DPLL output), or 96-MHz or 54-MHz clocks. It can be divided by 2, 4, 8, or 16. This output is active only when the CORE domain is ACTIVE. Also, the selected source clock must be enabled by software. Enabling sys_clkout2 does not automatically request the required source clock.

4.7.2.3 Summary

Table 4-22 summarizes the external clock I/O.

Table 4-22. External Clock I/Os

Name	I/O[1]	Source/Destination	Description
sys_xtalin	I	Oscillator	Main input clock. Crystal oscillator clock (only at 26 MHz) or CMOS digital clock (at 26 MHz).
sys_xtalout	O	Oscillator	Output of oscillator
sys_clkout1	O	PRCM	Configurable output clock 1
sys_clkout2	O	PRCM	Configurable output clock 2
sys_32k	I	PRCM	32-kHz clock input
sys_altclk	I	PRCM	Alternate selectable clock source for UARTs or NTSC/PAL. It can be 54 MHz, 48 MHz, or any frequency.

(1) I = Input, O = Output

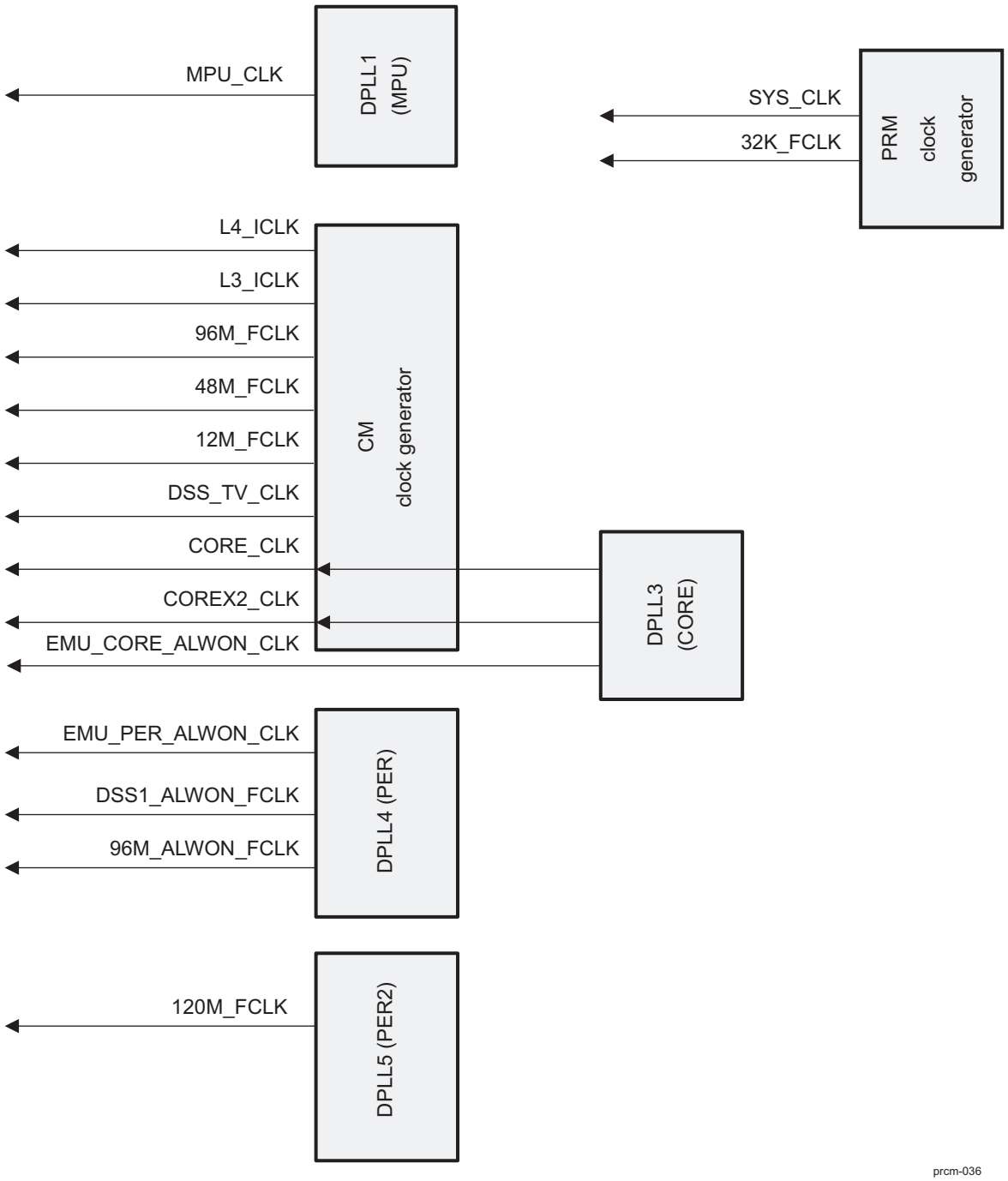
4.7.3 Internal Clock Generation

The device generates internal clocks from four sources:

- PRM
- CM
- DPLLs
- 32-kHz oscillator

Figure 4-22 shows the internal clock generation scheme of the device.

Figure 4-22. Internal Clock Sources



4.7.3.1 PRM

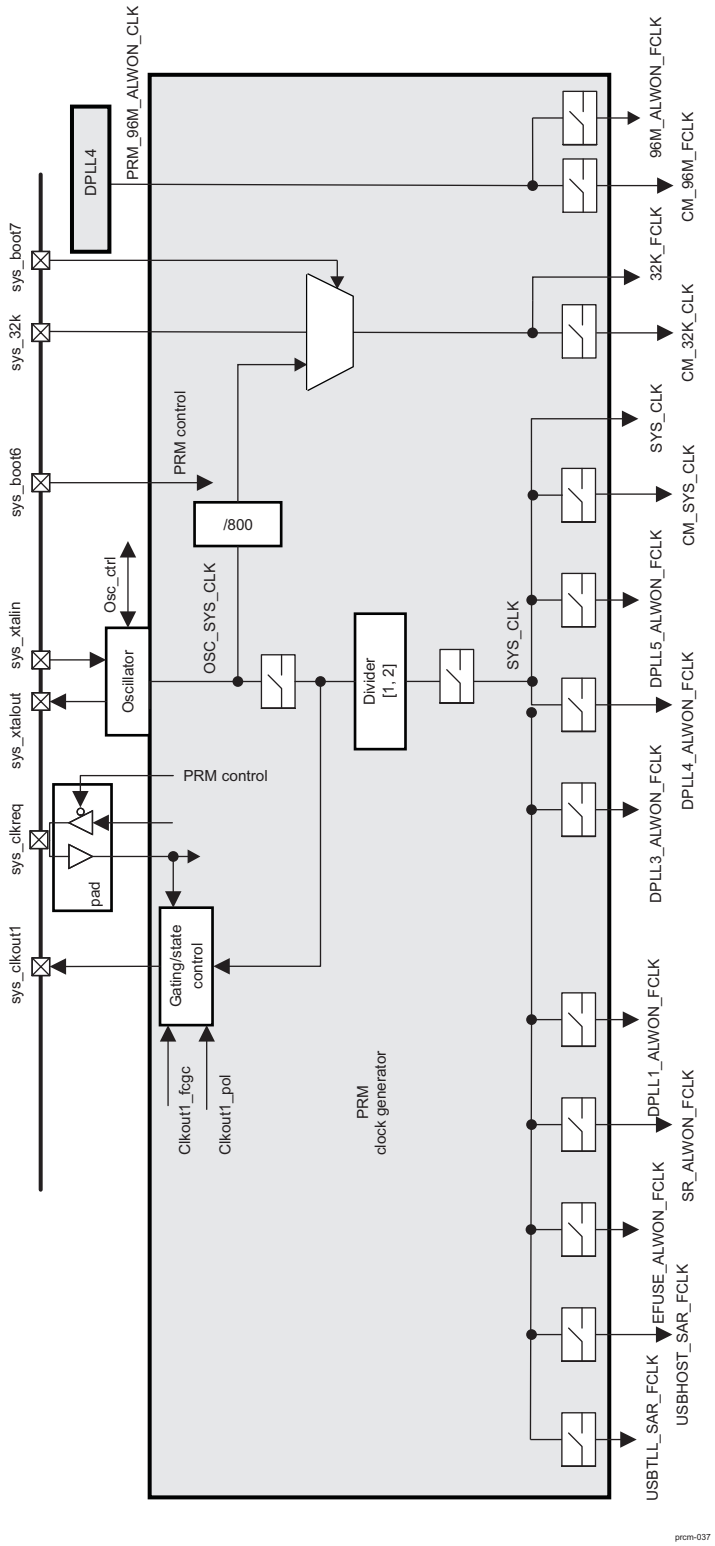
The PRM resides in the WKUP domain. It handles the generation of the 32-kHz low-frequency clocks and the high-frequency system clocks from the SYS_CLK. It also manages the clock oscillator and the external clock output sys_clkout1.

SYS_CLK is generated by the internal oscillator or supplied as the external clock signal on the sys_xtalin pin. It supplies most of the clocks in the device. SYS_CLK is also the source of the WKUP domain interface clocks.

It also handles the gating and distribution of the 96-MHz clock from DPLL4 to the CM and the PER domain modules.

Figure 4-23 is the functional overview of the PRM. The other clocks in the figure are explained in the following sections.

Figure 4-23. PRM Clock Generator



4.7.3.2 CM

The CM clock generator generates interface clocks and peripheral functional clocks for most of the modules. It also controls DPLL3, DPLL4, and the external peripheral clock output sys_clkout2 (see Figure 4-24).

The CM is in the CORE domain.

DPLL3 receives SYS_CLK from the PRM and generates CORE_CLK through the CM. CORE_CLK is the source for the interface clocks (L3 and L4) and the functional clock . The L3 and L4 interface clocks supply the device interconnects and all module interface clocks. The L4 clock is divided to supply the reset managers (PRM) in the WKUP domain. The clocks derived from CORE_CLK are fully balanced over the device.

The 96M_FCLK, 48M_FCLK, and 12M_FCLK clocks are functional unbalanced clocks for a number of modules in the CORE and PER domains.

The functional 96-MHz clock path can be bypassed with SYS_CLK to allow a peripheral such as I^2C to be functional while the DPLL4 is not yet enabled. The default configuration after initial power-on is bypassed with the system clock. Software must switch to the DPLL-generated clock after programming the proper system settings.

The 96-MHz clock input from PRM to the CM (that is, CM_96M_FCLK) is internally gated by the CM.

Figure 4-24 shows the functional overview of the CM.

Figure 4-24. CM Clock Generator Functional Overview

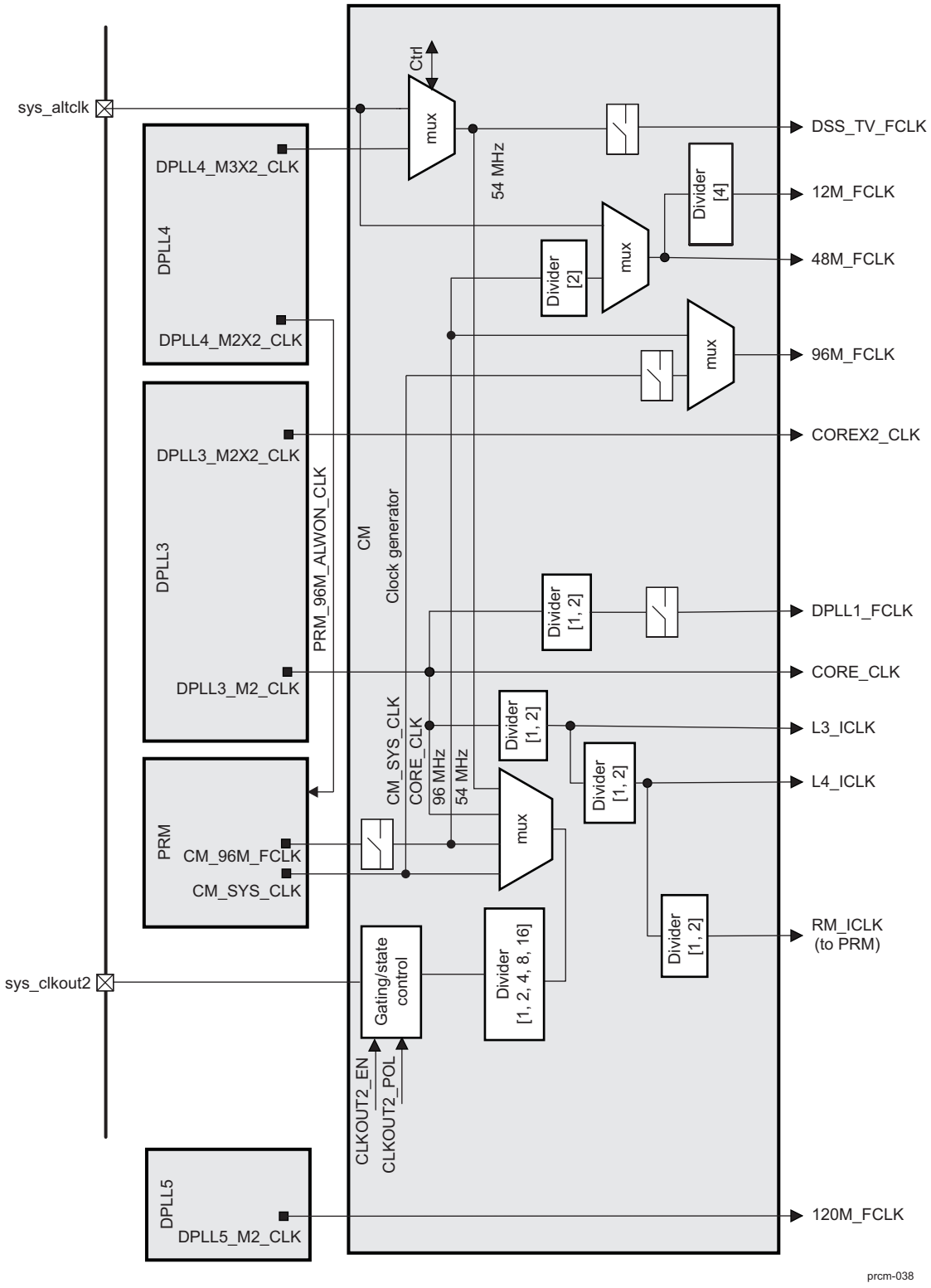
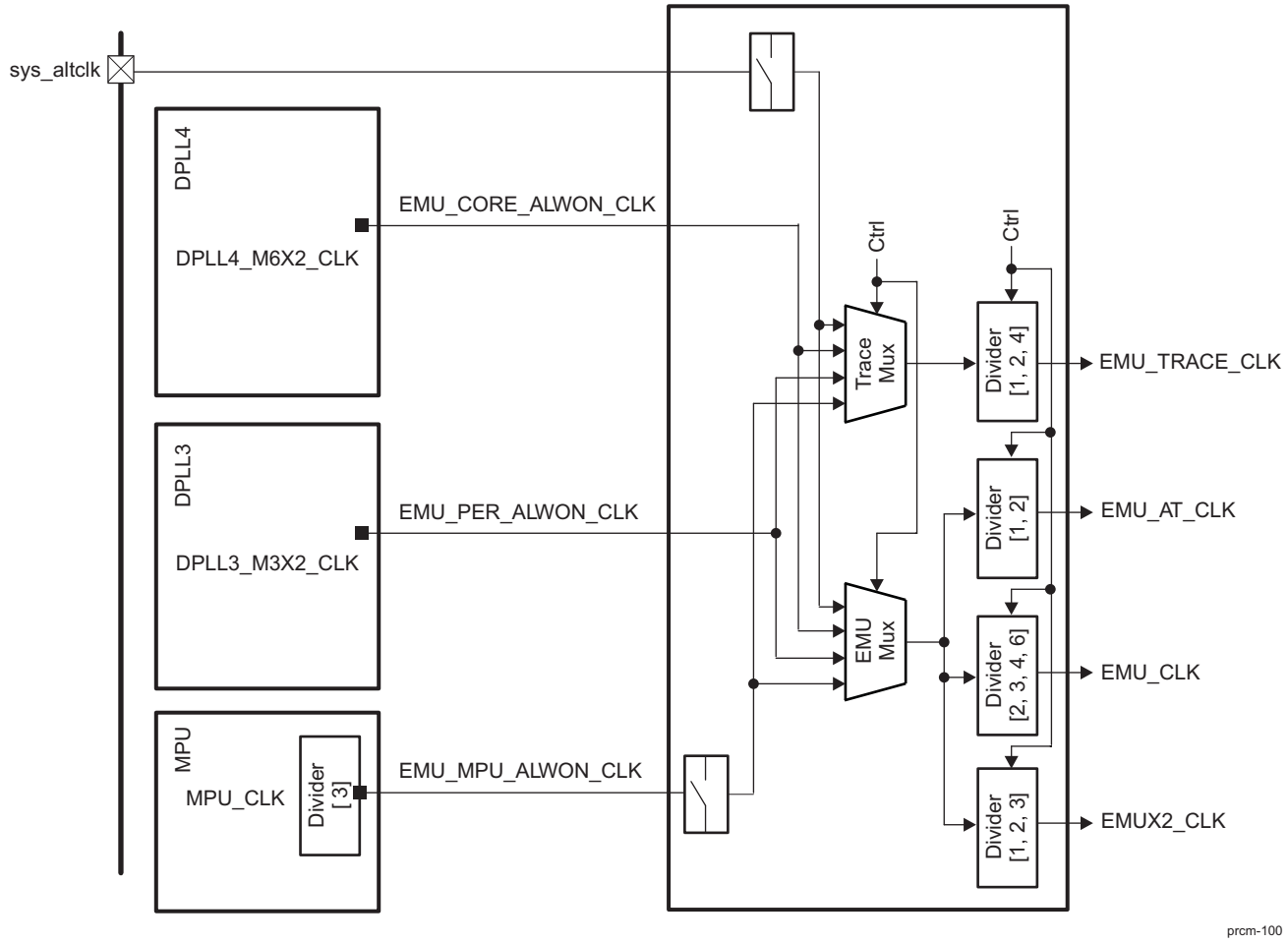


Figure 4-25 shows the functional overview of the CM for emulation clocks.

Figure 4-25. CM Emulation Clock Generator Functional Overview



4.7.3.3 DPLLs

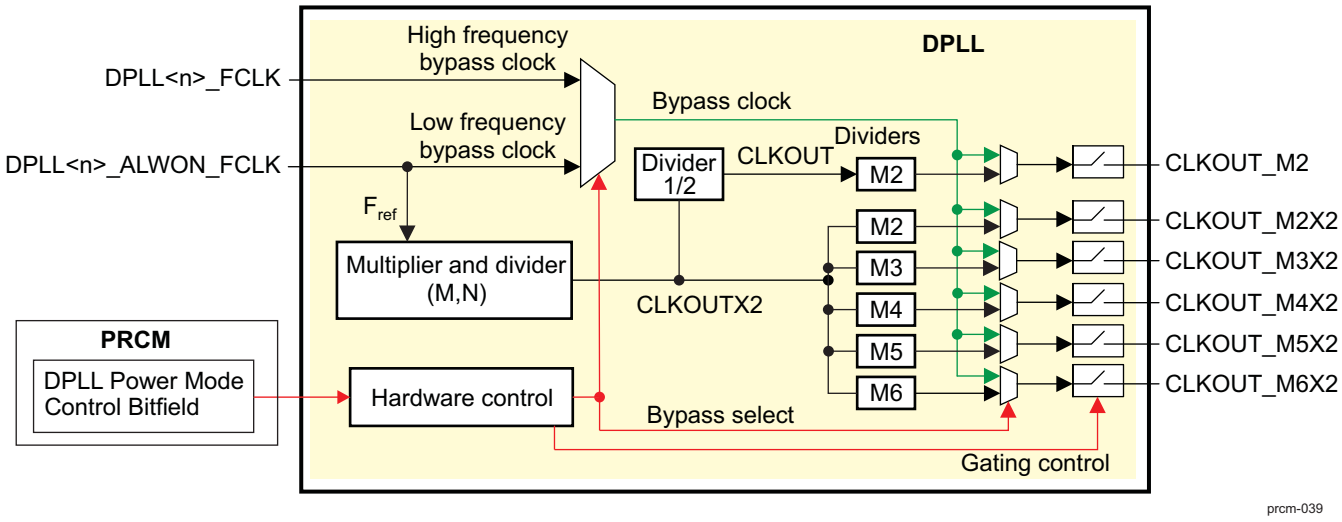
To generate high-frequency clocks, the device supports five on-chip DPLLs controlled directly by the PRCM:

- DPLL1 (MPU)
- DPLL3 (CORE)
- DPLL4 (PER)
- DPLL5 (PER2)

NOTE: This chapter discusses only DPLL1 to DPLL5, because they are directly controlled by the PRCM module. The *Display Interface Subsystem* chapter discusses the DPLLs in the DSS.

Figure 4-26 shows the functional architecture of a generic DPLL.

Figure 4-26. Generic DPLL Functional Diagram



Depending on its hardware configuration, the DPLL may receive one or two clock inputs.

When the DPLL has two clock inputs, it uses one as the reference clock (F_{ref}) to generate the high-frequency clock; the second one serves as the bypass clock when the DPLL is in bypass mode (that is, not locked and generating the high-frequency clock). For example, DPLL1 receives the high-frequency bypass clock from the DPLL3 output, and the reference clock from the PRM.

When the DPLL has only one clock input, it uses that clock input as the reference clock and bypass clock. For example, DPLL3, DPLL4, and DPLL5 receive only one input clock from PRM, and it is used as both the reference and the bypass clock.

It internally generates two main clocks according to the following equations:

- $CLKOUTX2 = (F_{ref} \times 2 \times M) / (N+1)$
- $CLKOUT = CLKOUTX2/2$

where M is an 11-bit multiplier and N is a 7-bit divider.

NOTE: When M is set to 0 or 1, the DPLL is forced into bypass mode.

The internal clocks (CLKOUT and CLKOUTX2) of the DPLL may then be used to generate six independent output clocks:

- CLKOUT_M2 = CLKOUT / M2
- CLKOUT_M2X2 = CLKOUTX2 / M2
- CLKOUT_M3X2 = CLKOUTX2 / M3

- CLKOUT_M4X2 = CLKOUTX2 / M4
- CLKOUT_M5X2 = CLKOUTX2 / M5
- CLKOUT_M6X2 = CLKOUTX2 / M6

where M2, M3, M4, M5, and M6 are additional dividers for the DPLL-synthesized clock.

The output clock frequencies defined by these equations are generated by the DPLL only when it is locked. When the DPLL is in bypass mode, however, all clock outputs run at the bypass clock frequency. The bypass clock can either be a high-frequency bypass clock (only for DPLL1) or the low-frequency reference clock.

The DPLL also provides an independent clock-gating signal for each of the six output clocks. The PRCM provides the DPLL with a clock-gating control signal, and the DPLL returns a clock activity status signal indicating whether the output clock is effectively gated or running.

For an explanation of the DPLL multiplier, divider settings, and gating controls, see Section 4.7.6, *DPLL Control.*

Each clock-generating DPLL of the device has the following features:

- Independent domain
- Control by the CM
- Fed by always-on SYS_CLK with independent gating control for the SYS_CLK
- Analog part supplied by a dedicated power supply and an embedded LDO to eliminate 1-MHz noise
- Up to six independent output dividers for simultaneous generation of multiple output clocks with different frequencies

4.7.3.3.1 DPLL1 (MPU)

DPLL1 is in the MPU subsystem. It supplies clocks to this subsystem that are used as the source clocks for internally generating all subsystem clocks.

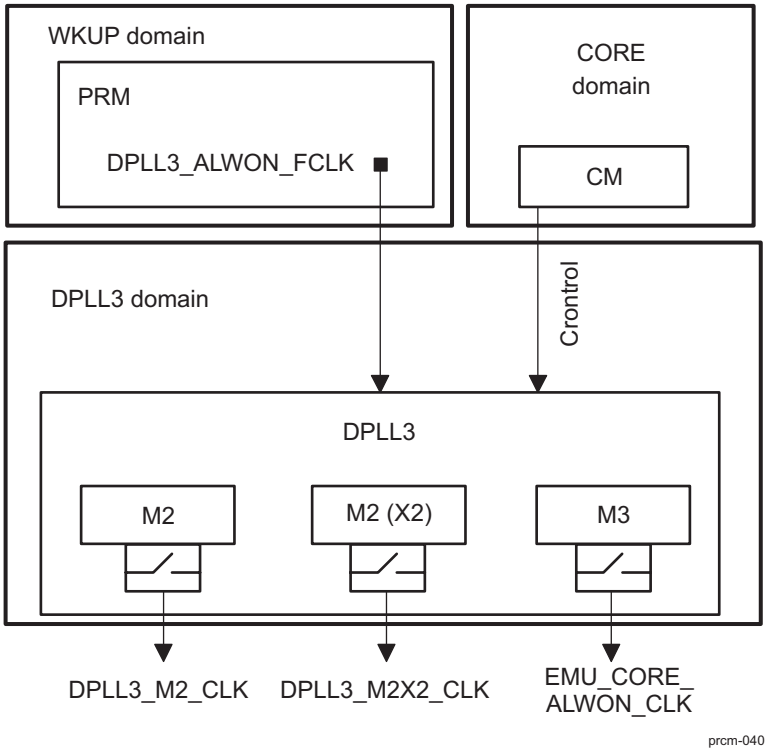
DPLL1 uses a reference clock to produce its synthesized clock. It receives the reference clock (DPLL1_ALWON_FCLK) from the PRM and its high-frequency bypass clock (DPLL1_FCLK) from the CM. The reference clock is SYS_CLK and the high-frequency bypass clock is CORE_CLK.

To save on DPLL power consumption by the processor, the high-frequency bypass input clock from DPLL3 (CORE) is used when the DPLL is set to bypass mode (either statically, or dynamically during relock time) or when the processor is not required to run faster than at L3 clock speed. This use of the high-frequency bypass input clock also optimizes the performance of the DPLLs during frequency scaling.

4.7.3.3.2 DPLL3 (CORE)

Figure 4-27 is the block diagram of DPLL3.

Figure 4-27. DPLL3 Clocks

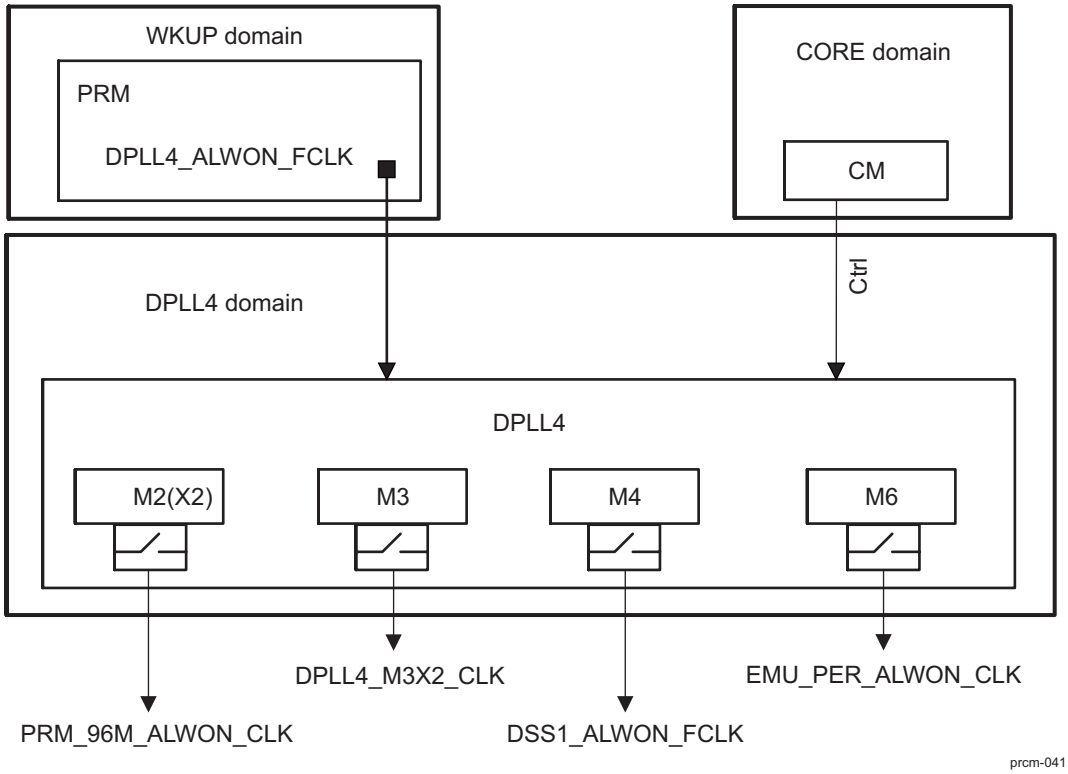


DPLL3 receives its reference clock (DPLL3_ALWON_FCLK), which is the SYS_CLK, from the PRM. DPLL3 does not receive a high-frequency bypass clock, and it uses the reference clock as the low-frequency bypass clock. DPLL3 supplies the source clock for all interfaces and a few functional clocks for the device modules. It also serves as the source of the emulation trace clock. While the CORE domain is on, the output of DPLL3 can be used as HS bypass clock input to DPLL1.

4.7.3.3.3 DPLL4 (Peripherals)

Figure 4-28 is the block diagram of DPLL4.

Figure 4-28. DPLL4 Clocks



DPLL4 receives its reference clock (DPLL4_ALWON_FCLK), which is the SYS_CLK, from the PRM. DPLL4 does not receive a high-frequency bypass clock, and it uses the reference clock as the low-frequency bypass clock. DPLL4 generates clocks for the peripherals, supplying five clock sources:

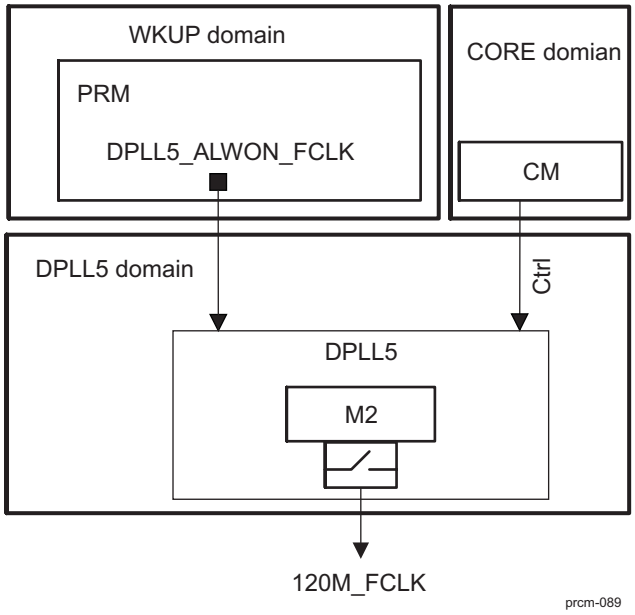
- 96-MHz always-on source clock for the PRM
- 54-MHz to TV DAC
- Display functional clock
- Emulation trace clock

The clock outputs to the DSS, PER, and EMU domains are always on.

4.7.3.3.4 DPLL5 (Peripherals)

Figure 4-29 is the block diagram of DPLL5.

Figure 4-29. DPLL5 Clocks



DPLL5 receives its reference clock (DPLL5_ALWON_FCLK), which is the SYS_CLK, from the PRM. DPLL5 does not receive a high-frequency bypass clock, and it uses the reference clock as the low-frequency bypass clock. DPLL5 generates clocks for the peripherals, supplying five clock sources:

- 120-MHz functional clock to the peripheral domain modules.
- USIM source clock for the functional clock of the USIM open-core protocol (OCP) in the WKUP domain.

4.7.3.3.5 DPLL Clock Summary

Table 4-23 summarizes the use of the divided output clocks of the five DPLLs in the device.

Table 4-23. DPLL Output Clocks

	CLKOUT_M2	CLKOUT_M2X2	CLKOUT_M3X2	CLKOUT_M4X2	CLKOUT_M5X2	CLKOUT_M6X2
DPLL1		X[1]				
DPLL3	X	X	X			
DPLL4		X	X	X	X	X
DPLL5	X					

(1) X represents the DPLL clock output used.

4.7.3.4 32-kHz Oscillator

An internal 32-kHz always-on oscillator feeds the secure watchdog timer (WDTIMER1) and the secure timer (GPTIMER12). It is not software-controllable. The oscillator ensures protection for these timers against clock stoppage caused by an external attack.

4.7.3.5 Summary

Table 4-24 summarizes the source clocks in the device.

Table 4-24. Source-Clock Summary

Clock Name	External/Internal Source	Clock Generator	Description
32K_FCLK	sys_32k (input pin)	PRM	
SYS_CLK	Oscillator	PRM	System clock. Serves as primary source clock of the device. Also used as functional and interface clock for PRM.
DSS_TV_CLK	DPLL4/sys_altclk (input pin)	CM	DSS TV clock
120M_FCLK	DPLL5	CM	
96M_FCLK	DPLL4	CM	
48M_FCLK	DPLL4/sys_altclk (input pin)	CM	
12M_FCLK	DPLL4/sys_altclk (input pin)	CM	
96M_ALWON_CLK		DPLL4	Direct from DPLL4
CORE_CLK	DPLL3	CM	DPLL3 clock output frequency
COREX2_CLK	DPLL3	CM	DPLL3 clock output frequency x 2
L3_ICLK	DPLL3	CM	L3 interconnect interface clock
L4_ICLK	DPLL3	CM	L4 interconnect interface clock
MPU_CLK		DPLL1	MPU subsystem source clock
SECURE_32K_FCLK		32-kHz oscillator	Generated internally by an RC oscillator. It is always active.

4.7.4 Clock Distribution

4.7.4.1 Domain Clock Distribution

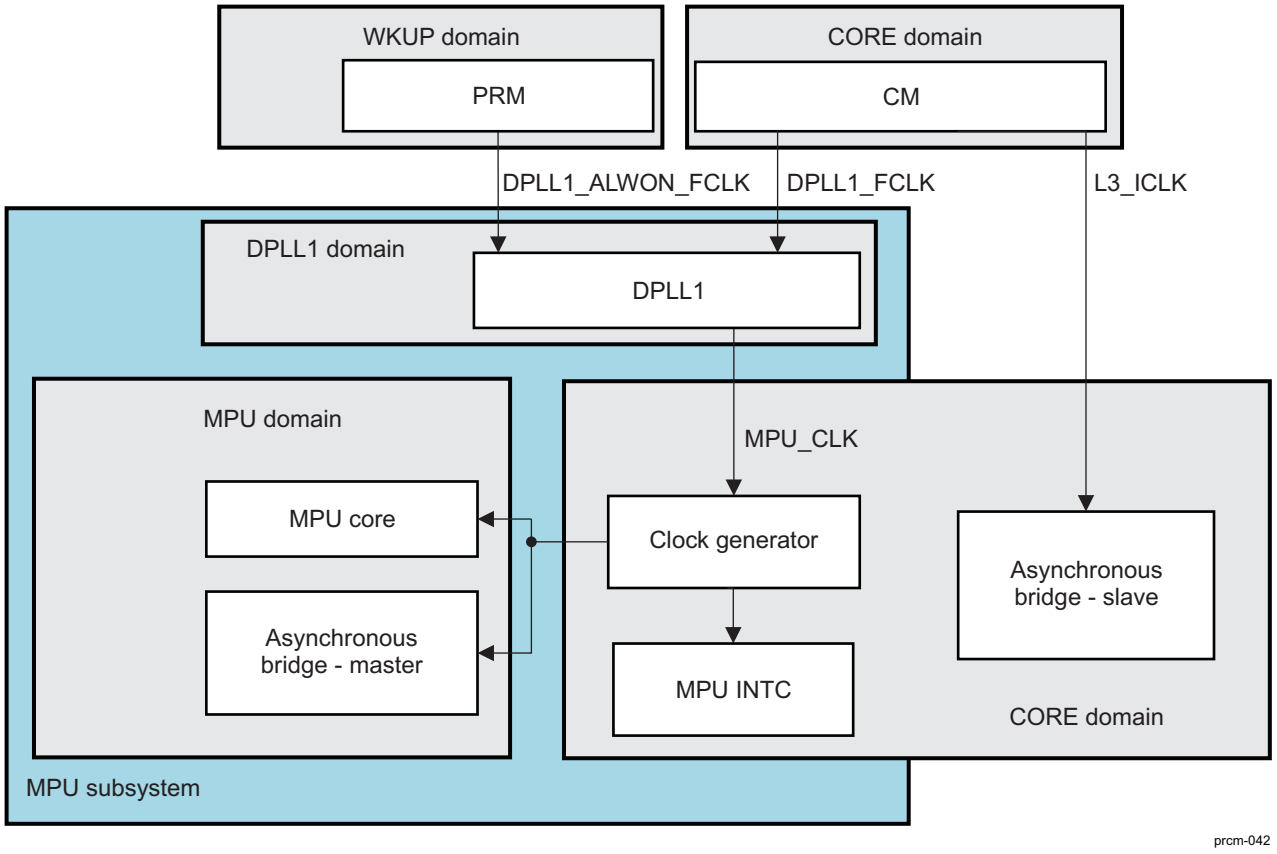
This section describes the PRCM clock distribution over device domains:

4.7.4.1.1 MPU Domain

The PRCM does not directly provide any clock to the MPU domain. It feeds only DPLL1, which generates MPU_CLK. All clocks are then locally generated by the clock generator in the MPU subsystem.

Figure 4-30 shows the clocking scheme in the MPU domain.

Figure 4-30. MPU Domain Clocking Scheme



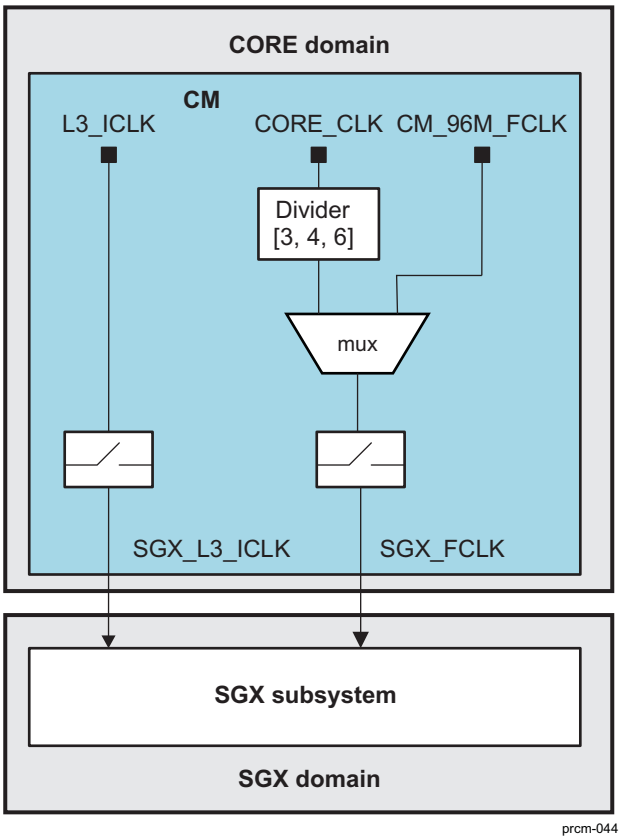
4.7.4.1.2 SGX Domain

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power savings considerations, ensure that clocks to unused modules are properly cut off.

The SGX subsystem interface clock is sourced by the L3 clock, whereas the functional clock source can be selected between CORE_CLK and CM_96M_FCLK. When the functional clock source is CORE_CLK, its frequency can be divided (by 3, 4, or 6).

Figure 4-31 shows the clocking scheme in the SGX domain.

Figure 4-31. SGX Domain Clocking Scheme



4.7.4.1.3 CORE Domain

The CORE domain has both L3- and L4-derived clock domains.

The CORE domain receives several functional clocks (12-, 48-, 96-MHz, system, and 32-kHz) that feed its peripherals and modules, with an exception:

- The McBSP 1 and McBSP 5 modules can be clocked either by CORE_96M_FCLK from the CM or from an external clock, MCBSP_CLKS. The SCM manages the selection between the two sources. For more information about the SCM, see *System Control Module* chapter.

Figure 4-32 through Figure 4-34 show the clock signals and their relationships in the CORE domain.

Figure 4-32. CORE Clock Signals: Part 1

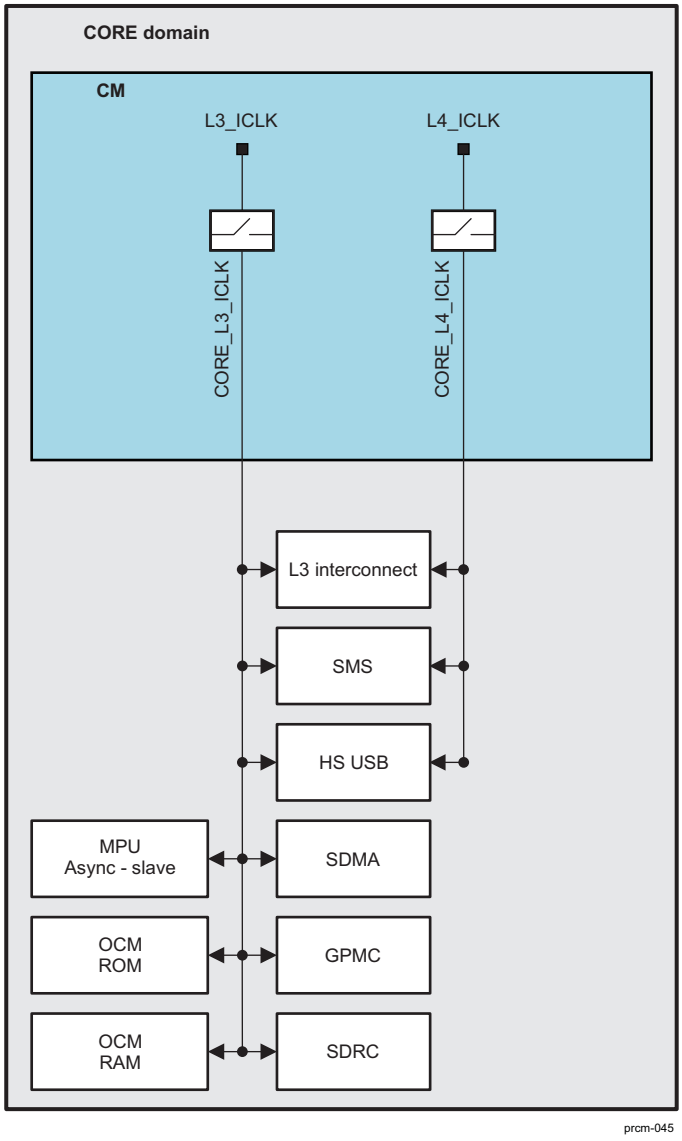


Figure 4-33. CORE Clock Signals: Part 2

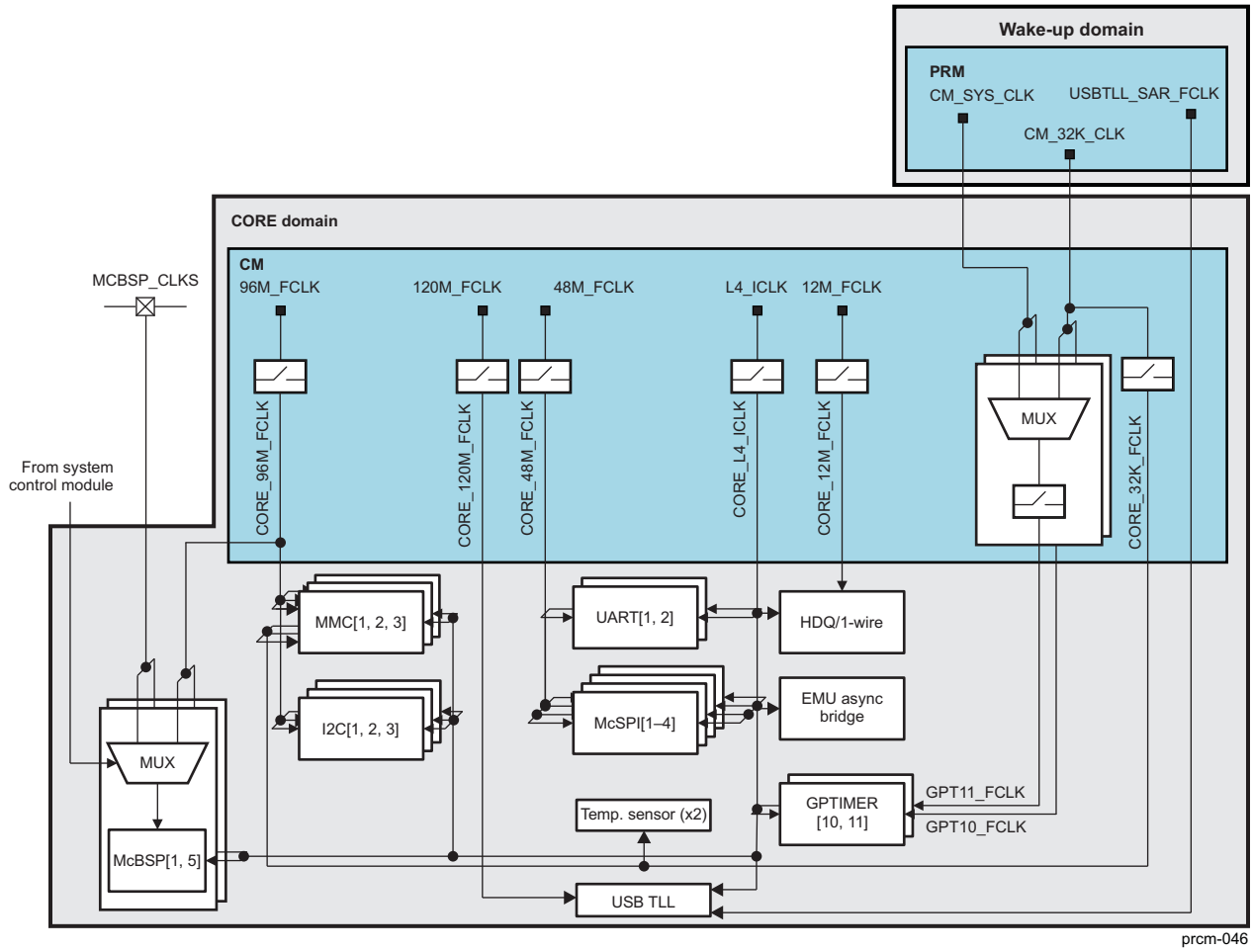
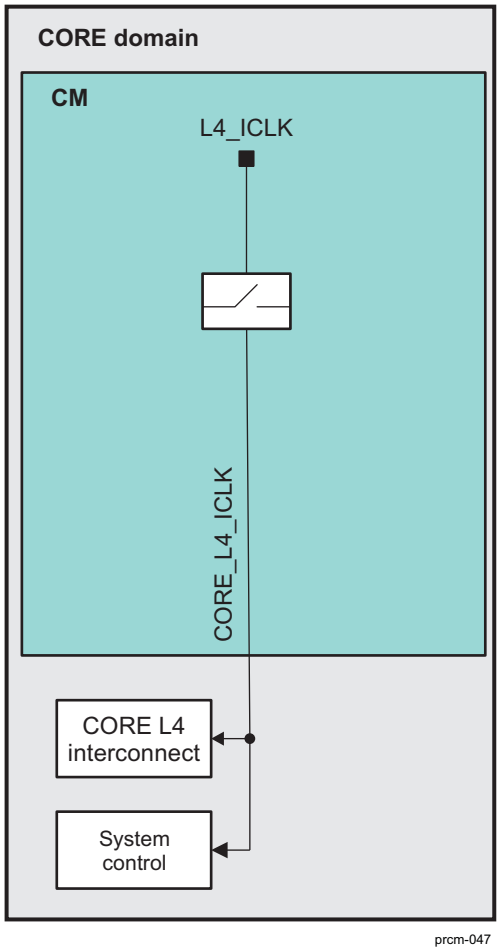


Figure 4-34. CORE Clock Signals: Part 3



4.7.4.1.4 IPSS Domain

The IPSS domain receives following clocks from the system:

- **L3_ICLK:** interface clock that feeds VBUS, VBUS2OCP and OCP2VBUS architecture of IPSS. This clock is used as interface clock for all the IPSS modules.
- **SYS_CLK:** used as functional clock for HECC module and as source clock for USBOTG SS.

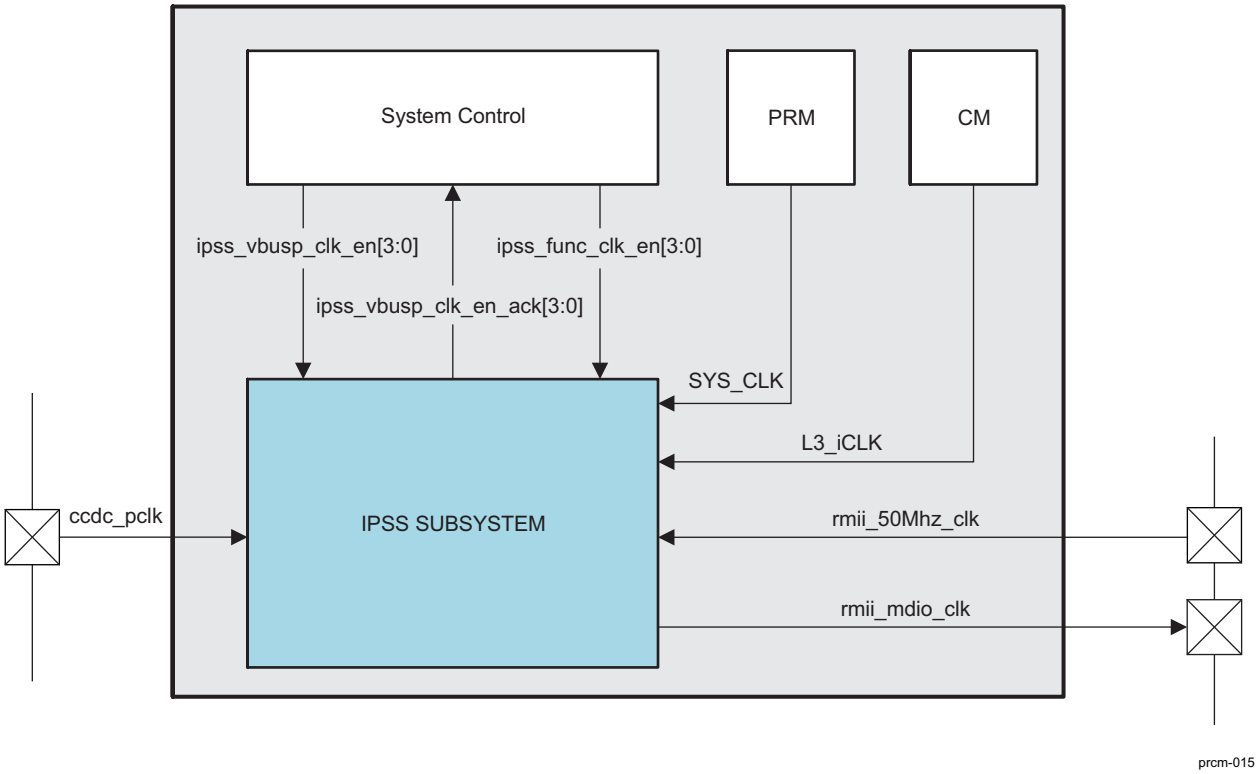
External clocks include:

- rmii_50mhz_clk: This clock feeds EMAC module for RMII functionality. (Refer to the EMAC chapter for more information on EMAC and RMII clocks.)
- rmii_mdio_clk: This is the output clock coming from EMAC which is used for external PHY.
- ccdc_pclk: This is the VPFE pixel clock input from the external sensor. (Refer to the VPFE chapter for more information on ccdc_pclk.)

The CONTROL_IPSS_CLK_CTRL register controls the clocking for IPSS. The field ipss_vbusp_clk_en has the gating control for interface clock and ipss_func_clk_en field will switch ON the functional clock for the IPSS modules (VPFE, USBOTG, HECC and EMAC). Few of the modules use their interface clock as functional clock, for more details refer to corresponding modules chapter.

Software can program ipss_vbusp_clk_en register bits to put corresponding modules in IDLE mode. It needs to read ipss_vbusp_clk_en_ack bits to make sure that these module clocks are gated. (Refer to the System Control Module chapter for more details.)

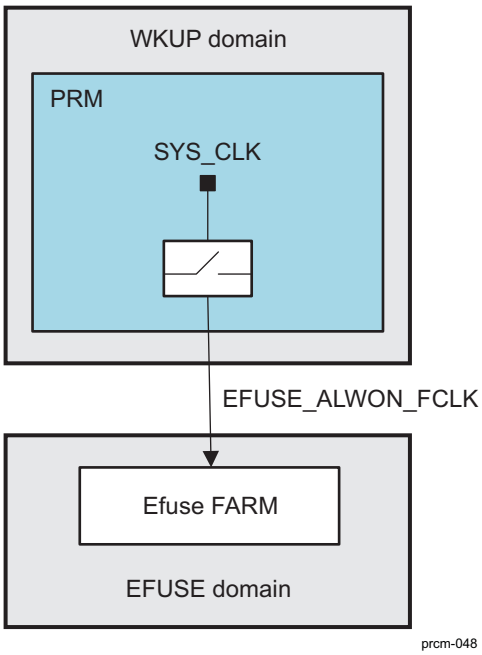
Figure 4-35. IPSS Domain



4.7.4.1.5 EFUSE Domain

Figure 4-36 shows the clock signals and their relationships in the EFUSE domain.

Figure 4-36. EFUSE Clock Signals



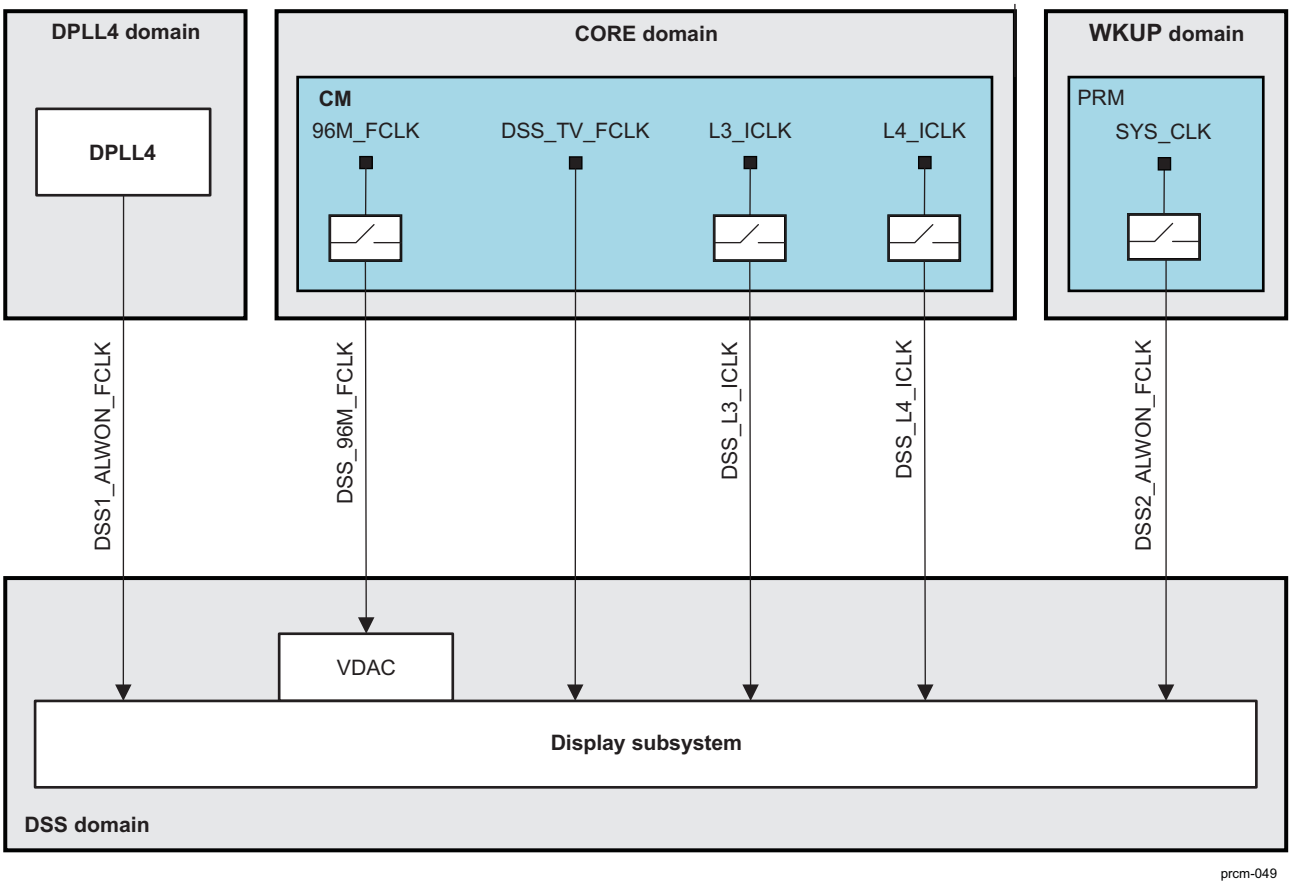
The EFUSE sense procedure is performed at device power up. During this procedure, the PRM enables the EFUSE_ALWON_FCLK clock.

4.7.4.1.6 DSS Domain

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power savings considerations, ensure that clocks to unused modules are properly cut off.

Figure 4-37 shows the clock signals and their relationships in the DSS domain.

Figure 4-37. DSS Clock Signals



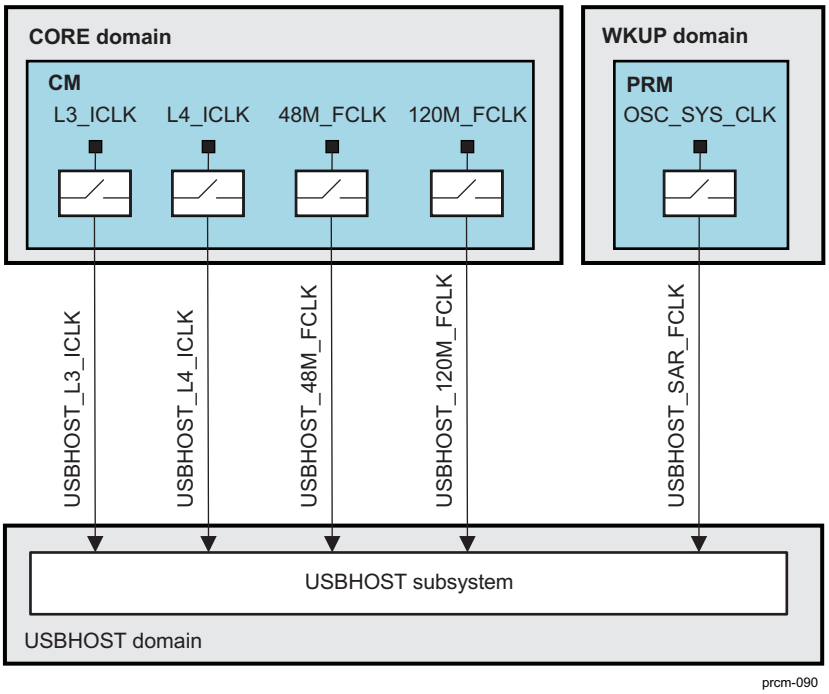
The DSS subsystem interface is clocked with the L3 and L4 clocks. It receives four functional clocks:

- DSS1_ALWON_FCLK: Issued from DPLL4. Its frequency can be a division by 1 to 16 of the frequency of the DPLL4 synthesized clock.
- DSS2_ALWON_FCLK: The gated SYS_CLK. Used mainly for display in low-power refresh modes.
- DSS_96M_FCLK: Required when TV output is activated
- DSS_TV_FCLK: Required when TV output is activated

4.7.4.1.7 USBHOST Domain

Figure 4-38 shows the clock signals and their relationships in the USBHOST domain.

Figure 4-38. USBHOST Clock Signals



The HS USB Host subsystem interface is clocked with the L3 and L4 clocks (USBHOST_L3_ICLK and USBHOST_L4_ICLK, respectively).

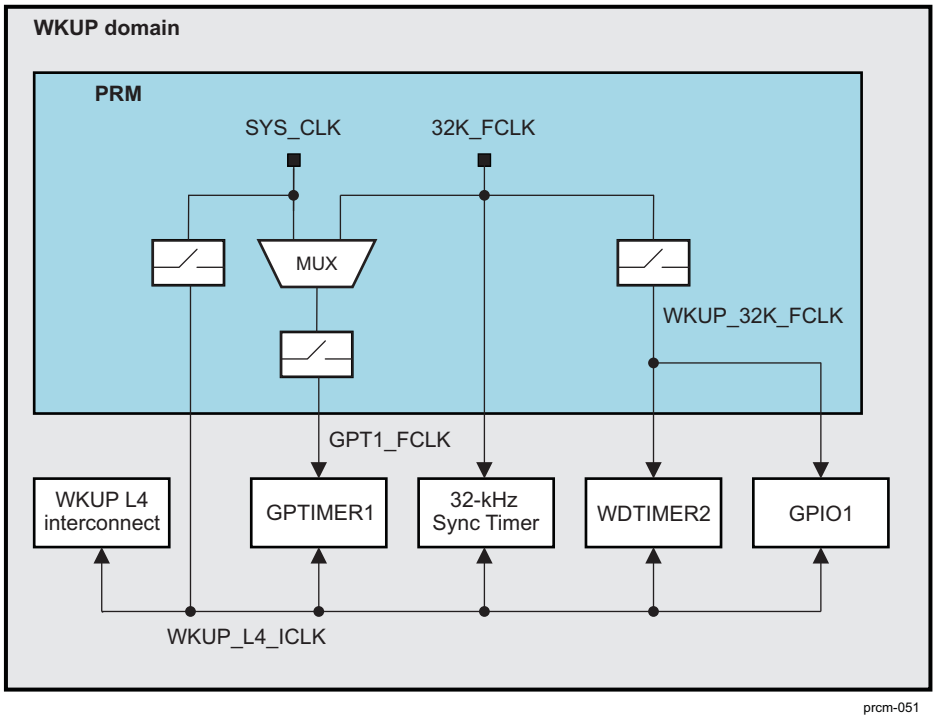
The HS USB Host subsystem requires two functional clocks (USBHOST_120M_FCLK and USBHOST_48M_FCLK) that may or may not be requested simultaneously. Therefore, they are gated independently based on the configuration of the CM_FCLKEN_USBHOST[0] EN_USBHOST1 and CM_FCLKEN_USBHOST[1] EN_USBHOST2 bits.

The HS USB Host subsystem gets an additional functional clock from the PRM (USBHOST_SAR_FCLK). It is dedicated to the save-and-restore mechanism and is automatically gated/enabled by the PRM, based on the HS USB Host save-and-restore bit configuration (that is, the PM_PWSTCTRL_USBHOST[4] SAVEANDRESTORE bit) and on the USBHOST domain state transitions.

4.7.4.1.8 WKUP Domain

Figure 4-39 shows the clock signals and their relationships in the WKUP domain.

Figure 4-39. WKUP Clock Signals



All clocks in the WKUP domain are generated by the PRM, except for the functional clock for the secure timers (WDTIMER1 and GPTIMER12). This clock is supplied directly by the internal 32-kHz oscillator (SECURE_32K_FCLK). The functional clock GPT1_FCLK of GPTIMER1 can be selected as either SYS_CLK or 32K_FCLK. The 32-kHz sync timer, WDTIMER2, and GPIO1 receive 32K_FCLK as their functional clock. This is the low-frequency always-on clock.

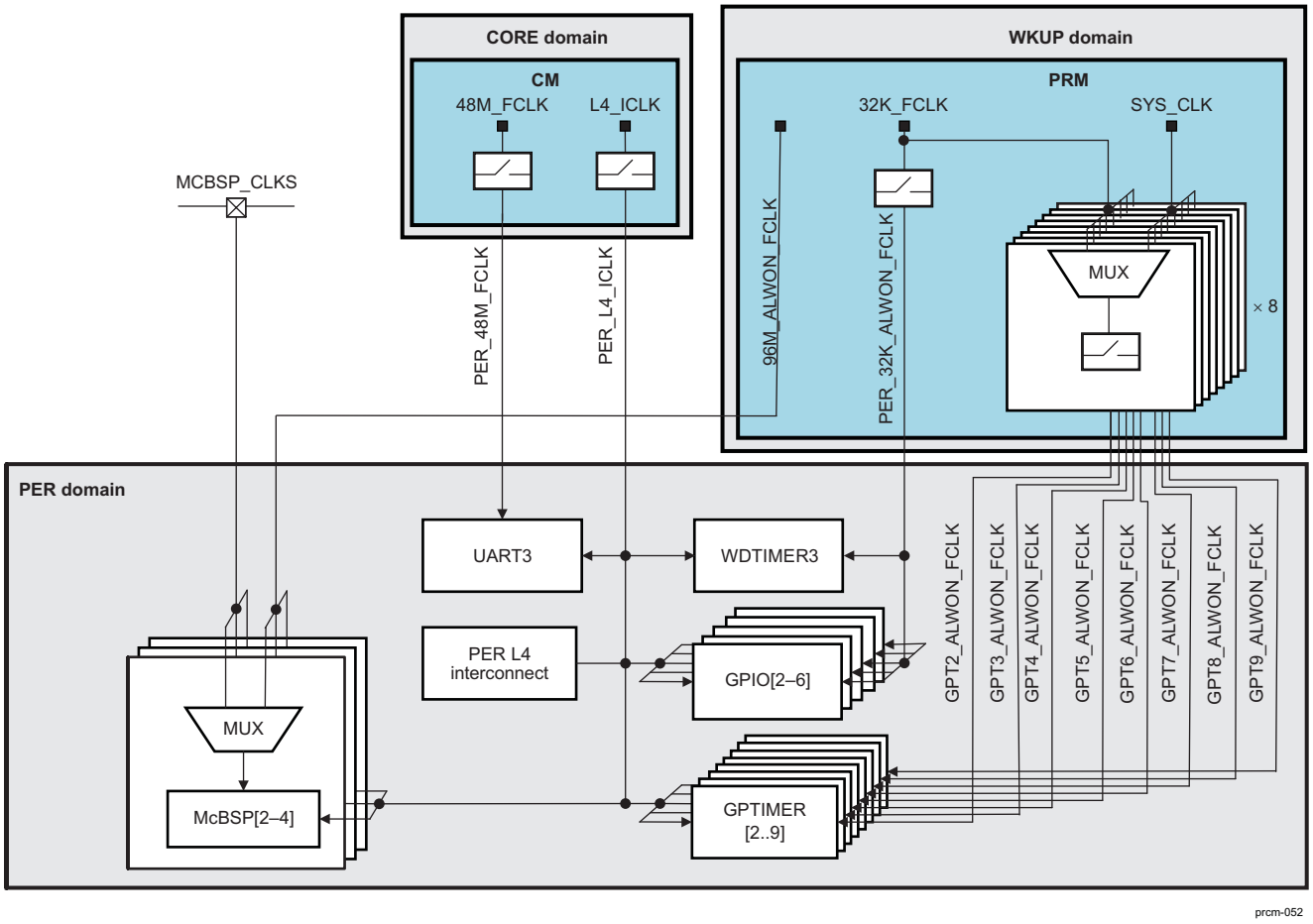
The PRM receives SYS_CLK as the L4 interface clock. For all other modules of the WKUP domain, the L4 interface clock WKUP_L4_ICLK is derived from SYS_CLK. Communication between the WKUP domain and CORE L4 interconnects is asynchronous.

The USIM OCP module receives the functional clocks (USIM_FCLK and 32K_FCLK) and the L4 interface clock (WKUP_L4_ICLK) from the PRM.

4.7.4.1.9 PER Domain

Figure 4-40 shows the clock signals and their relationships in the PER domain.

Figure 4-40. PER Clock Signals



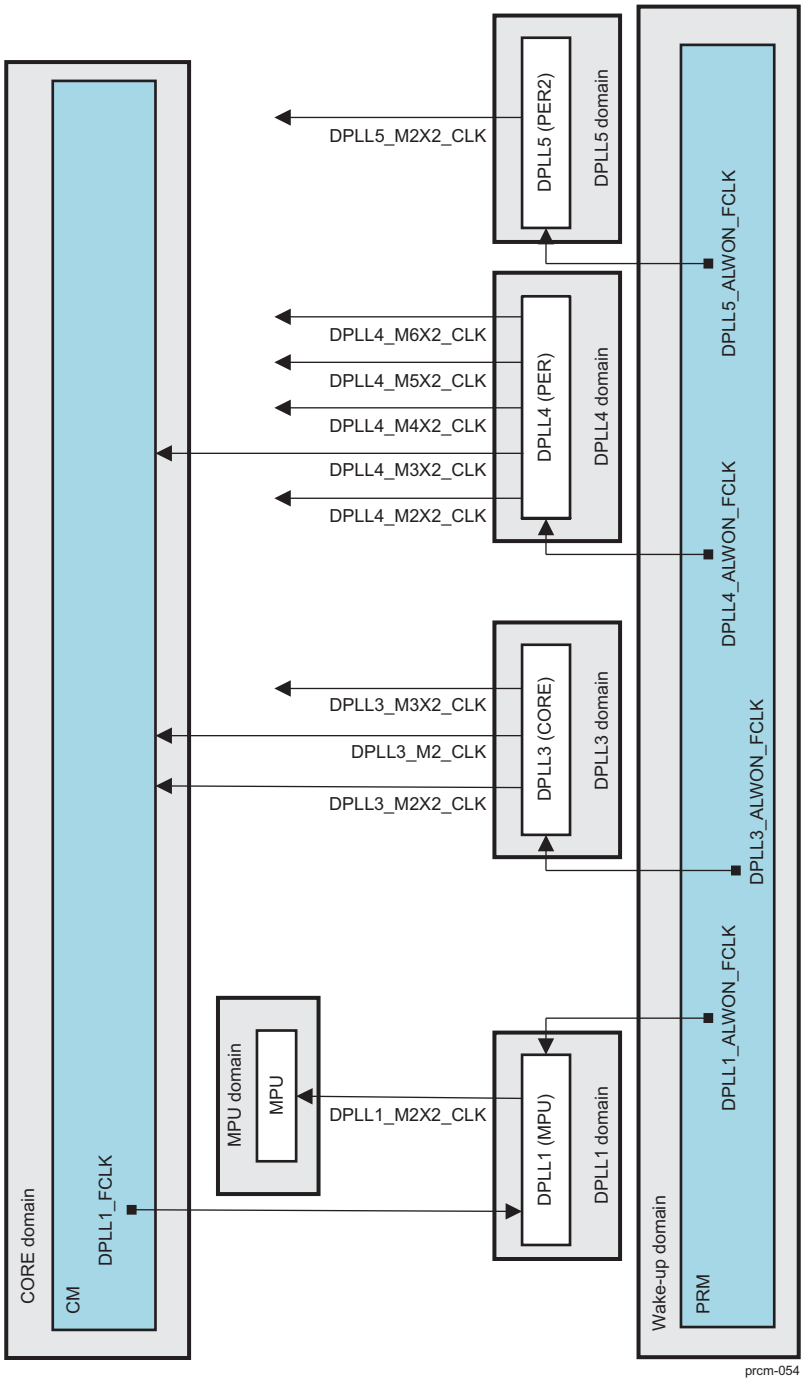
The PER domain receives several functional clocks (48M_FCLK, 96M_ALWON_FCLK, SYS_CLK, and 32K_FCLK) that feed its peripherals and modules. All the functional clocks (except 48M_FCLK) are permanently supplied so that the peripherals can be used during low-power scenarios. Figure 4-40 shows the clock distribution scheme in the PER domain.

The McBSP 2, 3, and 4 modules can be clocked either by a clock from PRM (PER_96M_FCLK) or from an external clock (MCBSP_CLKS). This clock must be permanently buffered from the pad to the PER domain. The device SCM manages the selection between the two (see *System Control Module* chapter).

4.7.4.1.10 DPLL Domains

The PRCM provides clock sources for the DPLLs, as shown in Figure 4-41

Figure 4-41. DPLL Clock Signals



The PRCM also manages clock-gating control for these DPLL outputs.

The MPU uses clock outputs locally in the subsystem. The CM uses the DPLL3 clock outputs to generate all interface clocks and some functional clocks. The CM uses two of the five clocks generated by DPLL4 and one clock output of DPLL5 to generate functional clocks for the peripheral domain modules. The remaining three clocks of DPLL4 are propagated to their corresponding modules.

4.7.4.2 Clock Distribution Summary

4.7.4.2.1 Domain Source Clocks

Table 4-25 summarizes clock distribution for each domain.

Table 4-25. Clock Distribution

Power Domain	Clock	Generator	Destination
MPU	MPU_CLK	DPLL1	MPU subsystem
NEON			NEON
SGX	SGX_FCLK	CM	SGX subsystem
	SGX_L3_ICLK	CM	
CORE	CORE_ 120M_FCLK	CM	USB TLL
	CORE_ 96M_FCLK	CM	McBSP[1,5], MMC[1,2,3], I2C[1,2,3]
	CORE_48M_FCLK	CM	UART[1,2], McSPI[1..4]
	CORE_12M_FCLK	CM	HDQ
	GPT10_FCLK	CM	GPTIMER10
	GPT11_FCLK	CM	GPTIMER11
	CPEFUSE_FCLK	CM	CPEFUSE
	USBTLL_SAR_FCLK	PRM	USB TLL
	CM_32K_CLK	PRM	Temperature sensor (x2), MMC[1,2,3]
	CORE_L3_ICLK	CM	L3 interconnect, SDMA, MPU Async Bridge(Slave), USB20OTGSS, SMS, GPMC, OCM ROM, SDRC, OCM RAM, CORE L3 interconnect
	CORE_L4_ICLK	CM	L3 interconnect, SDMA, McBSP[1,5], MMC[1,2], I2C[1..3], GPTIMER [10,11], UART[1,2], McSPI[1..4], HDQ, CORE L4 interconnect, SCM
DSS	DSS_TV_FCLK	CM	DSS, VDAC
	DSS_96M_FCLK	CM	VDAC
	DSS1_ALWON_FCLK	DPLL4	DSS
	DSS2_ALWON_FCLK	PRM	
	DSS_L3_ICLK	CM	
	DSS_L4_ICLK	CM	
PER	96M_ALWON_FCLK	PRM	McBSP[2..4]
	PER_48M_FCLK	CM	UART3
	PER_32K_ALWON_FCLK	CM	WDTIMER3, GPIO[2..6],
	GPT2_ALWON_FCLK	PRM	GPTIMER2
	GPT3_ALWON_FCLK	PRM	GPTIMER3
	GPT4_ALWON_FCLK	PRM	GPTIMER4
	GPT5_ALWON_FCLK	PRM	GPTIMER5
	GPT6_ALWON_FCLK	PRM	GPTIMER6
	GPT7_ALWON_FCLK	PRM	GPTIMER7
	GPT8_ALWON_FCLK	PRM	GPTIMER8
	GPT9_ALWON_FCLK	PRM	GPTIMER9

Table 4-25. Clock Distribution (continued)

Power Domain	Clock	Generator	Destination
	PER_L4_ICLK	CM	UART3, PER L4 interconnect, WDTIMER3, GPIO[2..6], GPTIMER[2..9], McBSP[2..4]
WKUP	WKUP_32K_FCLK	PRM	WDTIMER2, GPIO1
	SECURE_32K_FCLK	32-kHz oscillator	WDTIMER1, GPTIMER12
	32K_FCLK	PRM	32-kHz sync timer, USIM OCP
	GPT1_FCLK	PRM	GPTIMER1
	USIM_FCLK	CM	USIM OCP
	WKUP_L4_ICLK	PRM	WKUP L4 interconnect, GPTIMER[1,12], 32-kHz sync timer, GPIO1, WDTIMER[1,2], USIM OCP
EFUSE	EFUSE_ALWON_FCLK	PRM	eFuse farm
DPLL1	DPLL1_ALWON_FCLK	PRM	DPLL1
	DPLL1_FCLK	CM	
DPLL3	DPLL3_ALWON_FCLK	PRM	DPLL3
DPLL4	DPLL4_ALWON_FCLK	PRM	DPLL4
DPLL5	DPLL5_ALWON_FCLK	PRM	DPLL5

NOTE:

- Modules supplied by the L3 interface clock only:
  - MPU asynchronous bridge
  - All memory controllers (OCM ROM, OCM RAM, SDRC, SMS, and GPMC)
- Modules that require both L3 and L4 clocks:
  - SDMA
  - HS USB
  - All security modules (FPKA, AES, RNG, SHAM, and D3D)
- Modules fed by the L4 clock:
  - SCM
  - All peripherals (McBSP1, McBSP5, MMC1, MMC2, I2C1, I2C2, I2C3, McSPI1, McSPI2, McSPI3, McSPI4, UART1, UART2, HDQ, GPTIMER10, and GPTIMER11)

4.7.4.2.2 Peripheral Module Clocks

Table 4-26 lists the peripherals and DSS functional clock frequency requirements.

Table 4-26. Peripheral Module Functional Clock Frequencies

Module	Functional Clock	Frequency
MMC-SDIO[1,2,3]	96M_FCLK	96 MHz
McBSP[1, 5]	96M_FCLK	96 MHz
McSPI[1..4]	CORE_48M_FCLK	48 MHz
UART[1..3]		
Display subsystem	DSS1_ALWON_FCLK	Up to 173 MHz
	DSS2_ALWON_FCLK	System clock
	DSS_96M_FCLK	96 MHz
	DSS_TV_FCLK	54 MHz
I2C[1..3]	CORE_96M_FCLK	96 MHz
HDQ	CORE_12M_FCLK	12 MHz
GPTIMER1	GPT1_FCLK	32-kHz (p) or system clock
GPTIMER[2..9]	GPTn_ALWON_FCLK	32-kHz (p) or system clock

Table 4-26. Peripheral Module Functional Clock Frequencies (continued)

Module	Functional Clock	Frequency
GPTIMER[10, 11]	GPTn_FCLK	32-kHz or system clock
GPTIMER12	SECURE_32K_FCLK	32 kHz (p) (s)
FPKA1	SECURITY_L3_ICLK	L3_ICLK
AES 1	SECURITY_L4_ICLK2	L4_ICLK
D3D 1		
SHAM 1		
RNG1		
AES 2	CORE_L4_ICLK	L4_ICLK
SHAM 2		
D3D 2		
WDTIMER1	SECURE_32K_FCLK	32 kHz (p) (s)
WDTIMER2	WKUP_32K_FCLK	32 kHz
WDTIMER3	PER_32K_ALWON_FCLK	32 kHz
GPIO1	WKUP_32K_FCLK	32 kHz
GPIO[2-6]	PER_32K_ALWON_FCLK	32 kHz
32-kHz sync timer	32K_FCLK	32 kHz (p)
Bandgap/temp sensor	32K_FCLK	32 kHz (p)
System control	CORE_L4_ICLK	L4_ICLK

4.7.5 External Clock Controls

Because the use of sys_32k and sys_altclk was discussed previously (see Section 4.7.3.1, *PRM*, and Section 4.7.3.2, *CM*), these clock signals are not discussed here. This section discusses the remaining external clock signals.

4.7.5.1 Clock Request (sys_clkreq) Control

The system clock request signal sys_clkreq is bidirectional. In bypass mode in the system clock oscillator (see Section 4.7.5.2), it is an output signal driven by the device to request an external clock. In this case, the output buffer is driven as long as the system clock is requested by the device; otherwise, it remains in high impedance. In this way, other external peripherals can share the same clock request signal with the device.

If PRM_POLCTRL.CLKREQ_POL = 1, the software must configure the SCM to select the internal pulldown on the sys_clkreq pad, or an external pulldown is connected to the pad. If PRM_POLCTRL.CLKREQ_POL = 0, the internal pull-up on the sys_clkreq pad, or an external pull-up is connected to the pad.

In master mode in the system clock oscillator (see Section 4.7.5.2), sys_clkreq is an input. If PRM_POLCTRL.CLKREQ_POL = 1, the software must configure the SCM to select the internal pulldown on the sys_clkreq pad, or an external pulldown is connected to the pad. If PRM_POLCTRL.CLKREQ_POL = 0, the internal pull-up on the sys_clkreq pad, or an external pull-up is connected to the pad.

The PRCM.PRM_POLCTRL[1] CLKREQ_POL bit allows software control over the polarity of sys_clkreq. This software setting takes effect when the clock is requested by the device and also when the clock request is driven externally. The output buffer is directly driven by this register when the clock request comes from the device.

Table 4-27 shows the bidirectional control of the sys_clkreq Pad:

Table 4-27. sys_clkreq Pad Direction Control

					Description
Oscillator Mode	Sys_boot6	Internal Clock request (always active high)	External Clock request (Note: polarity depends on CLKREQ_POL)	Sys_clkreq Direction	
Master Mode	0	0	0[1]	Input (Output buffer in Hi-Z) Note: (Input is not driven from outside of the device in that case)	The clock is neither requested internally nor externally (external device/peripheral)
	0	0	1[1]	Input (Output buffer in Hi-Z)	The clock is requested externally
	0	1	0[1]	Output	The clock is requested internally
	0	1	1[1]	Output Note: (The pad is driven both by the device and from outside of the device in taht case.)	The clock is requested both internally and externally
Bypass Mode	1	0	0[1]	Input (Output buffer in Hi-Z) Note: (Input is not driven from outside of the device in that case.)	The clock is neither requested internally nor externally
	1	0	1[2]	Input (output buffer in Hi-Z)	The clock is requested externally
	1	1	0[2]	Output	The clock is requested internally
	1	1	1[2]	Output Note: (Input is not driven from outside of the device in that case.)	The clock is requested both internally and externally

(1) Case when PRM_POLCTRL.CLKREQ_POL = 1 (sys_clkreq active high). These values would be inverted in the table above, in case PRM_POLCTRL.CLKREQ_POL = 0 (sys_clkreq active low).

(2) Case when PRM_POLCTRL.CLKREQ_POL = 1 (sys_clkreq active high). These values would be inverted in the table above, in case PRM_POLCTRL.CLKREQ_POL = 0 (sys_clkreq active low).

4.7.5.2 System Clock Oscillator Control

Depending on the hardware configuration, the device can receive the system clock from an external source or generate it locally using the internal system clock crystal oscillator. Thus, the device oscillator has two possible operating modes:

- Master (oscillator enable) mode: The oscillator is enabled and connected to an external quartz. It provides the system clock to the device. The oscillator is activated on a device wake-up or on an external clock request.
- Bypass (oscillator inactive) mode: The system clock is supplied by an external device and the oscillator is always set in bypass mode. The oscillator is insensitive to the external system clock request on the sys_clkreq pin.

NOTE: An external pullup or pulldown tied on the sys_boot6 input pin of the device determines whether the oscillator is in master or bypass mode. See Section 4.3.1, *External Clock Signals*.

When operating in master mode, the device receives an external clock request (sys_clkreq) and provides the system clock to external peripherals through the sys_clkout1 pin; in bypass mode, the device generates a clock request to the external clock source to request the system clock.

The selected mode of the oscillator can be read from the PRCM.PRM_CLKSRC_CTRL[1:0] SYSCLKSEL bit field.

Whatever the oscillator mode, the oscillator can be powered down when the device enters inactive mode, unless an external system clock request is active (from the sys_clkreq pin). Note that this option can only be used if the 32kHz clock is not derived from the high frequency sysclk, as the 32kHz clock is required for autonomous wakeup capability. This setting is configured in the PRCM.PRM_CLKSRC_CTRL[4:3] AUTOEXTCLKMODE bit field. Table 4-29 lists the four possible operation modes of the system clock.

Table 4-28. System Clock Operation Modes

AUTOEXTCLKMODE	System Clock Mode	Oscillator Mode	Description
0x0	Always-active mode	Master	The oscillator is kept active even when the clock is neither requested by the device internally (all device clocks are inactive) nor externally (that is, the sysclkreq input signal is not asserted).
		Bypass	The sys_clkreq output signal is permanently asserted by the device, regardless of its internal clocks state (active or inactive).
0x1	Off when device in INACTIVE state	Master	The oscillator is switched off when the device is in INACTIVE mode and the sys_clkreq input signal is not asserted.
		Bypass	The sys_clkreq output signal is de-asserted when the device is in idle mode.

To exit power-down mode, the oscillator requires a device wakeup or an external clock request.

The device allows configuring of the system clock stabilization time to ensure a stable system clock in the device. This delay is configured in the PRCM.PRM_CLKSETUP[15:0] SETUP_TIME bit field.

Figure 4-42 shows the system clock oscillator controls in the device.

Figure 4-42. System Clock Oscillator Controls

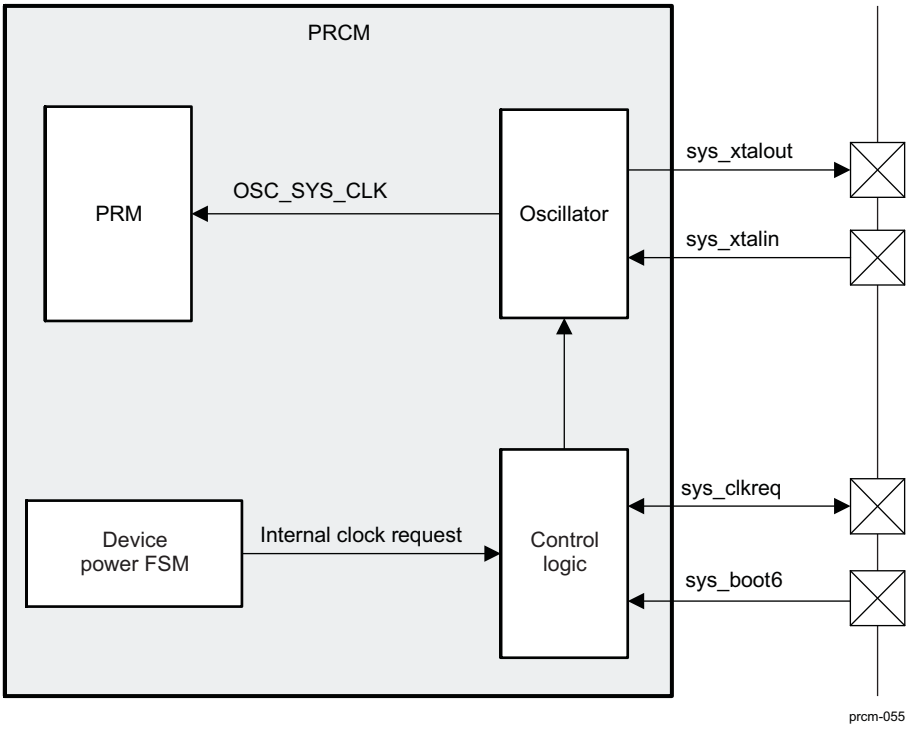


Table 4-29 lists the oscillator controls.

Table 4-29. Oscillator Controls

Oscillator Mode	Internal Clock Request[1]	External Clock Request[1]	Oscillator State	sys_clkreq Pad Direction	sys_clkreq[1]	Description
Master	Not asserted	Not asserted	Off	Both input and output	Not asserted	System clock not requested internally (within the device) and externally (by an external device or peripheral).
	Not asserted	Asserted	Active	Input	Asserted	System clock is requested externally only.
	Asserted	Not asserted	Active	Output	Asserted	System clock is requested internally only.
	Asserted	Asserted	Active	Both input and output (that is, driven internally by device and externally by peripheral)	Asserted	System clock is requested internally and externally.
Bypass	Not asserted	x[2]	Bypass	Both input and output (when external request is not asserted) or input (when external request is asserted)	External clock request state	System clock is not requested internally (sys_clkreq input has no effect).
	Asserted	x[2]	Bypass	Output (when only internal request is asserted) or both input and output (when external and internal request are asserted)	Asserted	System clock is requested internally (sys_clkreq input has no effect).

(1)
- If the PRCM.PRM_POLCTRL[1] CLKREQ_POL is set to active high (that is, 0x1), Asserted = 1, and Not asserted = 0.
- If the PRCM.PRM_POLCTRL[1] CLKREQ_POL is set to active low (that is, 0x0), Asserted = 0, and Not asserted = 1.

(2) x indicates that the signal may be asserted or not asserted.

4.7.5.3 External Output Clock1 (sys_clkout1) Control

The sys_clkout1 clock is active if the oscillator clock (OSC_SYS_CLK) is active (stable) and an external system clock request is active. It can be gated by programming the PRCM.PRM_CLKOUT_CTRL[7] CLKOUT_EN bit. The polarity of the sys_clkout1 signal, when the clock is gated, is controllable by programming the PRCM.PRM_POLCTRL[2] CLKOUT_POL bit.

When the device is in standby mode, both SYS_CLK and sys_clkout1 are disabled. In that case, reactivation of sys_clkout1 depends on the oscillator mode:

- Oscillator in active mode (sys_boot6 is 0): The sys_clkout1 clock can be reactivated (after oscillator stabilization), provided its gating was previously enabled by programming the PRCM.PRM_CLKOUT_CTRL[7] CLKOUT_EN bit and asserting an external clock request. This activation does not generate a device wake-up event; an external clock request activates only the internal SYS_CLK oscillator and sys_clkout1.
- Oscillator in bypass mode (sys_boot6 is 1): The sys_clkout1 clock can be reactivated only after the device wakes up (on any wake-up event) and SYS_CLK is active. When the device is active, SYS_CLK is running and sys_clkout1 is enabled as soon as requested by software.

4.7.5.4 External Output Clock2 (sys_clkout2) Control

A second output clock, sys_clkout2, is generated with a frequency that can be the source-clock frequency divided by 1, 2, 4, 8, or 16. Its source clock can be CORE_CLK, CM_SYS_CLK, 96 MHz, or 54 MHz. The selected source clock must be enabled by software. Enabling sys_clkout2 does not automatically request the required source clock. The polarity of the sys_clkout2 signal, when the clock is gated, is controllable by programming the PRCM.CM_POLCTRL[0] CLKOUT2_POL bit.

4.7.6 DPLL Control

The PRCM allows the configuration of the output clock frequencies of the DPLLs by setting their respective multipliers and dividers. It also allows control of the operating mode of the DPLLs and automatic recalibration mode.

4.7.6.1 DPLL Multiplier and Divider Factors

DPLL clock outputs are set by programming the corresponding multiplier and divider factors M, N, M2, M3, M4, M5, and M6. Table 4-30 lists the register bit fields for configuration of the multiplier and divider factors for the DPLLs.

Table 4-30. DPLL Multiplier and Divider Factors

	DPLL1	DPLL3	DPLL4	DPLL5
M	PRCM.CM_CLKSEL1_ PLL_MPU[18:8] MPU_ DPLL_MULT	PRCM.CM_CLKSEL1_ PLL[26:16] CORE_DPLL_MULT	PRCM.CM_CLKSEL2_ PLL[18:8] PERIPH_ DPLL_MULT	PRCM.CM_CLKSEL4_ PLL[18:8] PERIPH2_ DPLL_MULT
N	PRCM.CM_CLKSEL1_ PLL_MPU[6:0] MPU_ DPLL_DIV	PRCM.CM_CLKSEL1_ PLL[14:8] CORE_DPLL_DIV	PRCM.CM_CLKSEL2_ PLL[6:0] PERIPH_ DPLL_DIV	PRCM.CM_CLKSEL4_ PLL[6:0] PERIPH2_ DPLL_DIV
M2	PRCM.CM_CLKSEL2_ PLL_MPU[4:0] MPU_ DPLL_CLKOUT_DIV	PRCM.CM_CLKSEL1_ PLL[31:27] CORE_DPLL_ CLKOUT_DIV	PRCM.CM_CLKSEL3_ PLL[4:0] DIV_96M	PRCM.CM_CLKSEL5_ PLL[4:0] DIV_120M
M3	Not used	PRCM.CM_CLKSEL1_ EMU[20:16] DIV_DPLL3	PRCM.CM_CLKSEL_ DSS[12:8] CLKSEL_TV	Not used
M4	Not used	Not used	PRCM.CM_CLKSEL_ DSS[4:0] CLKSEL_DSS1	Not used
M5	Not used	Not used	PRCM.CM_CLKSEL_	Not used
M6	Not used	Not used	PRCM.CM_CLKSEL1_ EMU[28:24] DIV_DPLL4	Not used

4.7.6.2 DPLL Jitter Correction

To satisfy the jitter specification of the DPLL at a specific internal clock frequency, set the corresponding CM_CLKEN_PLL_processor_name>[7:4]processor_name>_DPLL_FREQSEL, CM_CLKEN_PLL[23:20] PERIPH_DPLL_FREQSEL, CM_CLKEN_PLL[7:4] CORE_DPLL_FREQSEL, CM_CLKEN2_PLL[7:4] PERIPH2_DPLL_FREQSEL bit field. Table 4-31 lists the possible values for the bit field and the corresponding internal clock frequency ranges.

NOTE: The internal clock frequency is the frequency of the internal interface clock Fint, with Fint= (CLKINP/(N + 1).

Table 4-31. Internal Clock Frequency

Bit Field Values	Clock Frequency Ranges
0x3	0.75 MHz—1.0 MHz
0x4	1.0 MHz—1.25 MHz
0x5	1.25 MHz—1.5 MHz
0x6	1.5 MHz—1.75 MHz
0x7	1.75 MHz—2.1 MHz
0xB	7.5 MHz—10 MHz
0xC	10 MHz—12.5 MHz
0xD	12.5 MHz—15 MHz
0xE	15 MHz—17.5 MHz
0xF	17.5 MHz—21 MHz
Other cases are reserved.	

NOTE: In the DPLL programming sequence, the DPLL_FREQSEL must be programmed before the new Multiplier factor M and the Divider factor N are programmed so that the new value is taken into account during current DPLL relock.

4.7.6.3 DPLL Frequency Ramp-Up Delay

When the DPLL switches from bypass mode to lock mode, the clock output frequency changes from bypass clock frequency to normal operating frequency. The frequency ramp-up feature allows the DPLL output frequency to switch gradually (in steps) from the bypass to locked frequency. Before reaching the locked frequency, the DPLL output switches to four intermediate frequencies:

1. Fout/8
2. Fout/4
3. Fout/2
4. Fout

The time delay in passing from the bypass clock frequency to normal frequency is the ramp-up delay. The ramp-up delay is configured by setting the CM_CLKEN_PLL_processor_name>[9:8]processor_name>_DPLL_RAMPTIME, CM_CLKEN_PLL[25:24] PERIPH_DPLL_RAMPTIME, CM_CLKEN_PLL[9:8] CORE_DPLL_RAMPTIME, CM_CLKEN2_PLL[9:8] PERIPH2_DPLL_RAMPTIME bit field.

There are three possible values for the bit field and the corresponding ramp-up delays:

- 0x0: Disables the frequency ramping feature.
- 0x1: Ramp step size range is 2–40 Fint cycles.
- 0x2: Ramp step size range is 4–80 Fint cycles.
- 0x3: Ramp step size range is 12–240 Fint cycles.

NOTE: If the ramp-up time configured for the DPLL is less than the DPLL lock time, the last frequency step, Fout/2 to Fout, gets stretched to the DPLL lock time.

4.7.6.4 DPLL Modes

DPLL supports several power modes (see Table 4-32). Each mode results in a tradeoff between power savings and relock time.

Table 4-32. DPLL Power Modes

Mode	Clock Input	Clock Output	DPLL Power State	Power Consumption	Latency
Locked	On	Lock frequency	ON	Maximum	N/A
Low-power bypass	On	Bypass frequency	ON	Less than locked	Same as low-power stop
Fast-relock bypass	On	Bypass frequency	ON	Less than locked	Less than low-power bypass
Low-power stop	On	Bypass frequency	ON	Less than locked	Same as low-power bypass
MN bypass	On	Bypass frequency	ON	Less than locked	Maximum
Off	Off	Off	OFF	Minimum	Maximum

A DPLL power mode can be achieved on a software request (manual) and/or automatically (automatic), depending on the specific hardware conditions. After a device power-on reset, the DPLL can be kept in either low-power stop mode (DPLL4 and DPLL5) or MN bypass mode (DPLL1 and DPLL3).

A DPLL can switch from one mode to the other as a result of the following:

- Software-programmed transition only (manual): The software configures a dedicated register for the

next desired DPLL mode. It must ensure that the transition can be performed based on the activity on the device.

- Combined software-programmed and hardware-conditions-based transition (automatic): The PRCM triggers the transition when the software requests it (by configuring the registers) and the hardware conditions are satisfied. When the hardware conditions are no longer met, the PRCM triggers the return transition.

For automatic transition, automatic mode must be enabled by programming the PRCM.CM_AUTOIDLE_PLL or the PRCM.CM_AUTOIDLE_PLLprocessor_name> registers.

Table 4-33 describes the manual and automatic control of the DPLL power modes by the PRCM.

Table 4-33. DPLL Power Modes Support

Mode	DPLL1	DPLL3	DPLL4	DPLL5
Locked	Software request (manual) or MPU wakes up (automatic).	Software request (manual) or CORE wakes up (automatic).	Software request (manual) or at least one peripheral clock is used (automatic).	Software request (manual) or at least one peripheral clock is used (automatic).
Low-power bypass	Software request (manual)	Software request (manual) or all interface clocks are gated (automatic).	N/A	N/A
Fast-relock bypass	N/A	Software request (manual).	N/A	N/A
Low-power stop	MPU is idle (automatic).	Device is idle (automatic).	(Default state) Software request (manual) or all functional clocks from DPLL are unused or on global reset release (automatic).	(Default state) Software request (manual) or all functional clocks from DPLL (120-MHz clock) are unused or on global reset release (automatic).
MN bypass	Global reset (automatic)	Global reset (automatic).	N/A	N/A
Off	Device off (automatic)	Device off (automatic).	Device off (automatic)	Device off (automatic)

NOTE: DPLL1 and DPLL3 cannot be manually forced to switch to Low-Power Stop mode from any other power mode. They must be in Locked state with automatic transition to Low-Power Stop mode configured and the hardware condition for the transition (identified in Table 4-33) must be satisfied, in order to switch to the Low-Power Stop mode.

Table 4-34 lists the bit fields that must be programmed for manual and automatic mode control of the four DPLLs.

Table 4-34. DPLL Power Mode Control

Mode	Manual Control	Auto Control
DPLL1	PRCM.CM_CLKEN_PLL_MPU[2:0] EN_MPU_DPLL	PRCM.CM_AUTOIDLE_PLL_MPU[2:0] AUTO_MPU_DPLL
DPLL3	PRCM.CM_CLKEN_PLL[2:0] EN_CORE_DPLL	PRCM.CM_AUTOIDLE_PLL[2:0] AUTO_CORE_DPLL
DPLL4	PRCM.CM_CLKEN_PLL[18:16] EN_PERIPH_DPLL	PRCM.CM_AUTOIDLE_PLL[5:3] AUTO_PERIPH_DPLL
DPLL5	PRCM.CM_CLKEN2_PLL[2:0] EN_PERIPH2_DPLL	PRCM.CM_AUTOIDLE2_PLL[2:0] AUTO_PERIPH2_DPLL

NOTE: The DPLL automatically enters locked mode on a domain wakeup only if the DPLL is locked before the sleep transition and one of the automatic modes is enabled.

4.7.6.5 DPLL Low-Power Mode

The DPLL can operate in a low-power mode by reducing the operating frequency range. This in turn reduces the power consumption of the DPLL. In this mode, however, there is a period and phase jitter effect.

The DPLL can enter this mode only if the targeted lock frequency of the DPLL is less than 600 MHz. This implies locking or relocking the DPLL to a new targeted locked-frequency when entering or exiting low-power mode. Software must ensure that the DPLL lock-frequency does not exceed 600 MHz in low-power mode.

Software can enable/disable automatic switching of the DPLL between normal mode and low-power mode. The new mode is effective only after the DPLL is relocked. Low-power mode control is considered only during the following transitions:

- From bypass mode to lock
- From stop mode to lock
- From lock to relock

Table 4-35 lists the bit fields that must be programmed for manual control of the four DPLLs.

Table 4-35. LP Mode Control

Mode	Manual Control
DPLL1	PRCM.CM_CLKEN_PLL_MPU[10] EN_MPU_DPLL_LPMODE
DPLL3	PRCM.CM_CLKEN_PLL[10] EN_CORE_DPLL_LPMODE
DPLL4	PRCM.CM_CLKEN_PLL[26] EN_PERIPH_DPLL_LPMODE
DPLL5	PRCM.CM_CLKEN2_PLL[10] EN_PERIPH2_DPLL_LPMODE

4.7.6.6 DPLL Clock Path Power Down

DPLL3 and DPLL4 can power down the CLKOUTX2 path. A small section of logic is powered down as the M2 post divider is also shared with the CLKOUT path, which remains functional.

The HSDIVIDER can power down each CLKOUTn path (with n in the range of 3 to 6) independently, therefore allowing further power savings. The clock output path is also powered down when the DPLL is in stop mode, regardless of the software setting.

Software must ensure the proper sequencing of the control. To avoid a glitch at the output, activate this control when the clock is no longer required, and when the output clock is gated. Conversely, ensure a delay between deactivation and reactivation of the clock by using the power-down control.

Table 4-36 lists the bit fields and the corresponding clock outputs of the DPLLs.

Table 4-36. Clock Path Power-Down Control

DPLL	Control Bit Field	Clock Path
DPLL4	PRCM.CM_CLKEN_PLL[27] PWRDN_96M	96-MHz clock output (DPLL4 output M2X2)
	PRCM.CM_CLKEN_PLL[28] PWRDN_TV	DSS TV clock output (DPLL4 output M3X2)
	PRCM.CM_CLKEN_PLL[29] PWRDN_DSS1	DSS1 clock output (DPLL4 output M4X2)
	PRCM.CM_CLKEN_PLL[31] PWRDN_EMU_PERIPH	EMU_PERIPH clock output (DPLL4 output M6X2)
DPLL3	PRCM.CM_CLKEN_PLL[12] PWRDN_EMU_CORE	EMU_CORE clock output (DPLL3 output M3X2)

4.7.6.7 Latencies

Lock mode for the DPLL is frequency lock.

Lock latencies depend on the internal reference frequency (F_{INT}) of the DPLL, calculated as:

$F_{INT} = F_{ref} / (N+1)$

where F_{ref} is the reference clock input to DPLL.

Table 4-37 lists the operating modes of the DPLL and the output clock frequency and associated lock latency.

FREQSEL[3] = 0 refers to the following bits:

- CM_CLKEN_PLL_MPU[7]
- CM_CLKEN_PLL[7]
- CM_CLKEN_PLL[23]
- CM_CLKEN2_PLL[7]

Table 4-37. DPLL Operating Mode and Latency

Mode	CLKOUTX2	Lock Time in FINT cycles (FREQSEL[3] = 0)	Lock Time in Fint Cycles (FREQSEL[3]= 1)
OFF	Off	150	780
M, N reprogramming	bypass clock	150	780
Low-power stop mode	Off	40	400
Low-power bypass	bypass clock	40	400
Fast-relock bypass	bypass clock	10	100
Active (locked)	CLKINP x M/(N+1)/M2 x 2	N/A	N/A

The post-divider M2 supports change on the fly. The time required to switch from the clock with the old period to the clock with the new period depends on the old value of M2 (see Table 4-38).

Table 4-38. Time Required to Switch Clocks

Old Value of M2	Time for Switching
1	4 REFCLKs + 8 CLKOUTX2s
Even	4 REFCLKs + 2 CLKOUTX2s
Odd & > 1	4 REFCLKs + 4 CLKOUTX2s

4.7.6.8 Recalibration

A lock sequence occurs during an initial lock or during a relock following a new multiplier or divider value. Each time the DPLL is reset or performs a lock sequence, it performs a recalibration of the output frequency, based on the voltage and temperature conditions. By compensating for voltage and temperature changes within a certain range, the calibration allows the lock frequency to remain steady. If the voltage or temperature drifts outside the acceptable range, the DPLL asserts a recalibration flag.

For example, a large temperature drift can cause the DPLL to lose its lock and require recalibration. When the DPLL locks at a temperature within the 080 degrees Celsius range, the maximum temperature drift is approximately 55 degrees Celsius. When DPLL starts at a negative temperature, the maximum temperature drift is higher.

If the DPLL locks at 30 degrees Celsius, the temperature can change by 60 degrees Celsius (from 30 to +90 degrees Celsius) and the DPLL will not lose the lock. However, for temperatures above the 60 degrees Celsius range, the DPLL may need to be relocked. A new relock sequence reinitializes the starting temperature.

This compensation mechanism is active only while the DPLL is locked. When the DPLL is in off or bypass mode (low-power or fast-relock), it does not assert the recalibration flag. If the voltage or temperature exceeds the drift limits while the DPLL is not locked, and then the DPLL tries to relock, the DPLL fails to lock within the normal delay and recalibrates automatically before eventually locking. The only difference from a standard relock is the delay.

The DPLL can automatically start recalibration when the recalibration flag is asserted, or recalibration can be managed by the software. The mode of operation is selected by configuring the corresponding registers in the PRCM (see Table 4-39). The software or manual control mode is selected by default.

NOTE: Automatic recalibration of the DPLL can start at any time. While relocking, the DPLL switches to bypass mode. For modules that are sensitive to frequency change while operating, this can introduce operational instability. For example, the SDRC is sensitive to a frequency change on DPLL3 because its embedded DLL relocks on a frequency change. Any access during this DLL relock period can be corrupted. It is important, therefore, to stall SDRC access during DPLL recalibration.

To allow the software to recalibrate the DPLL at the correct time depending on the device activity, the PRCM can generate a wake-up event on the MPU domain, followed by an interrupt on the MPU when the DPLL recalibration flag is asserted.

Table 4-39 summarizes the software programming control over the DPLL recalibration feature.

Table 4-39. DPLL Recalibration Controls

DPLL	Software Control	Description
DPLL1 (MPU)	PRCM.CM_CLKEN_PLL_MPU[3] MPU_DPLL_DRIFTGUARD	Enable/disable the MPU DPLL automatic recalibration feature.
	PRCM.PRM_IRQENABLE_MPU[7] MPU_DPLL_RECAL_EN	Enable/disable the MPU DPLL recalibration interrupt to MPU.
	PRCM.PRM_IRQSTATUS_MPU[7] MPU_DPLL_ST	Status of the MPU DPLL recalibration interrupt
DPLL3 (CORE)	PRCM.CM_CLKEN_PLL[3] EN_CORE_DPLL_DRIFTGUARD	Enable/disable the CORE DPLL automatic recalibration feature.
	PRCM.PRM_IRQENABLE_MPU[5] CORE_DPLL_RECAL	Enable/disable the CORE DPLL recalibration interrupt to MPU.
	PRCM.PRM_IRQSTATUS_MPU[5] CORE_DPLL_ST	Status of the CORE DPLL recalibration interrupt
DPLL4 (PER)	PRCM.CM_CLKEN_PLL[19] EN_PERIPH_DPLL_DRIFTGUARD	Enable/disable the PER DPLL automatic recalibration feature.
	PRCM.PRM_IRQENABLE_MPU[6] PERIPH_DPLL_RECAL	Enable/disable the PER DPLL recalibration interrupt to MPU.
	PRCM.PRM_IRQSTATUS_MPU[6] PERIPH_DPLL_ST	Status of the PER DPLL recalibration interrupt
DPLL5 (PER2)	PRCM.CM_CLKEN2_PLL[3] EN_PERIPH2_DPLL_DRIFTGUARD	Enable/disable the PER DPLL2 automatic recalibration feature.
	PRCM.PRM_IRQENABLE_MPU[25] SND_PERIPH_DPLL_RECAL_EN	Enable/disable the PER DPLL2 recalibration interrupt to MPU.
	PRCM.PRM_IRQSTATUS_MPU[25] SND_PERIPH_DPLL_ST	Status of the PER DPLL2 recalibration interrupt

NOTE: DPLL recalibration is not necessary in real use (specified operating voltage and temperature range).

4.7.6.9 DPLL Programming Sequence

The DPLL programming sequence follows:

1. Set the multiplier (M) and divider (N) values for the desired CLKOUT frequency (see Section 4.7.6.1).
2. Set the corresponding output dividers (M2, M3, M4, M5, and M6) (see Section 4.7.6.1).
3. Set the corresponding FREQSEL bit field to satisfy the jitter specification (see Section 4.7.6.2).

4. Set the corresponding ramp-up delay (see Section 4.7.6.3).
5. Enable/disable the auto-recalibration feature (see Section 4.7.6.8).
6. Enable/disable the autoidle feature (see Section 4.7.6.4).
7. Mask/unmask the interrupt to the MPU (see Section 4.7.6.8).
8. Enable the DPLL lock mode (see Section 4.7.6.4).

4.7.7 Internal Clock Controls

This section describes the software and hardware controls of the internal source clocks. Figure 4-43 through list the clock controls. The *Source selection/division* column lists the PRCM register bits used to select or divide the clocks. The *Software control* column lists the PRCM register bits used to enable/disable the clocks. The *Hardware control* column lists the hardware conditions required to effectively gate the clocks.

In the *Hardware control* column, the boxes labeled CL, GS, GC, and HC indicate specific information about the hardware clock controls:

- CL (combinational logic): The functional or interface clock is required by more than one module across more than one domain. The gating control is the OR combination of all the domain clock requests. If any module of this clock domain requests the clock, the clock is not gated.
- GS (gating selection): The clock is selectable among several possible source clocks for a module. The gating control depends on the software programming of the CM_CLKSEL_<domain_name> type of register. The clock request of the module or domain must be set by the CM_CLKSEL bit.
- GC (gating control): The functional/interface clock is required by a single module across the domain. The gating control depends only on the software programming of the FCLKEN/ICLKEN bit. For the interface clock, the enable bit is effective only if autoidle mode is not used. If autoidle mode is used, the gating control also depends on the state of the domain.
- HC (hardware control): A specific rule not covered by CL, GS, or GC.

NOTE: Because the PRCM must receive hardware acknowledgement from the different modules before it can gate the clock, the clock is not gated immediately after the software requests clock-gating conditions.

4.7.7.1 PRM Source-Clock Controls

Figure 4-43 shows the common source-clock controls for the PRM.

Figure 4-43. Common PRM Source-Clock Controls

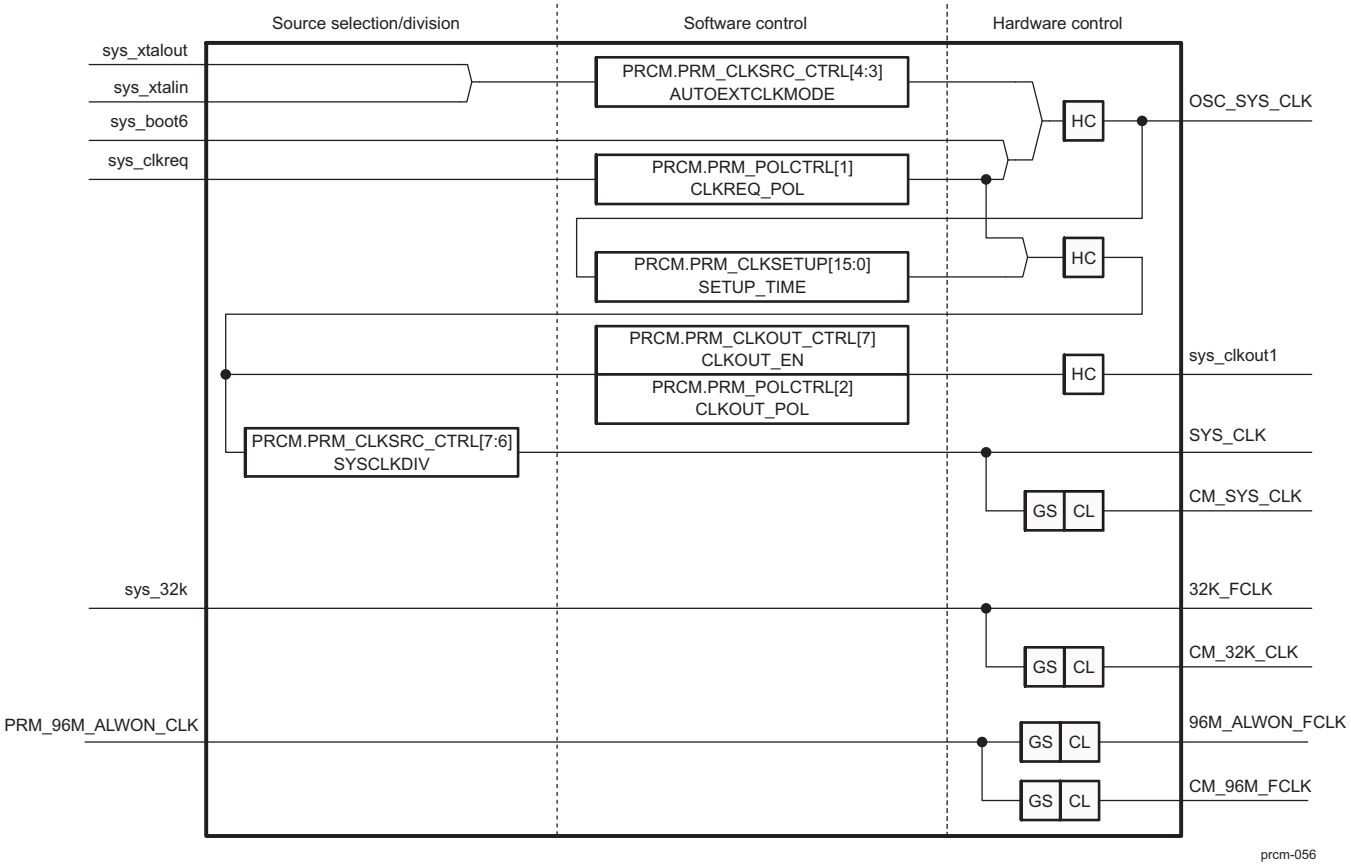


Table 4-40 shows the common source-clock gating controls for the PRM.

Table 4-40. Common PRM Source-Clock Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
OSC_SYS_CLK	Running	PRCM.PRM_CLKSRC_CTRL[4:3] AUTOEXTCLKMODE and device power state and sys_clkreq	Gated when the oscillator is programmed to power down with the device sleep/retention/off transition.
sys_clkout1	Running	PRCM.PRM_CLKOUT_CTRL[7] CLKOUT_EN and PRCM.PRM_POLCTRL[2] CLKOUT_POL and sys_clkreq	Active when OSC_SYS_CLK is active, sys_clkout1 is enabled, and sys_clkreq is asserted.
SYS_CLK	Running	Activated after clksetup_count_overflow	Active when OSC_SYS_CLK is active and the SYS_CLK setup time is up.
CM_SYS_CLK	Running	PRCM.CM_CLKSEL_CORE[6] CLKSEL_GPT10, PRCM.CM_CLKSEL_CORE[7] CLKSEL_GPT11, and depends on the clock-gating conditions of GPT10_FCLK and GPT11_FCLK	Active if it is the source clock of the GPT10_FCLK or GPT11_FCLK and the functional clock is active.
32K_FCLK	Running	None	Always-active clock from sys_32k input pin
CM_32K_CLK	Running	PRCM.CM_CLKSEL_CORE[6] CLKSEL_GPT10, PRCM.CM_CLKSEL_CORE[7] CLKSEL_GPT11, and depends on the clock-gating conditions of GPT10_FCLK and GPT11_FCLK	Active if it is the source clock of the GPT10_FCLK or GPT11_FCLK and the functional clock is active.
CM_96M_FCLK	Gated	CM_CLKSEL1_PLL[3] SOURCE_48M bit cleared, and depends on the clock-gating condition of 96M_FCLK, 48M_FCLK, and 12M_FCLK	Active if the derived clocks (96M_FCLK, 48M_FCLK, and 12M_FCLK) are active.
96M_ALWON_FCLK	Gated	CM_FCLKEN_PER[0] EN_MCBSP2, CM_FCLKEN_PER[1] EN_MCBSP3, and CM_FCLKEN_PER[2] EN_MCBSP4	Gated when none of the three McBSPs [2..4] have their functional clock enable requested.

The oscillator output clock (OSC_SYS_CLK) is gated whenever the PRCM.PRM_CLKSRC_CTRL[4:3] AUTOEXTCLKMODE bit field is programmed to power down the oscillator when the device enters retention. In this condition, all the clock trees in the device must be gated, and the four DPLLs (DPLL1, DPLL3, and DPLL4) must enter stop mode before this transition can occur.

SYS_CLK is gated under the same conditions as the oscillator output clock, but it is enabled only after the oscillator stabilizes. Oscillator stabilization is determined by a counter overflow configured in the PRCM.PRM_CLKSETUP[15:0] SETUP_TIME bit field.

The sys_clkreq active condition is described in Section 4.7.5, *External Clock Control*.

4.7.7.2 CM Source-Clock Controls

Figure 4-44 shows the common source-clock controls for the CM.

Figure 4-44. Common CM Source-Clock Controls

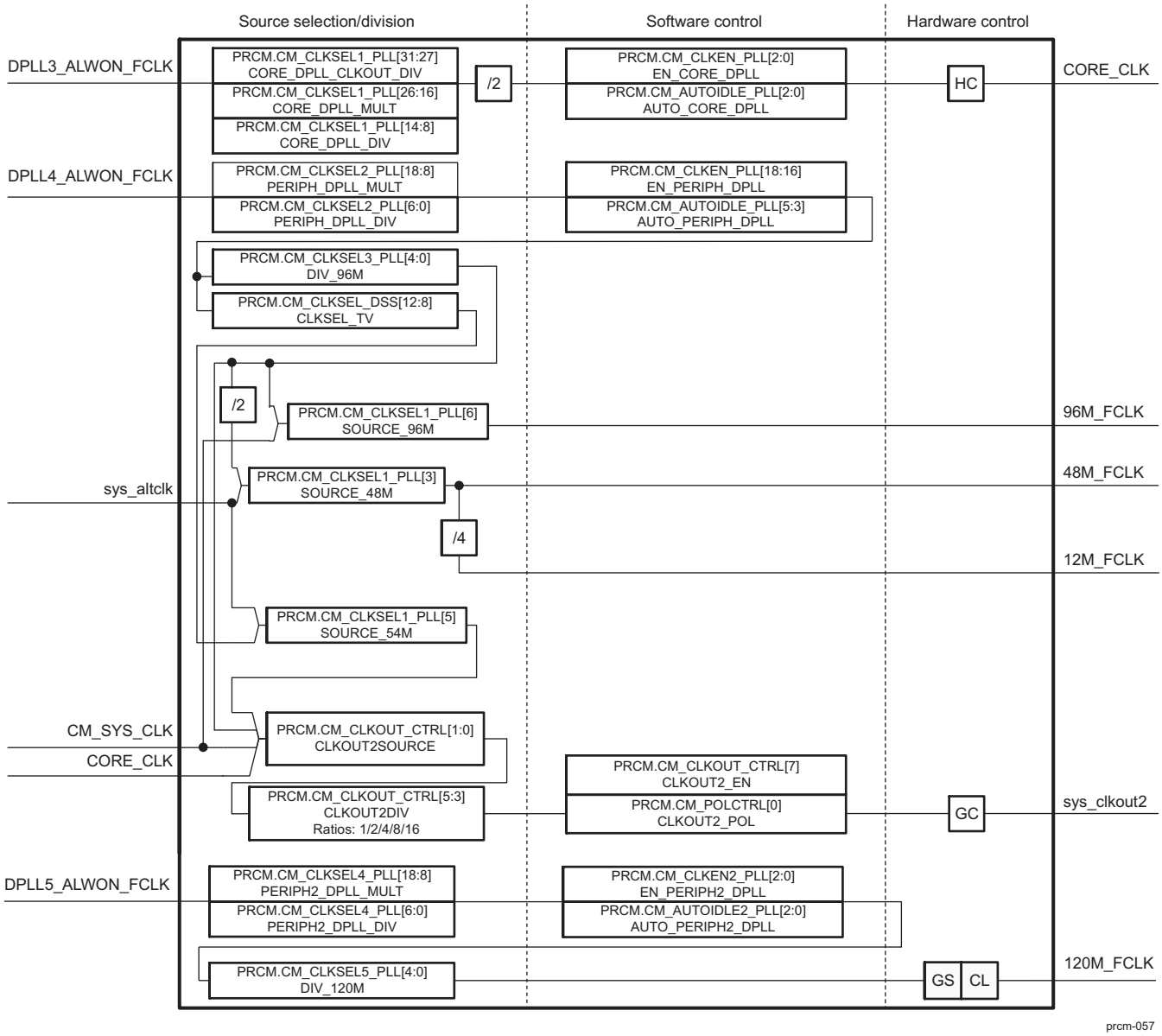


Table 4-41 shows the common source-clock gating controls for the CM.

Table 4-41. Common CM Source-Clock Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
CORE_CLK	Running	Depends on the clock-gating conditions of: L3_ICLK and L4_ICLK	Gated when all interface clocks of the different modules of the device are gated
96M_FCLK	Stopped	Depends on the clock-gating conditions of: CORE_96M_FCLK and DSS_96M_FCLK	If the dependent clocks are active, the clock is active.
48M_FCLK	Stopped	Depends on the clock-gating conditions of CORE_12M_FCLK, PER_48M_FCLK and USBHOST_48M_FCLK	If the dependent clocks are active, the clock is active.
12M_FCLK	Stopped	Depends on the clock-gating conditions of CORE_12M_FCLK	If the dependent clock is active, the clock is active.
sys_clkout2	Stopped	PRCM.CM_CLKOUT_CTRL[7] CLKOUT2_EN	Active if enabled
DPLL4_M2_CLK	Stopped	Depends on the clock-gating conditions of: 96M_FCLK	If the dependent clocks are active, the clock is active.
DPLL4_M3_CLK	Stopped	CM_CLKSEL1_PLL.SOURCE_54M and depends on the clock-gating conditions of: DPLL4_M2_CLK and DSS_TV_CLK	If the dependent clocks are active, the clock is active. DSS_TV_CLK is a dependent clock if set by the register configuration.
120M_FCLK	Stopped	CM_FCLKEN_USBHOST[1] EN_USBHOST2, CM_FCLKEN3_CORE[2] EN_USBTLL, CM_FCLKEN_WKUP[9] EN_USIMOCP, CM_CLKSEL_WKUP[6:3] CLKSEL_USIMOCP	If any of the dependent clocks (that is, USIM_FCLK, CORE_120M_FCLK, or USBHOST_12M_FCLK) is active, the clock is active. The USIM_FCLK is a dependent clock if set by the register configuration.

4.7.7.3 Common Interface Clock Controls

Figure 4-45 shows the clock controls for the common interface.

Figure 4-45. Common Interface Clock Controls

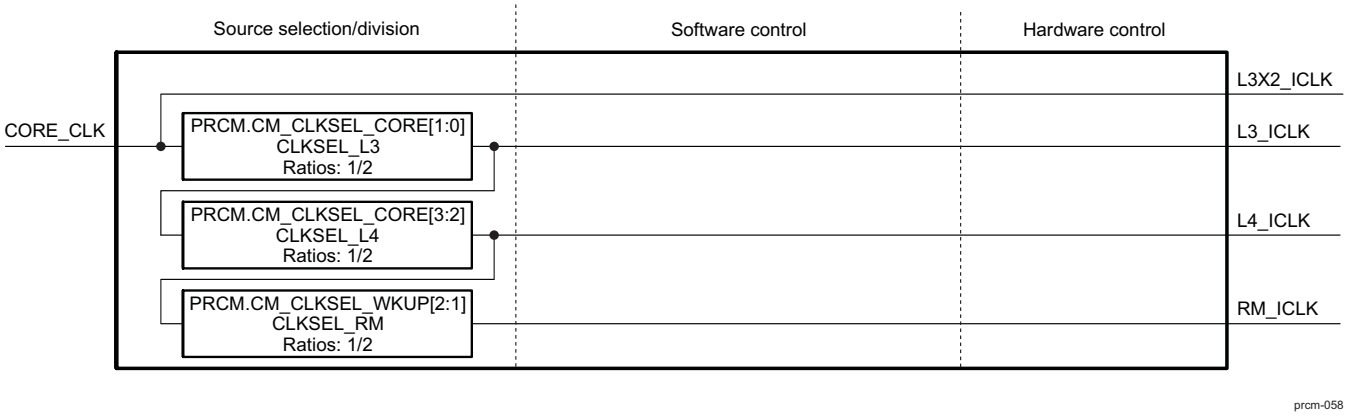


Table 4-42 shows the clock-gating controls for the common interface.

Table 4-42. Common Interface Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
L3X2_ICLK	Running	CORE_CLK gating conditions	Depends on the gating conditions of the CORE_CLK
L3_ICLK	Running	Depends on the clock-gating conditions of: SGX_L3_ICLK, CORE_L3_ICLK, SECURITY_L3_ICLK	Gated when all L3 interface clocks of the different modules of the device are gated
L4_ICLK	Running	Depends on the clock-gating conditions of CORE_L4_ICLK, SECURITY_L4_ICLK, DSS_L4_ICLK, PER_L4_ICLK, SR_L4_ICLK, and WKUP_L4_ICLK	Gated when all L4 interface clocks of the different modules of the device are gated
RM_ICLK	Running	None	Gated with source clock (CORE_CLK)

4.7.7.4 DPLL Source-Clock Controls

Figure 4-46 shows the clock controls for the DPLL domain.

Figure 4-46. DPLL Domain Clock Controls

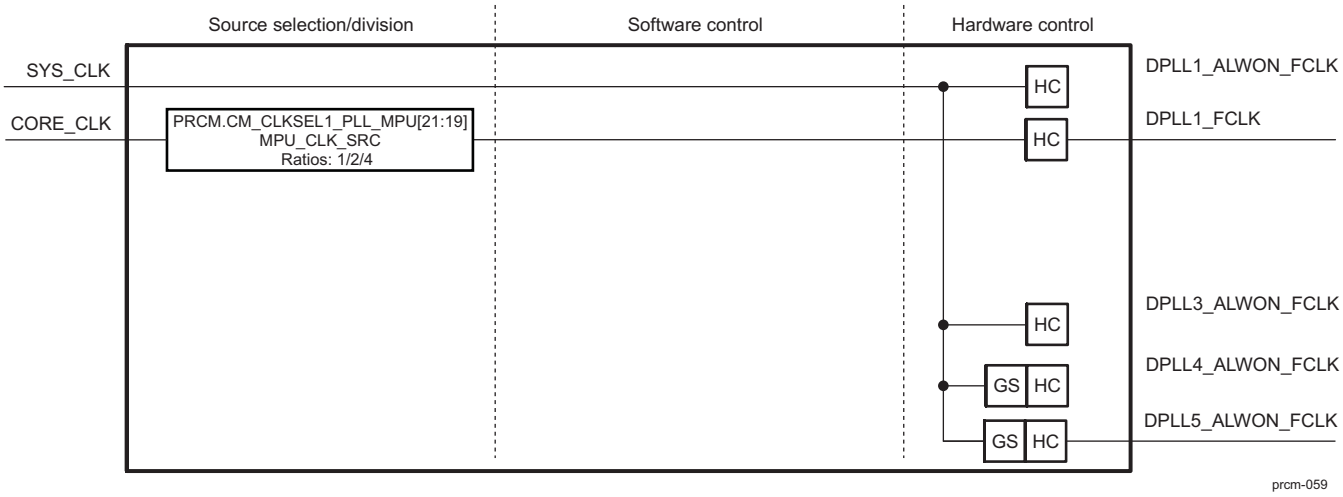


Table 4-43 shows the clock-gating controls for the DPLL domain.

Table 4-43. DPLL Domain Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
DPLL1_ALWON_FCLK	Running	PRCM.CM_AUTOIDLE_PLL_MPU[2:0] AUTO_MPU_DPLL, PRCM.CM_CLKEN_PLL_MPU[2:0] EN_MPU_DPLL, and MPU domain power state	Gated if the DPLL is set to automatic active control and enabled in lock mode while the MPU domain goes into retention or off mode. It is also gated if DPLL is set to low-power bypass mode.
DPLL1_FCLK	Stopped		
DPLL3_ALWON_FCLK	Running	PRCM.CM_AUTOIDLE_PLL[2:0] AUTO_CORE_DPLL, PRCM.CM_CLKEN_PLL[2:0] EN_CORE_DPLL, and CORE domain power clocks state	Gated if the DPLL is set to automatic active control and enabled in lock mode while the CORE domain is idle. It is also gated if DPLL is set to low-power or fast-relock bypass mode.
DPLL4_ALWON_FCLK	Stopped	PRCM.CM_AUTOIDLE_PLL[5:3] AUTO_PERIPH_DPLL, PRCM.CM_CLKEN_PLL[18:16] EN_PERIPH_DPLL, and depends on the clock-gating conditions of DPLL4_M2_CLK	Gated if the DPLL is set to automatic active control and enabled in lock mode while its dependent clock is inactive. It is also gated if DPLL is set to low-power stop mode.
DPLL5_ALWON_FCLK	Stopped	PRCM.CM_AUTOIDLE2_PLL[2:0] AUTO_PERIPH2_DPLL, PRCM.CM_CLKEN2_PLL[2:0] EN_PERIPH2_DPLL, and depends on the clock-gating conditions of DPLL5_M2_CLK	Gated if the DPLL is set to automatic active control and enabled in lock mode while its dependent clock is inactive. It is also gated if DPLL is set to low-power stop mode.

4.7.7.5 SGX Domain Clock Controls

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power savings considerations, ensure that clocks to unused modules are cut off.

Figure 4-47 shows the clock controls for the SGX domain.

Figure 4-47. SGX Domain Clock Controls

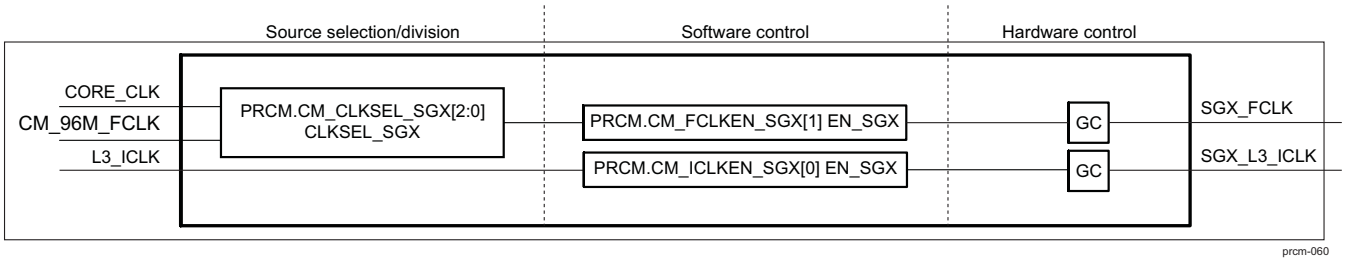


Table 4-44 lists the clock-gating controls for the SGX domain.

Table 4-44. SGX Domain Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
SGX_FCLK	Stopped	PRCM.CM_FCLKEN_SGX[1] EN_SGX	Gated when the enable bit is set to 0
SGX_ICLK	Stopped	PRCM.CM_ICLKEN_SGX[1] EN_SGX	Gated when the enable bit is set to 0

4.7.7.6 CORE Domain Clock Controls

Figure 4-48 through Figure 4-50 show the clock controls for the CORE domain.

Figure 4-48. CORE Domain Clock Controls: Part 1

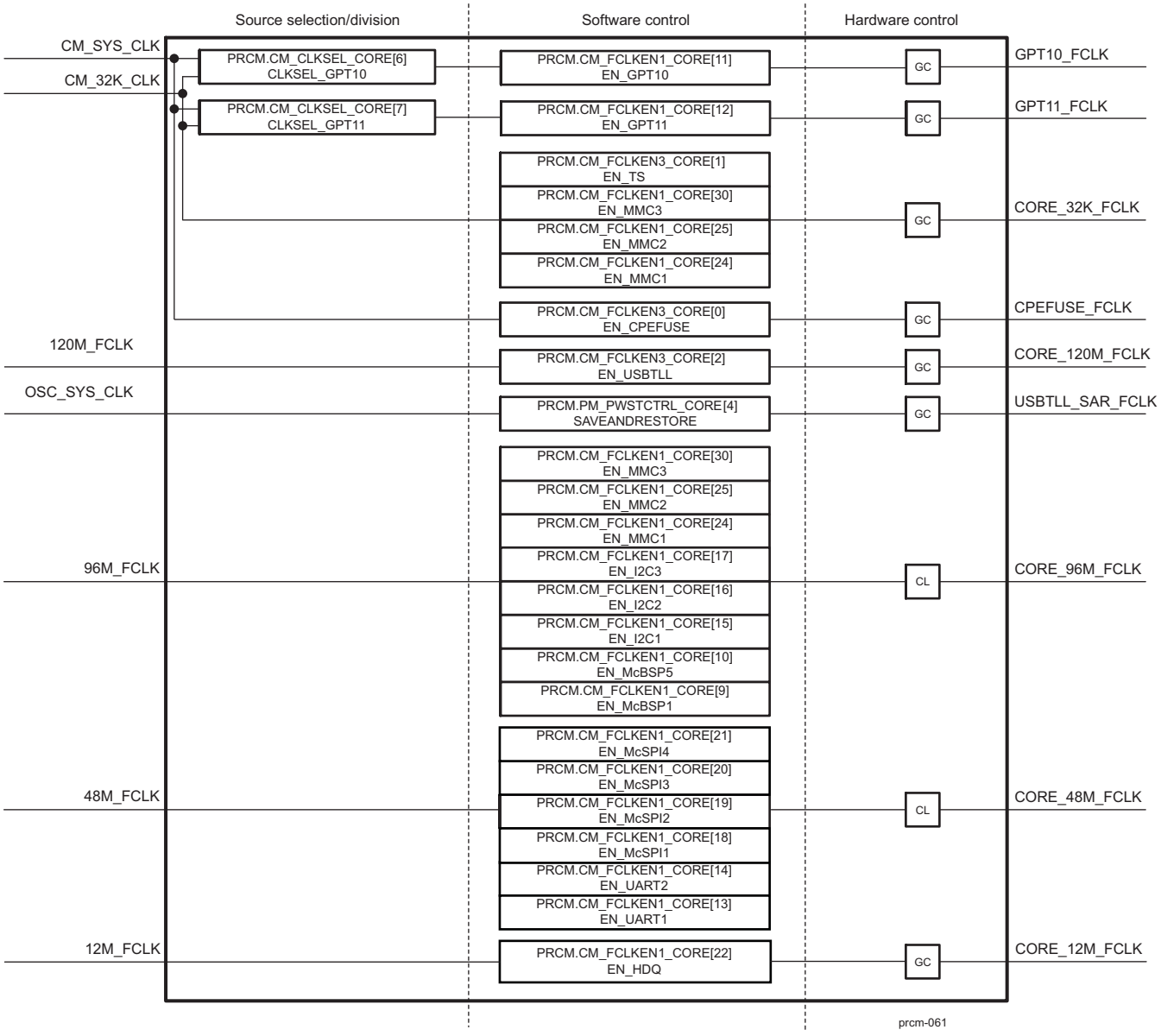


Figure 4-49. CORE Domain Clock Controls: Part 2

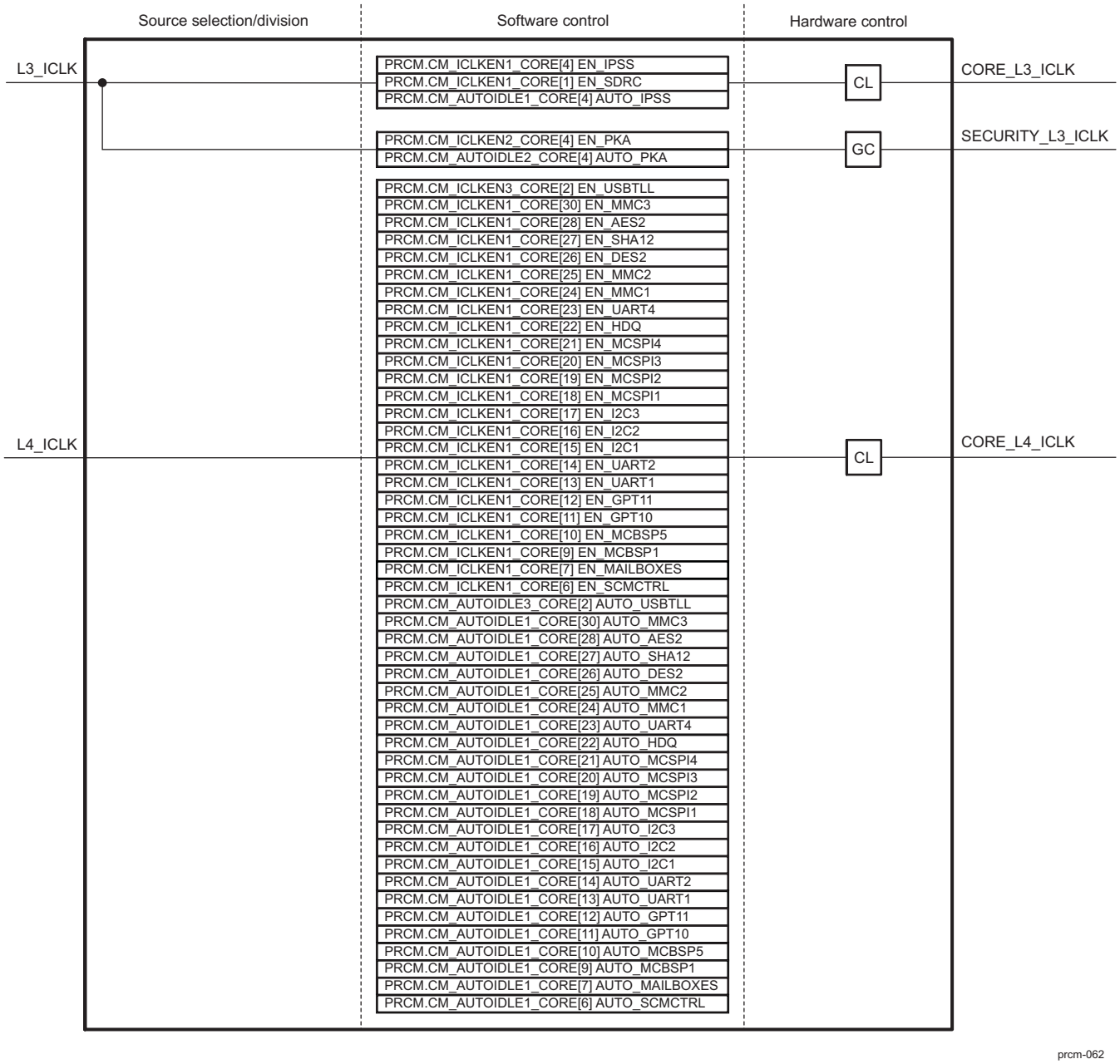


Figure 4-50. CORE Domain Clock Controls: Part 3

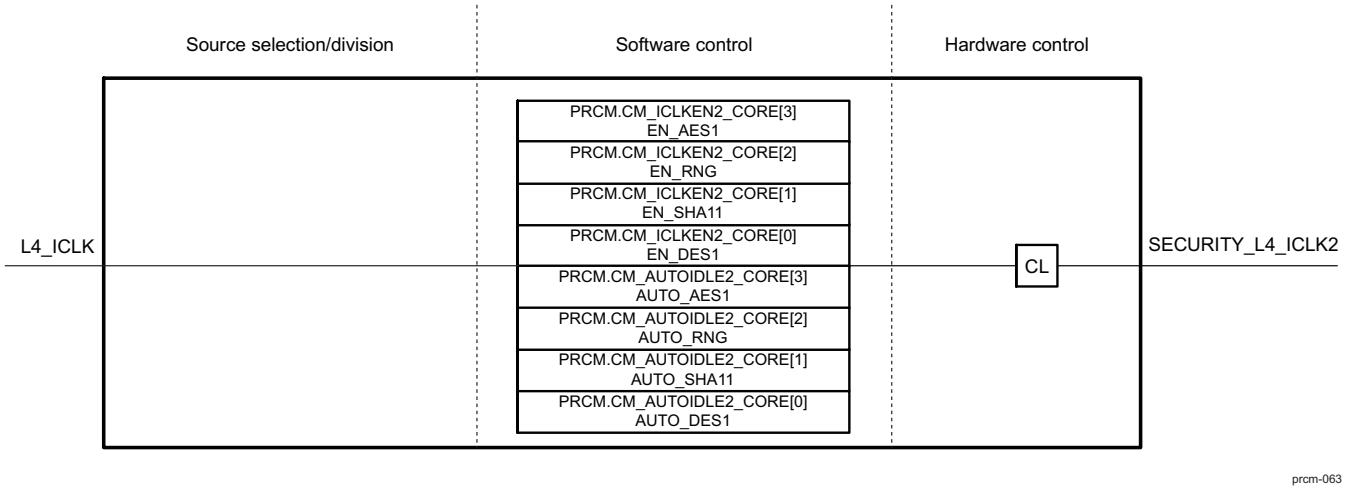


Table 4-45 lists the clock-gating controls for the CORE domain.

Table 4-45. CORE Domain Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
GPT10_FCLK	Stopped	PRCM.CM_FCLKEN1_CORE[11] EN_GPT10	Gated when the enable bit is set to 0
GPT11_FCLK	Stopped	PRCM.CM_FCLKEN1_CORE[12] EN_GPT11	Gated when the enable bit is set to 0
CORE_96M_FCLK	Stopped	McBSP[1..5] input clock source select (in SCM) and PRCM.CM_FCLKEN1_CORE (MMC[1-2], McBSP[1, 5], I2C[1-3])	Gated when the enable bits of the module functional clock are set to 0. (The McBSPs can have MCBSP_CLKS as an alternate functional clock.)
CORE_48M_FCLK	Stopped	PRCM.CM_FCLKEN1_CORE (UART[1-2], McSPI[1-4])	Gated when the functional clock enable bits of the module are set to 0
CORE_12M_FCLK	Stopped	PRCM.CM_FCLKEN1_CORE[22] EN_HDQ	Gated when the enable bit is set to 0
CORE_L3_ICLK	Running	PRCM.CM_ICLKEN1_CORE EN_(SDRC, IPSS), PRCM.CM_AUTOIDLE1_CORE4] AUTO_IPSS	Gated when: 1. All enable bits are set to 0. 2. The enable-autoidle bit pair is set to 1, the remaining enable bits are set to 0, and the clock is not requested by any module.
CORE_L4_ICLK	Running	PRCM.CM_ICLKEN1_CORE (AES2, SHAM2, D3D2, MMC[1..2], HDQ, MCSPI[1-4], I2C[1-3], UART[1,2], GPT[10,11], MCBSP[1,5], SCMCTRL) and PRCM.CM_AUTOIDLE1_CORE (AES2, SHAM2, D3D2, MMC[1..2], HDQ, MCSPI[1-4], I2C[1-3], UART[1,2], GPT[10,11], MCBSP[1,5], SCMCTRL)	Gated when: 1. All enable bits are set to 0. 2. All enable-autoidle bit pairs are set to 1, and the clock is not requested by any module.
CORE_32K_FCLK	Stopped	CM_FCLKEN1_CORE[24] EN_MMC1, CM_FCLKEN1_CORE[25] EN_MMC2, CM_FCLKEN1_CORE[30] EN_MMC3, CM_FCLKEN3_CORE[1] EN_TS	Gated when: 1. All enable bits are set to 0.
CPEFUSE_FCLK	Stopped	CM_FCLKEN3_CORE[0] EN_CPEFUSE	Gated when CPEFUSE autoload sequence is performed, and the enable bit is set to 0
CM_USIM_CLK	Stopped	CM_FCLKEN_WKUP[9] EN_USIMOCP	Gated when the enable bit is set to 0
USBTLL_SAR_FCLK	Stopped	CORE domain power state and PM_PWSTCTRL_CORE[4] SAVEANDRESTORE	Gated when the save-restore bit is set to 0 after the restore operation completes.
CORE_120M_FCLK	Stopped	CM_FCLKEN3_CORE[0] EN_USBTLL and DPLL5 operating mode	Gated when the enable bit is set to 0, or the DPLL5 is in stop or bypass mode

4.7.7.7 EFUSE Domain Clock Controls

Figure 4-51 shows the clock controls for the EFUSE domain. Table 4-46 lists the clock-gating control for the EFUSE domain.

Figure 4-51. EFUSE Domain Clock Controls

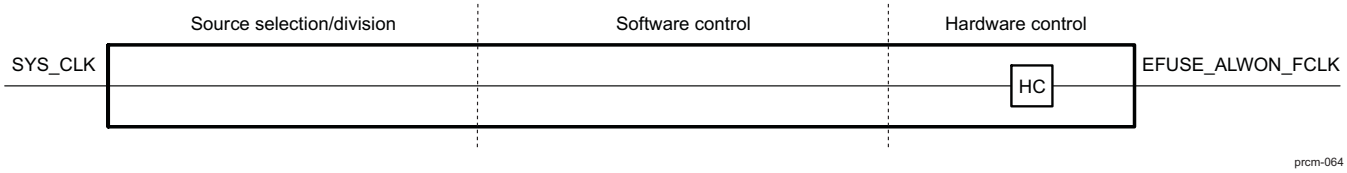


Table 4-46. EFUSE Domain Clock-Gating Control

Clock Name	Reset	Clock-Gating Control	Gating Description
EFUSE_ALWON_FCLK	Running	None	Active when VDD_CORE is ramped up to power on the device and the eFuse-ready hardware signal is released.

4.7.7.8 DSS Domain Clock Controls

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power savings considerations, ensure that clocks to unused modules are cut off.

Figure 4-52 shows the clock controls for the DSS domain. Table 4-47 lists the clock-gating controls for the DSS domain.

Figure 4-52. DSS Domain Clock Controls

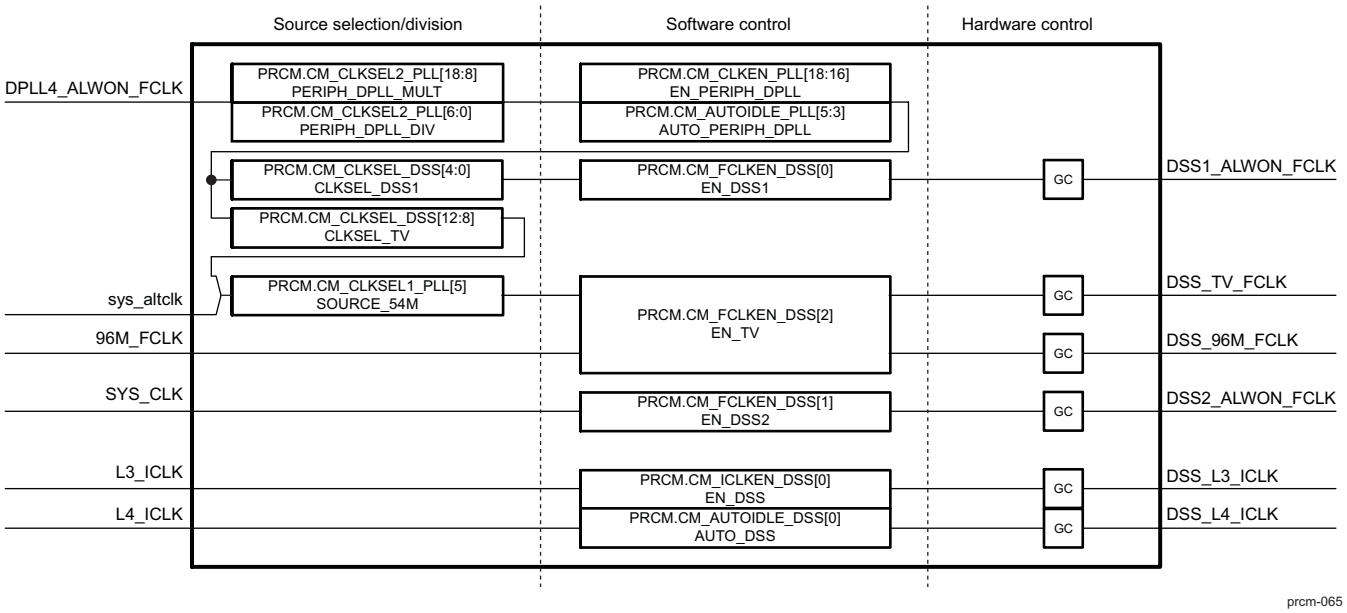


Table 4-47. DSS Domain Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
DSS1_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_DSS[0] EN_DSS1	Gated when the enable bit is set to 0
DSS2_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_DSS[1] EN_DSS2	Gated when the enable bit is set to 0
DSS_TV_FCLK	Stopped	PRCM.CM_FCLKEN_DSS[2] EN_TV	Gated when the enable bit is set to 0
DSS_96M_FCLK	Stopped		

Table 4-47. DSS Domain Clock-Gating Controls (continued)

Clock Name	Reset	Clock-Gating Control	Gating Description
DSS_L3_ICLK	Stopped	PRCM. CM_ICLKEN_DSS[0] EN_DSS, PRCM.CM_AUTOIDLE_DSS[0] AUTO_DSS	Gated when: 1. Enable bit is set to 0. 2. Enable-autoidle bit pair is set to 1, and the clock is not requested by any module.
DSS_L4_ICLK	Stopped		

4.7.7.9 USBHOST Domain Clock Controls

Figure 4-53 shows the clock controls for the USBHOST domain. Table 4-48 lists the clock-gating controls for the USBHOST domain.

Figure 4-53. USBHOST Domain Clock Controls

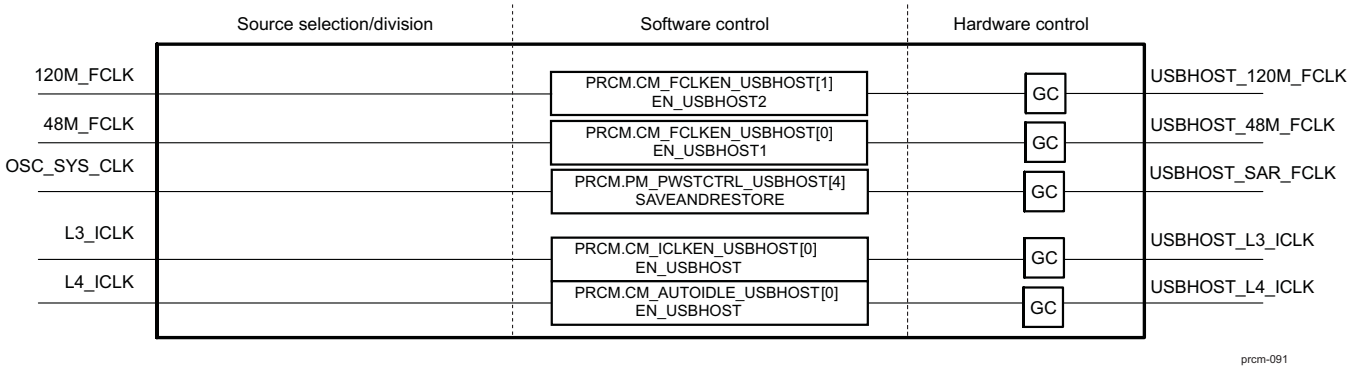


Table 4-48. USBHOST Domain Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
USBHOST_48M_FCLK	Stopped	PRCM.CM_FCLKEN_USBHOST[0] EN_USBHOST1	Gated when the enable bit is set to 0
USBHOST_120M_FCLK	Stopped	PRCM.CM_FCLKEN_USBHOST[1] EN_USBHOST2	Gated when the enable bit is set to 0
USBHOST_L3_ICLK	Stopped	PRCM.CM_ICLKEN_USBHOST[0] EN_USBHOST, PRCM.CM_AUTOIDLE_USBHOST[0] AUTO_USBHOST	Gated when: 1. Enable bit is set to 0.
USBHOST_L4_ICLK	Stopped		2. Enable-autoidle bit pair is set to 1, and the clock is not requested by subsystem.
USBHOST_SAR_FCLK	Stopped	PRCM.PM_PWSTCTRL_USBHOST[4] SAVEANDRESTORE	Gated when the save-restore bit is set to 0, or when the domain is in OFF state after the save operation completes or in ON state after the restore operation completes.

4.7.7.10 WKUP Domain Clock Controls

Figure 4-54 shows the clock controls for the WKUP domain. Table 4-49 lists the clock-gating controls for the WKUP domain.

Figure 4-54. WKUP Domain Clock Controls

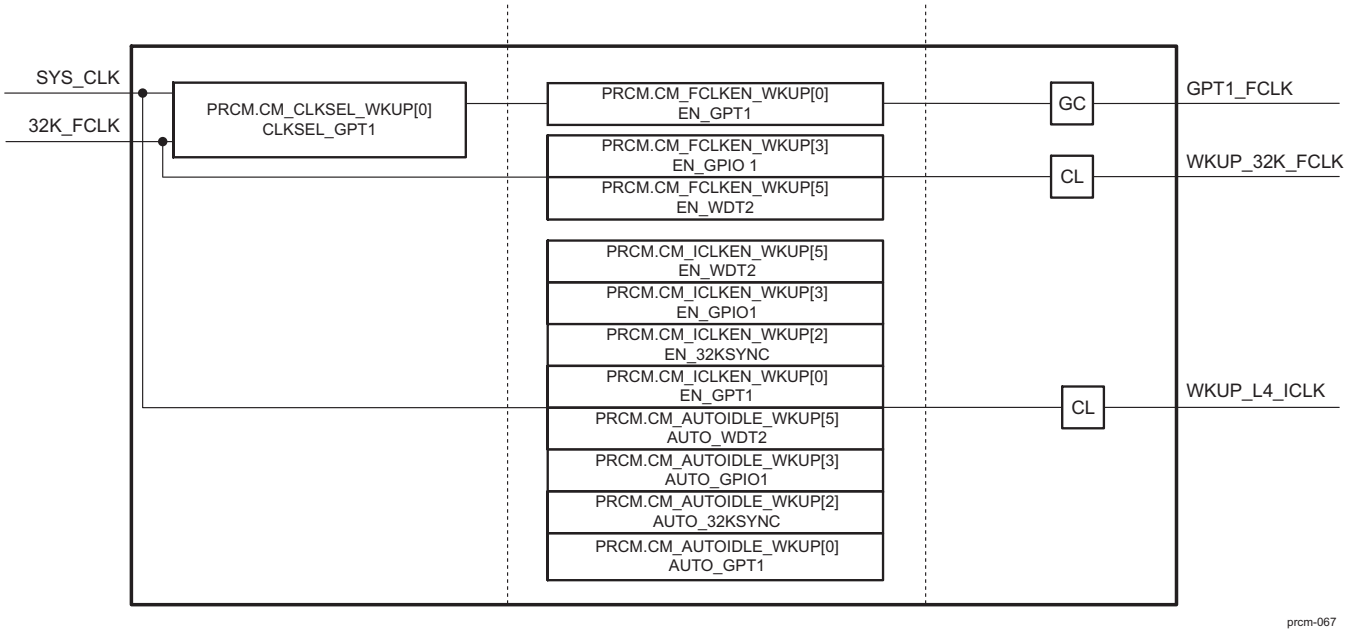


Table 4-49. WKUP Domain Clock-Gating Controls

Clock Name	Reset	Clock-Gating Control	Gating Description
GPT1_FCLK	Stopped	PRCM.CM_FCLKEN_WKUP[0] EN_GPT1	Gated when the enable bit is set to 0
WKUP_32K_FCLK	Stopped	PRCM.CM_FCLKEN_WKUP[3] GPIO1 and PRCM.CM_FCLKEN_WKUP[5] WDT2	Gated when the enable bits are set to 0
SECURE_32K_FCLK	Running	None	
WKUP_L4_ICLK	Running	PRCM.CM_ICLKEN_WKUP EN_(WDT[1,2], GPIO1, 32KSYNC, GPTIMER[1,12]), PRCM.CM_AUTOIDLE_WKUP AUTO_ (WDT[1,2], GPIO1, 32KSYNC, and GPTIMER[1,12])	Gated when: 1. All enable bits are set to 0. 2. All enable-autoidle bit pairs are set to 1, and the clock is not requested by any module.
USIM_FCLK	Running	PRCM.CM_FCLKEN_WKUP[9] EN_USIMOCP	Gated when the enable bit is set to 0

4.7.7.11 PER Domain Clock Controls

Figure 4-55 and Figure 4-56 show the clock controls for the PER domain.

Figure 4-55. PER Domain Clock Controls: Part 1

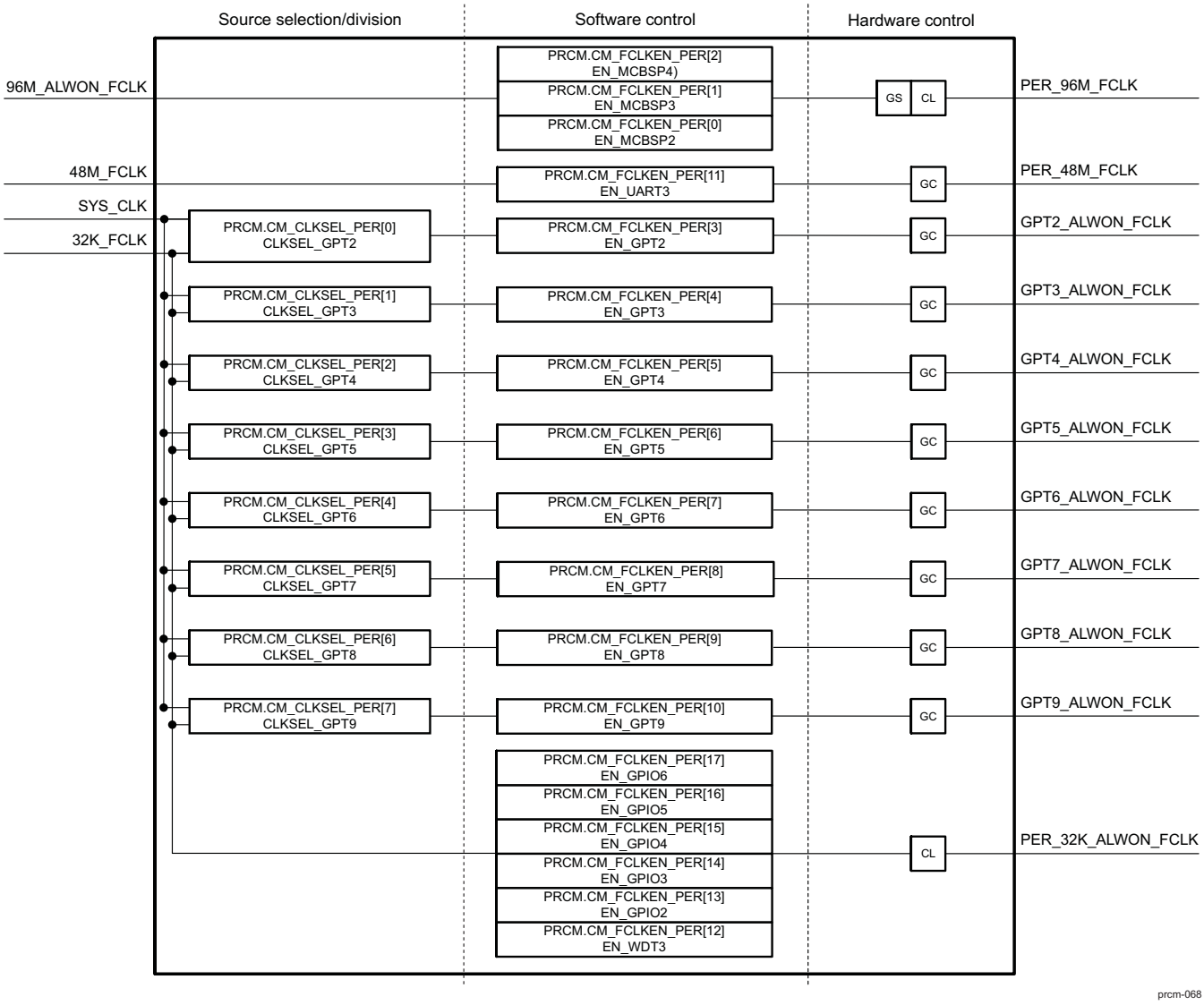


Figure 4-56. PER Domain Clock Controls: Part 2

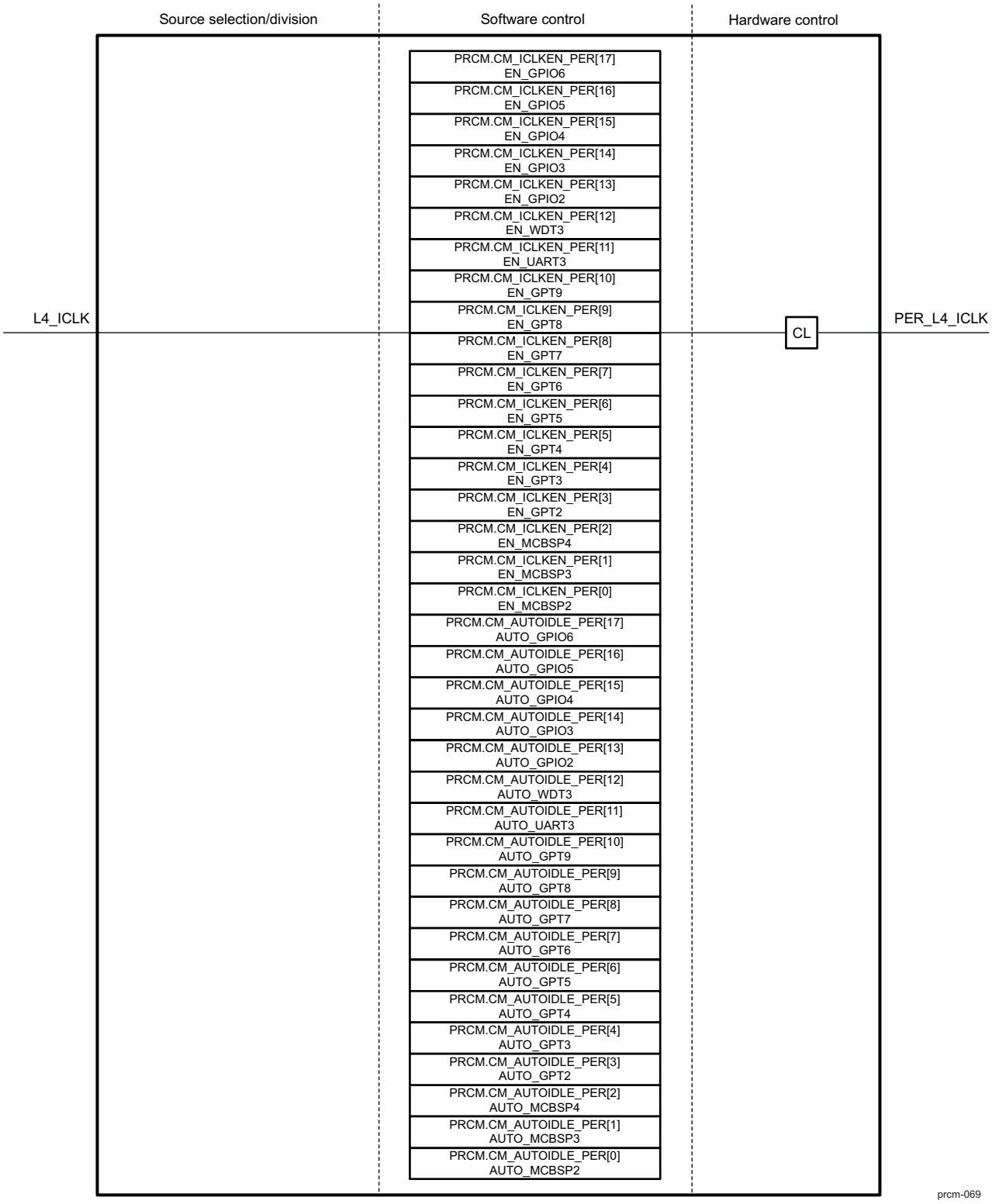


Table 4-50 lists the clock-gating controls for the PER domain.

Table 4-50. PER Domain Clock-Gating Controls

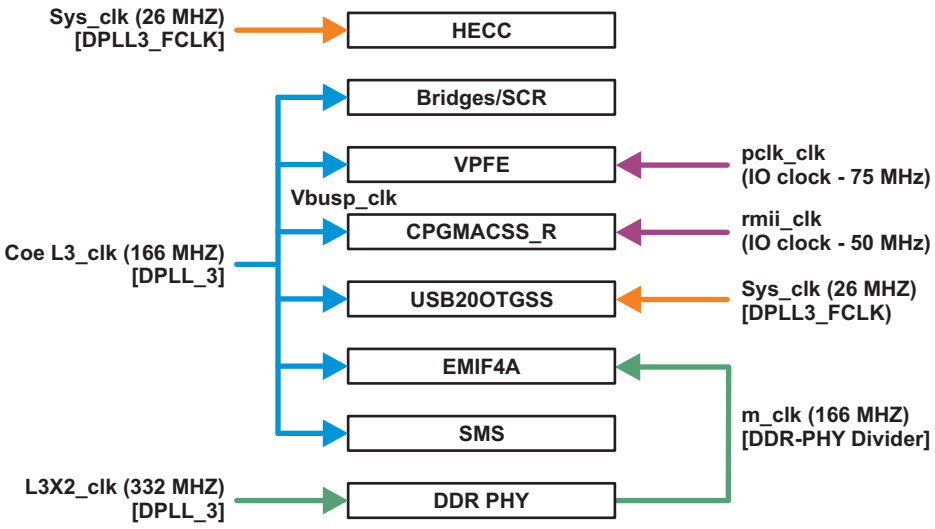
Clock Name	Reset	Clock-Gating Control	Gating Description
PER_48M_FCLK	Stopped	PRCM.CM_FCLKEN_PER[11] EN_UART3	Gated when the enable bit is set to 0
PER_96M_FCLK	Stopped	McBSP[2..4] input clock source select (in SCM) and PRCM.CM_FCLKEN_PER EN_MCBSP[2-4] and the DPLL4 operating mode	Gated when the enable bits of the module functional clock are set to 0 (the McBSPs can have MCBSP_CLKS as an alternate functional clock) or DPLL4 is in stop or bypass mode
MCBSP_CLKS	Stopped	See the *System Control Module* chapter.	
PER_32K_ALWON_ FCLK	Stopped	PRCM.CM_FCLKEN_PER EN_GPIO[2-6] and PRCM.CM_FCLKEN_PER[12] EN_WDT3	Gated when all the enable bits are set to 0
GPT2_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[3] EN_GPT2	Gated when the enable bit is set to 0
GPT3_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[4] EN_GPT3	Gated when the enable bit is set to 0
GPT4_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[5] EN_GPT4	Gated when the enable bit is set to 0
GPT5_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[6] EN_GPT5	Gated when the enable bit is set to 0
GPT6_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[7] EN_GPT6	Gated when the enable bit is set to 0
GPT7_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[8] EN_GPT7	Gated when the enable bit is set to 0
GPT8_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[9] EN_GPT8	Gated when the enable bit is set to 0
GPT9_ALWON_FCLK	Stopped	PRCM.CM_FCLKEN_PER[10] EN_GPT9	Gated when the enable bit is set to 0
PER_L4_ICLK	Stopped	PRCM.CM_ICLKEN_PER EN_(GPIO[2..6], WDT3, UART3, GPT[2..9], MCBSP[2..4]) and PRCM.CM_AUTOIDLE_PER AUTO_(GPIO[2..6], WDT3, UART3, GPT[2..9], and MCBSP[2..4])	Gated when: 1. All enable bits are set to 0. 2. All enable-autoidle bit pairs are set to 1, and the clock is not requested by any module.
96M_ALWON_FCLK	Stopped	None	Always-on clock

For audio applications, the McBSP functional clocks are provided externally by the MCBSP_CLKS pin. This clock must be permanently supplied to allow McBSPs to function. If the external clock input is selected as the functional clock, the PER domain sleep transition is prevented.

4.7.7.12 Other Modules Clocks

Device IPs get their clocks from existing clock sources or externally from an IO. Figure 4-57 shows the clock connections for the HECC, VPFE, CPGMACSS_R, USB20OTGSS, EMIF4A, SMS, DDR PHY and Bridges/SCR.

Figure 4-57. Clock Sources for Other Modules



4.7.8 Clock Configurations

The device supports several clock configurations. A clock configuration is a consistent set of divider ratios programmed into PRCM to obtain a certain combination of clock speed to match the performance requirement.

In the device, the MPU is connected to the interconnects through asynchronous bridges. The functional frequency of the processor can be configured independently of its interface clock frequency.

4.7.8.1 Processor Clock Configurations

Generic processor operating performance points may be defined as:

- DPLL1 (MPU DPLL) synthesized clock frequency (CLKOUTX2) configured by setting the M and N parameters of the DPLL
- DPLL1 (MPU DPLL) output clock frequency (MPU_CLK) configured by setting the M2 parameter of the DPLL

NOTE: The MPU_CLK is divided by 2 inside the MPU subsystem to generate the ARM_FCLK This divider is only active when the DPLL is locked. (Refer to MPU Subsystem for information on ARM_FCLK).

The frequency of the ARM processor clock (MPU_FCLK) can be configured. Table 4-51 identifies the processor clock, the source clock and the configuration register bits fields.

Table 4-51. Processor Clock Configuration Controls

Module	Clocks	Reference Clock	Multiplier (factors)	Divider (factors)	Configuration Bits
DPLL1	CLKOUTX2	SYS_CLK	M (0 ... 2047)		PRCM.CM_CLKSEL1_PLL_MPU[18:8] MPU_DPLL_MULT
				N (0 ... 127)	PRCM.CM_CLKSEL1_PLL_MPU[6:0] MPU_DPLL_DIV
	MPU_CLK[1]	CLKOUTX2		M2	PRCM.CM_CLKSEL2_PLL_MPU[4:0] MPU_DPLL_CLKOUT_DIV

(1) The MPU_CLK is divided by 2 inside the MPU subsystem to generate the ARM_FCLK This divider is only active when the DPLL is locked. (Refer to MPU Subsystem for information on ARM_FCLK).

Table 4-52. Processor Clock Configurations

CLOCK	BYPASS	LOCKED
MPU_CLK	DPLL1_FCLK (Bypass Clock from DPLL3)	(SYS_CLK * M * 2) / ([N+1] * M2)
ARM_FCLK	MPU_CLK	MPU_CLK / 2
MPUSS Internal Modules Clocks	ARM_FCLK / 2	ARM_FCLK / 2

4.7.8.2 Interface and Peripheral Functional Clock Configurations

The DPLL3 (CORE DPLL) generates the CORE_CLK which serves as the source clock for the L3_ICLK and L4_ICLK interface clocks of the device. The CORE_CLOCK is also used by the DPLL1as the bypass clock. The L3_ICLK is supplied to the SGX module as SGX_L3_ICLK and is used as its functional clock. The L4_ICLK is used by RM_L4_CLK as its source clock.

The interface and peripheral functional clocks frequencies can be configured according to the device performance requirements.

DPLL3 synthesized clock frequencies are configured as:

- $f_{CLKOUT} = (f_{SYS\_CLK} \times M) / (N+1)$
- $f_{CLKOUTX2} = f_{CLKOUT} \times 2$

DPLL3 output clock frequencies are configured as:

- $f_{CORE\_CLK} = f_{CLKOUT} / M2$
- $f_{COREX2\_CLK} = f_{CLKOUTX2} / M2$

L3_ICLK, L4_ICLK and RM_L4_ICLK frequencies are configured as:

- $f_{L3\_ICLK} = f_{CORE\_CLK} / DIV\_L3$
- $f_{L4\_ICLK} = f_{L3\_ICLK} / DIV\_L4$
- $f_{RM\_L4\_ICLK} = f_{L4\_ICLK} / DIV\_RM$

Table 4-53 identifies the interface clocks, their reference clocks, and the control bits for configuration of the interface clock frequencies.

Table 4-53. Interface Clock Configuration Controls

Module	Clock	Reference Clock	Multiplier (Factor)	Divider (Factor)	Configuration Bits
DPLL3	CLKOUT	SYS_CLK	M (0 ... 2047)		PRCM.CM_CLKSEL1_PLL[26:16] CORE_DPLL_MULT
	CLKOUTX2			N (0 ... 127)	PRCM.CM_CLKSEL1_PLL[14:8] CORE_DPLL_ DIV
CORE_CLK	CORE_CLK	CLKOUT		M2 (1 ... 31)	PRCM. CM_CLKSEL1_PLL[31 :27] CORE_DPLL_CLKOUT_DIV
	COREX2_CLK	CLKOUTX2			
L3 interconnect	L3_ICLK	CORE_CLK		DIV_L3 (1 ... 2)	PRCM. CM_CLKSEL_CORE[1:0] CLKSEL_L3
L4 interconnect	L4_ICLK	L3_ICLK		DIV_L4 (1 ... 2)	PRCM.CM_CLKSEL_CORE[3:2] CLKSEL_L4
RM clock	RM_L4_ICLK	L4_ICLK		DIV_RM (1 ... 2)	PRCM.CM_CLKSEL_WKUP[2:1] CLKSEL_RM

SGX_FCLK and DPLL1_FCLK (bypass mode) frequencies are configured as:

- $f_{SGX\_FCLK} = f_{CORE\_CLK} / DIV\_SGX$
- $f_{SGX\_FCLK} = f_{CM\_96M\_FCLK}$
- $f_{DPLL1\_FCLK} = f_{CORE\_CLK} / DIV\_DPLL1$

Table 4-54 identifies the functional clocks, their reference clocks and the control bits for configuration of the functional clock frequencies.

Table 4-54. Functional Clock Configuration Controls

Module	Clock	Reference Clock	Divider (Factor)	Configuration Bits
SGX	SGX_FCLK	CORE_CLK	DIV_SGX (3, 4, 6)	PRCM.CM_CLKSEL_SGX[2:0] CLKSEL_SGX
		CM_96M_FCLK		PRCM.CM_CLKSEL_SGX[2:0] CLKSEL_SGX
MPU HS bypass	DPLL1_FCLK	CORE_CLK	DIV_DPLL1 (1, 2, 4)	PRCM.CM_CLKSEL1_PLL_MPU[20:19] MPU_CLK_SRC

The rest of the functional clocks are issued from DPLL4 and DPLL5 and remain invariable, regardless of the interface clock configuration. In all clock configurations, any divider ratio to generate functional clocks is applicable, provided it complies with the maximum frequency specification.

4.8 PRCM Idle and Wake-Up Management

4.8.1 Overview

When a group of modules belonging to a clock domain do not require a clock (interface or functional), the PRCM can be programmed to automatically cut the clock to those modules, thereby reducing their power consumption. When all clocks in a domain are cut, the domain is idle.

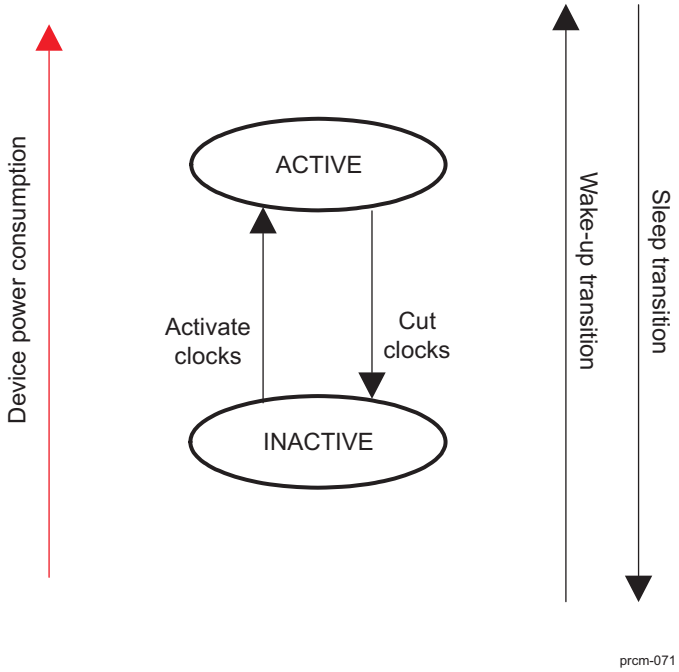
Similarly, when a module in low-power idle mode is required to switch to active mode, the PRCM activates the necessary clock signals to the module. This is a wake-up transition. Generally, a wake-up event triggers the wake-up transition.

A sleep/wake-up dependency can be defined between different domains. A sleep dependency ensures that a domain does not make a sleep transition unless all the dependent domains are in idle mode and do not require the domain. Similarly, a wake-up dependency ensures that a domain wakes up when any of its dependent domains wakes up.

The PRCM automatically handles the sequence clock-gating conditions for each domain, based on the configured dependencies between the domains and the clock-control bits of the modules.

Figure 4-58 shows the sleep/wake-up transition of the domains.

Figure 4-58. Domain Sleep/Wake-Up Transition



Functionally, the PRCM is composed of a single global control block and a power-control block for each domain. The domain power-control block handles wake-up events and clock-gating control for the domain. The domain power controllers communicate with each other for sleep dependencies and wake-up dependencies. All these blocks interact with the global control block that handles reset management, clock generation and distribution, and all PRCM registers.

Figure 4-59. Device Power Reset and Clock Controllers

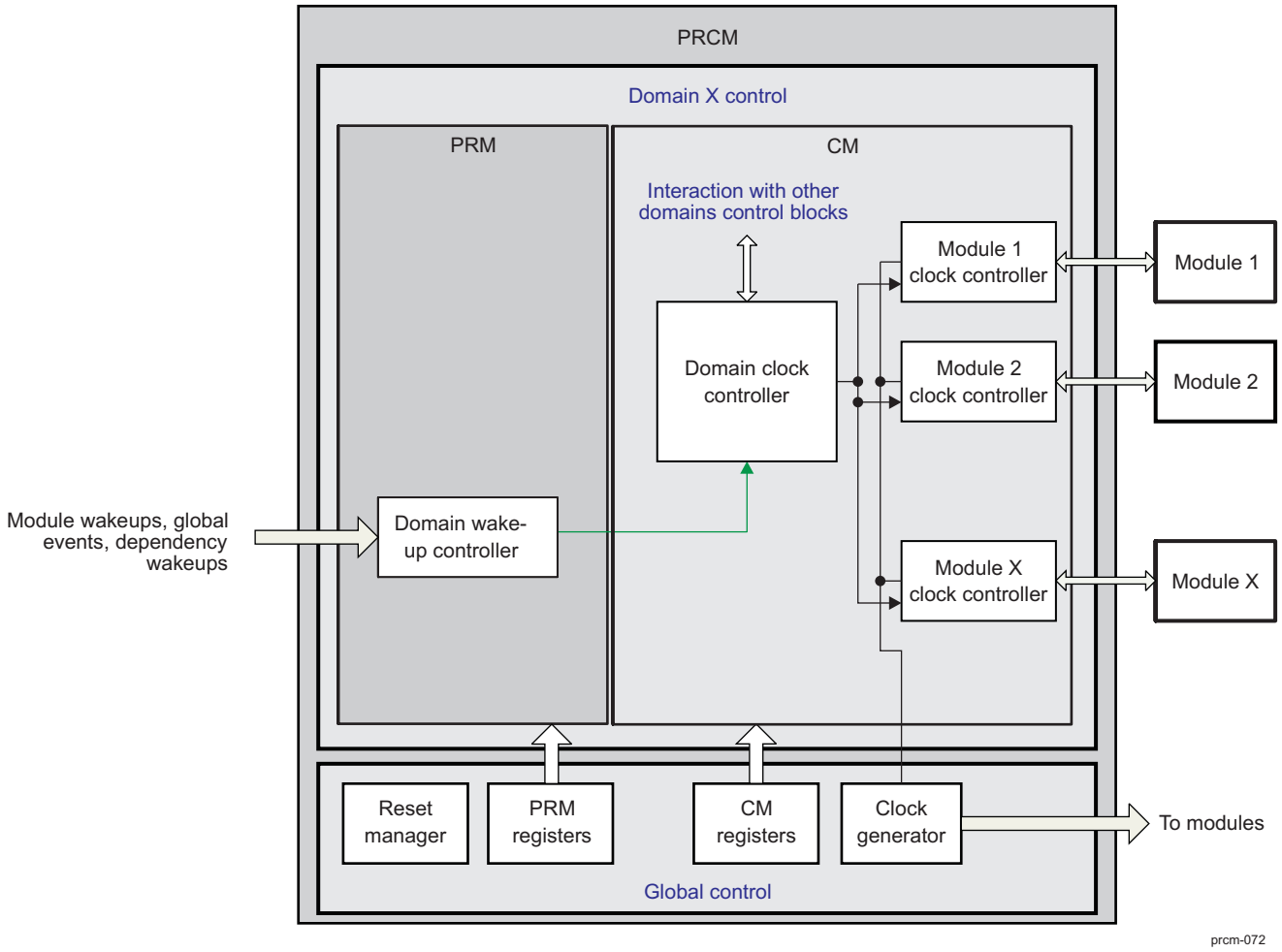


Figure 4-59 details the functions within one domain control block:

- A module clock controller (for example, module X clock controller) uses a hardware handshake protocol to communicate directly with the module. It controls the idle transition of the target module and responds to the standby requests of the initiator module (for details, see Section 4.1.3.2, *Autoidle Clock Control*). Depending on the mode, the clock controller sends clock commands (enable/disable) to the clock generator. The registers that control the functional and interface clocks in the module are directly mapped to the module clock controller.
- The domain clock controller gathers information from the following:
  - All module clock controllers of the domain
  - The domain wake-up controller
  - The other domain control blocks

  On a wake-up event, the domain clock controller starts the module domain clocks.
- The domain wake-up controller gathers all events that can wake up the domain.
- The reset manager globally controls all resets in the device. It gathers information from all domain control blocks to sequence the clocks, and resets the activation of each domain.
- The clock generator generates and distributes clocks over the device, depending on requests from all module clock controllers and on other global conditions.

4.8.2 Sleep Transition

The PRCM can initiate a domain sleep transition on a domain only if the domain meets the following conditions:

- All initiator modules are idle (they have completed their activity and idled themselves through software requests).
- All target modules are idle (automatically when all initiators are in standby mode or on software request).
- All sleep dependencies with other domains are met (can be set by software).

The domain state transition can be set to automatic (hardware controlled) or software-controlled by setting the PRCM. CM_CLKSTCTRL_<domain> CLKTRCTRL_<domain> bit field.

4.8.3 Wakeup

There are three types of wake-up events:

- Global: Generated on a particular device event (device wakeup, DPLL recalibration, etc.). Used mainly to wake up the MPU domain
- Module: Functional wake-up event issued from a module, which wakes up the domain where the module resides. It can also directly wake up the MPU depending on software settings in the PRCM.PM_MPUGRPSEL_<domain>.
- Dependency: A domain can wake up on the wakeup of another domain. The dependency is software-controllable by configuring the PRCM.PM_WKDEP_<domain> register.

4.8.4 Device Wake-Up Events

This section summarizes the wake-up events for each domain. Table 4-55 through Table 4-63 list the wake-up events, related control registers, and MPU interrupts.

The register PRCM.PRM_IRQENABLE_MPU enables the MPU interrupts. In some cases it can also enable the wake-up feature associated with the interrupt (DPLL recalibration requests and device wake-up event).

NOTE:

- The PRCM can be configured to generate an interrupt to the MPU subsystem as a result of a wake-up event from the CORE, WKUP, and PER domain modules to the MPU. However, these modules may also directly interrupt the MPU subsystem. To avoid a double interrupt (from the PRCM and one of these modules) as the result of a single event (wakeup), one interrupt must be masked when the other is unmasked. For further information on its interrupt capability, see the chapter of the corresponding module.
- The UART, GPIO, McSPI, and USIM OCP modules generate an asynchronous wake-up event (that is, their interface and functional clocks can be gated during the sleep period). However, because the GPTIMERs generate a synchronous wake-up event, they require their functional clock to be active during the sleep period; their interface clock can be gated. McBSP modules can generate a synchronous or asynchronous wake-up event, based on the mode configurations. For further information on its wake-up capability, see the chapter of the corresponding module.
- The ability of a module to generate a wake-up event depends on the state of the domain in which the module resides. If the domain is inactive (that is, both the functional and interface clocks of the domain are gated), only the asynchronous wake-up modules can wake up the domain. If only the interface clocks in the domain are gated, both synchronous and asynchronous wake-up modules can generate a wake-up event to activate the interface clock.

Table 4-55. MPU Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
CORE, DSS, and PER domain dependency	PRCM	PM_WKDEP_MPU	Yes	No	N/A
Peripherals wake-up events	Peripherals	PM_MPUGRPSEL1_CORE	Yes	MPU peripheral group event occurred.	MPU
		PM_MPUGRPSEL3_CORE			
		PM_MPUGRPSEL_WKUP			
		PM_MPUGRPSEL_PER			
Event generator on, off time	PRCM	PM_EVGENCTRL_MPU	Yes	Event generator on, off	MPU
Forced wake-up transition	PRCM	CM_CLKSTCTRL_NEON	No	Wake-up transition is complete (NEON, SGX, USBHOST, DSS, PER, EMU domains).	MPU
		CM_CLKSTCTRL_SGX			
		CM_CLKSTCTRL_DSS			
		CM_CLKSTCTRL_PER			
		CM_CLKSTCTRL_USBHOST			
		CM_CLKSTCTRL_EMU			
Forced sleep transition	PRCM	CM_CLKSTCTRL_NEON	No	Sleep transition is complete (NEON, SGX, USBHOST, DSS, PER, EMU domains).	MPU
		CM_CLKSTCTRL_SGX			
		CM_CLKSTCTRL_DSS			
		CM_CLKSTCTRL_PER			
		CM_CLKSTCTRL_USBHOST			
		CM_CLKSTCTRL_EMU			
DPLL1 recalibration request	PRCM	N/A	Yes	MPU DPLL recalibration event	MPU
DPLL3 recalibration request	PRCM	N/A	Yes	CORE DPLL recalibration event	MPU
DPLL4 recalibration request	PRCM	N/A	Yes	Peripheral DPLL recalibration event	MPU
DPLL5 recalibration request	PRCM	N/A	Yes	Peripheral DPLL2 recalibration event	MPU

Table 4-56. NEON Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
MPU domain dependency	PRCM	PM_WKDEP_NEON	Yes	No	N/A
Forced wake-up transition	PRCM	CM_CLKSTCTRL_NEON	Yes	Wake-up transition is complete.	MPU

Table 4-57. SGX Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
MPU domain dependency	PRCM	PM_WKDEP_SGX	Yes	No	N/A
WKUP domain dependency	PRCM	PM_WKDEP_SGX	Yes	No	N/A

Table 4-57. SGX Domain Wake-Up Events (continued)

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
Forced wake-up transition	PRCM	CM_CLKSTCTRL_SGX	Yes	Wake-up transition is complete.	MPU

Table 4-58. CORE Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
MPU domain dependency	PRCM	Hardware set (always-enabled)	Yes	No	N/A
DSS domain dependency	PRCM		Yes	No	N/A
USBHOST domain dependency	PRCM		Yes	No	N/A
PER domain dependency	PRCM		Yes	No	N/A
SGX domain dependency	PRCM		Yes	No	N/A
WKUP domain dependency	PRCM		Yes	No	N/A
IPSS wakeup	IPSS	PM_processor>GRPSEL1_ CORE, PM_WKEN1_CORE	Yes	No	N/A
McBSP1 wakeup	McBSP 1		Yes	No	N/A
McBSP5 wakeup	McBSP 5		Yes	No	N/A
GPTIMER10 wakeup	GPTIMER 10		Yes	No	N/A
GPTIMER11 wakeup	GPTIMER 11		Yes	No	N/A
UART1 wakeup	UART 1		Yes	No	N/A
UART2 wakeup	UART 2		Yes	No	N/A
McSPI1 wakeup	McSP 1		Yes	No	N/A
McSPI2 wakeup	McSP 2		Yes	No	N/A
McSPI3 wakeup	McSP 3		Yes	No	N/A
McSPI4 wakeup	McSP 4		Yes	No	N/A
MMC1 wakeup	MMC 1		Yes	No	N/A
MMC2 wakeup	MMC 2		Yes	No	N/A
MMC3 wakeup	MMC 2		Yes	No	N/A
USBTLL wakeup	USBTLL		Yes	No	N/A

Table 4-59. DSS Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
MPU domain dependency	PRCM	PM_WKDEP_DSS register	Yes	No	N/A
WKUP domain dependency	PRCM	PM_WKDEP_DSS register	Yes	No	N/A
Forced transition state wakeup	PRCM	CM_CLKSTCTRL_DSS register	Yes	Wake-up transition is complete.	MPU

Table 4-60. USBHOST Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
MPU domain dependency	PRCM	PM_WKDEP_USBHOST register	Yes	No	N/A
CORE domain dependency	PRCM	PM_WKDEP_USBHOST register	Yes	No	N/A
WKUP domain dependency	PRCM	PM_WKDEP_USBHOST register	Yes	No	N/A
USBHOST wake-up	HS USB Host	PM_processor>GRPSEL_ USBHOST, PM_WKEN_USBHOST	Yes	No	N/A
Forced transition state wakeup	PRCM	CM_CLKSTCTRL_USBHOST register	Yes	Wake-up transition is complete.	MPU

Table 4-61. PER Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
MPU domain dependency	PRCM	PM_WKDEP_PER register	Yes	No	N/A
CORE domain dependency	PRCM	PM_WKDEP_PER register	Yes	No	N/A
WKUP domain dependency	PRCM	PM_WKDEP_PER register	Yes	No	N/A
McBSP2 wakeup	McBSP 2	PM_processor>GRPSEL1_ PER, PM_WKEN_PER	Yes	No	N/A
McBSP3 wakeup	McBSP 3		Yes	No	N/A
McBSP4 wakeup	McBSP 4		Yes	No	N/A
GPTIMER2 wakeup	GPTIMER2		Yes	No	N/A
GPTIMER3 wakeup	GPTIMER3		Yes	No	N/A
GPTIMER4 wakeup	GPTIMER4		Yes	No	N/A
GPTIMER5 wakeup	GPTIMER5		Yes	No	N/A
GPTIMER6 wakeup	GPTIMER6		Yes	No	N/A
GPTIMER7 wakeup	GPTIMER7		Yes	No	N/A
GPTIMER8 wakeup	GPTIMER 8		Yes	No	N/A
GPTIMER9 wakeup	GPTIMER9		Yes	No	N/A
UART3 wakeup	UART3		Yes	No	N/A
GPIO2 wakeup	GPIO 2		Yes	No	N/A
GPIO3 wakeup	GPIO 3		Yes	No	N/A
GPIO4 wakeup	GPIO 4		Yes	No	N/A
GPIO5 wakeup	GPIO 5		Yes	No	N/A
GPIO6 wakeup	GPIO 6		Yes	No	N/A
Forced transition state wakeup	PRCM	CM_CLKSTCTRL_PER register	Yes	Wake-up transition is complete.	MPU

Table 4-62. EMU Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
EMU wakeup	ICEPick-C	N/A	Yes	No	N/A
Forced transition state wakeup	PRCM	CM_CLKSTCTRL_EMU register	Yes	Wake-up transition is complete.	MPU

Table 4-63. WKUP Domain Wake-Up Events

Internal Wake-Up Events	Source Module	PRCM Software Control	Wake-Up Event	Interrupt/Type	Interrupt to
Device wakeup	PRCM	PM_WKEN_WKUP	Yes	No	N/A
USIMOCP wakeup	USIMOCP	PM_processor>GRPSEL_ WKUP, PM_WKEN_WKUP	Yes	No	N/A
GPTIMER1 wakeup	GPTIMER1		Yes	No	N/A
GPTIMER12 wakeup	GPTIMER12		Yes	No	N/A
GPIO1 wakeup	GPIO 1	PM_WKEN_WKUP	Yes	No	N/A

4.8.5 Sleep and Wake-Up Dependencies

4.8.5.1 Sleep Dependencies

The (clock activity) dependencies between domains are implemented to manage their sleep and wake-up transitions to ensure stable operation of the device.

A domain sleep transition is achieved when all its clock domains (that is, functional and interface) are gated. For the gating of a clock domain, the following conditions must be satisfied:

- All initiator modules in the clock domain are in standby mode, that is, they cannot initiate any new transitions.
- All target modules in the clock domain are in idle mode and have no pending transitions.
- If the domain depends on another domain (that is, has a sleep dependency), the clock domains of the other domain must be muted.

A clock domain is said to be muted when all its public initiator modules (that is, the initiator modules of the clock domain that can generate interconnect transactions towards targets outside the clock domain) are in standby mode, and the domain cannot initiate new interconnect transactions toward other domains.

Table 4-64 describes the mute conditions for the clock domains.

Table 4-64. Clock Domain Mute Conditions

Clock Domain Name	Clock Domain Composition	Mute Conditions
MPU	MPU domain and MPU INTC in CORE domain	MPU in stand-by, MPU INTC idle
SGX	SGX domain	SGX in stand-by
DSS	DSS domain	DSS in stand-by
PER	PER domain	N/A (No public Initiator)
CORE_L3	L3 interconnect, L3 targets/initiators in CORE domain	sDMA in standby
CORE_L4	L4 interconnect, L4 targets in CORE domain	N/A (No public Initiator)
USBHOST	USBHOST domain	USBHOST in stand-by
WKUP	WKUP domain	N/A (No public Initiator)

A sleep dependency prevents the PRCM from gating the clocks to a domain (for sleep transition to inactive from on state) if a dependent domain is active (i.e., its clocks are active). A domain may depend on several other domains.

A domain sleep dependency is programmed by setting the PRCM.CM_SLEEPDEP_<domain> register.

Example:

The PER domain has a programmable sleep dependency with the MPU domain, and a hardwired sleep dependency with the CORE domain. Therefore, if all software dependencies are enabled, the PER domain can go into idle mode only if the MPU and CORE-L3 clock domains are idle. If all software dependencies are disabled, the PER domain can go into idle mode, provided that the CORE-L3 clock domain is muted (no access to the PER domain can be pending).

Table 4-65 summarizes the programmable and hardwired sleep dependencies among the domains.

Table 4-65. Sleep Dependencies

		Sleep Dependency									
Power Domain	**Clock Domain**	**MPU**	**NEON**	**SGX**	**DSS**	**PER**	**CORE_ L3**	**CORE_ L4**	**CORE_ CM**	**USB HOST**	**WKUP**
MPU	MPU	n/a	0	0	0	0	0	0	0	0	0
NEON	MPU	1	n/a	0	0	0	0	0	0	0	0
SGX	SGX	RW	0	n/a	0	0	0	0	0	0	0
DSS	DSS	RW	0	0	n/a	0	RW	0	0	0	0
PER	PER	RW	0	0	0	n/a	1	0	0	0	0
CORE	CORE_L3	1	0	1	1	1	n/a	1	0	1	1
	CORE_L4	1	0	0	0	0	1	n/a	0	0	0
	CORE_CM	1	1	1	1	1	1	1	n/a	1	1
USBHOST	USBHOST	RW	0	0		0	0	0	0	n/a	0
WKUP	WKUP	1	0	0	0	0	1	0	0	0	n/a

Notes:

No software control (hardwired values):	0	Does not depend on
	1	Depends on
Software controllable	RW	Read and write
	n/a	Not applicable

NOTE: The first row of the table identifies the dependent domains for a domain listed in the first column. Therefore, to identify the sleep dependency of domain X, identify domain X in the first column and its sleep dependency with domains (of the first row) is identified by its row.

4.8.5.2 Wake-Up Dependencies

A wake-up dependency allows a domain to wake up (from inactive to on state) when another domain wakes up (from off to on state). For example, domain one (PD1) provides a service to domain two (PD2), which creates a dependency between the two domains. When the dependent domain (PD2) wakes up, it signals PD1 with a broadcasted wake-up event, causing PD1 to wake up.

A broadcasted wake-up event can originate from one of two sources:

- PD2 internal wake-up event (typically, a peripheral wake-up event)
- Abortion of the mute mode (at least one initiator in the domain exits standby mode) of PD2 when both the sleep dependency and wake-up dependency with PD1 are enabled.

NOTE: The domain wake-up dependency is a nontransitive property.

If a domain PD1 has wake-up dependency with domain PD2, i.e., PRCM.PM_WKDEP_PD1. EN_PD2 bit is set to 1 and if PD2 has wake-up dependency with domain PD3, i.e., PRCM.PM_WKDEP_PD2. EN_PD3 bit is set to 1.

However, if PD1does not have a direct wake-up dependency with PD3, i.e., PRCM.PM_WKDEP_PD1. EN_PD3 bit is set to 0. Then if the PD1 and PD2 are INACTIVE and PD3 is INACTIVE and is woken-up by a wake-up event, the PD2 is also woken-up but not the PD1.

Because a domain can depend on several other domains, it can broadcast the wake-up signal to each domain on which it depends.

A domain wake-up dependency can either be hardwired (set in the hardware) or programmable through software by configuring the PM_WKDEP_<domain> register for that domain. Continuing the example of PD2 as dependent on PD1, setting the PM_WKDEP_PD1.PD2 bit means that when PD2 wakes up, PD1 also wakes up.

The MPU domain can be wakened only by the DSS, USBHOST, PER, CORE-L3, CORE-L4, and WKUP domains. All wake-up dependencies of the MPU domain are software-configurable.

When the wake-up dependency with other domains is software-programmable (that is, with the USBHOST, PER, CORE, and WKUP domains), two registers may need be configured: PM_WKDEP_<domain> and PM_<processor>GROUPSEL_<domain>.

The PM_WKDEP_<domain> register serves only to enable/disable the global wake-up dependency of the processor domain on these three domains. The PM_<processor>GROUPSEL_<domain> register must be configured to enable the particular wake-up event in these domains that will wake up the processor domain. Thus, the global wake-up dependency for a dependent domain must be enabled so that a wake-up event can occur, if it has been enabled in the corresponding GROUPSEL register.

For example, if the wake-up dependency of the MPU domain is to be enabled for a wake-up event from the GPIO2 module of the PER domain, the following configuration is required:

- PRCM.PM_MPUGRPSEL_PER[13] GRPSEL_GPIO2
- PRCM.PM_WKDEP_MPU[7] EN_PER

Table 4-66 summarizes the programmable and hardwired wake-up dependencies among the domains.

Table 4-66. Wake-Up Dependencies

		Wake-Up Dependency									
Power Domain	**Clock Domain**	**MPU**	**NEON**	**SGX**	**DSS**	**USB HOST**	**PER**	**CORE_ L3**	**CORE_ L4**	**CORE_ CM**	**WKUP**
MPU	MPU	n/a	0	0	RW	RW**	RW*	RW*	RW*	0	RW**
NEON	MPU	RW	n/a	0	0	0	0	0	0	0	0
SGX	SGX	RW	0	n/a	0	0	0	0	0	0	RW
DSS	DSS	RW	0	0	n/a	0	0	0	0	0	RW
USBHOST	USBHOST	RW	0	0	0	n/a	0	RW	0	0	RW
PER	PER	RW	0	0	0	0	n/a	RW	0	0	RW

Table 4-66. Wake-Up Dependencies (continued)

		Wake-Up Dependency									
Power Domain	**Clock Domain**	**MPU**	**NEON**	**SGX**	**DSS**	**USB HOST**	**PER**	**CORE_ L3**	**CORE_ L4**	**CORE_ CM**	**WKUP**
CORE	CORE_L3	1	0	0	1	1	1	n/a	1	0	1
	CORE_L4	1	0	0	0	0	0	0	n/a	0	1
	CORE_CM	1	0	0	1	1	1	1	1	n/a	1
WKUP	WKUP	1	0	0	0	1	1	1	1	0	n/a

Legend:

RW	Software wakeup dependency: dependency by PM_WKDEP_<> register.
RW*	Software wakeup dependency: dependency by PM_WKDEP_<> and PM_<>GRPSEL_<> registers.
RW**	Software wakeup dependency: dependency by PM_<>GRPSEL_<> registers.
1	Hardware wakeup dependency: dependency always enabled.
0	No wakeup dependency applicable.
n/a	Not applicable

4.8.6 Other Modules Idle/Wakeup Management

For the purposes of Idle and Wakeup Management in the device, a number of modules (HECC, CPGMAC, VPFE and USB20OTG) are grouped together in a domain called the IP subsystem (IPSS). The IPSS has a single Idle req/ack interface to the CM. When CM asserts ipss_idle_req signal to IPSS, ipss_idle_ack will be returned if all IPSS modules (HECC, CPGMAC, VPFE and USB20OTG) are clock gated. An MPU programmable register (IPSS_CLK_CTRL) is implemented in the system control module for gating the clocks to these modules.

To gate the VBUSP (interface) clock of a IPSS module, the MPU writes the <module>_vbusp_clk_en bit in IPSS_CLK_CTRL register. The MPU must poll the <module>_vbusp_clk_en_ack bit in IPSS_CLK_CTRL register to make sure that clock is gated. Due the clock gating procedure, the MPU must not send a VBUSP request to this module.

To exit clock gating mode for an IP, the MPU must clear <module>_vbusp_clk_en bit in IPSS_CLK_CTRL register. The <module>_vbusp_clk_en_ack bit will be updated to indicate when the clock is ungated.

The MPU can also gate the functional clock to a IPSS module by writing to the <module>_func_clk_en bit in the IPSS_CLK_CTRL register.

4.9 PRCM Interrupts

Table 4-67 lists the interrupts from the PRCM to the MPU.

Table 4-67. Interrupt Descriptions

Interrupt Name	Mapping	Description
PRCM_MPU_IRQ	M_IRQ_11	To MPU interrupt controller (MPU INTC)

Table 4-68 summarizes the event flags (in the PRM_IRQSTATUS_MPU register) and mask (in the PRM_IRQENABLE_MPU register) related to the events generating PRCM interrupts to the MPU.

Table 4-68. MPU Interrupt Event Descriptions

Event Flag	Event Mask	Map to	Automatic MPU Domain Wake-Up Event	Description
PRM_IRQSTATUS_ MPU[0] WKUP_ST	PRM_IRQENABLE_ MPU[0] WKUP_EN	PRCM_MPU_IRQ	No	MPU peripheral group wakeup
PRM_IRQSTATUS_ MPU[2] EVGENON_ST	PRM_IRQENABLE_ MPU[2] EVGENON_EN	PRCM_MPU_IRQ	No	Event generator end of on time
PRM_IRQSTATUS_ MPU[3] EVGENOFF_ST	PRM_IRQENABLE_ MPU[3] EVGENOFF_EN	PRCM_MPU_IRQ	No	Event generator end of off time
PRM_IRQSTATUS_ MPU[4] TRANSITION_ST	PRM_IRQENABLE_ MPU[4] TRANSITION_EN	PRCM_MPU_IRQ	No	A sleep or wake-up transition completion event for NEON, SGX, DSS, PER, EMU, USBHOST domains
PRM_IRQSTATUS_ MPU[5] CORE_DPLL_ST	PRM_IRQENABLE_ MPU[5] CORE_DPLL_ RECAL_EN	PRCM_MPU_IRQ	Yes	DPLL3 recalibration event
PRM_IRQSTATUS_ MPU[6] PERIPH_DPLL_ST	PRM_IRQENABLE_ MPU[6] PERIPH_DPLL_ RECAL_EN	PRCM_MPU_IRQ	Yes	DPLL4 recalibration event
PRM_IRQSTATUS_ MPU[7] MPU_DPLL_ST	PRM_IRQENABLE_ MPU[7] MPU_DPLL_ RECAL_EN	PRCM_MPU_IRQ	Yes	MPU DPLL recalibration event
PRM_IRQSTATUS_ MPU[9] IO_ST	PRM_IRQENABLE_ MPU[9] IO_EN	PRCM_MPU_IRQ	No	I/O pads wake-up event
PRM_IRQSTATUS_ MPU[25] SND_PERIPH_ DPLL_RECAL_ST	PRM_IRQENABLE_ MPU[25] SND_PERIPH_ DPLL_RECAL_EN	PRCM_MPU_IRQ	Yes	DPLL5 recalibration event

NOTE: The software must first read the event flag to know the interrupt cause, and then write 1 to it to clear the flag.

4.10 PRCM Voltage Management Functional Description

This section describes the voltage control architecture. It also explains the interactions between the device and the external power IC.

NOTE: For more information regarding the device power pads connection, please refer to your device-specific data manual.

4.11 PRCM Basic Programming Model

The PRCM supports an extensive set of module-specific registers that allow programming control over numerous features of the clocks, resets, and power-management signals for each domain of the device.

These registers are fully programmable and accessible by the MPU subsystem.

Logically, the registers are grouped into five categories:

- Global
- Clock management
- Reset management
- Power management
- Voltage management

4.11.1 Global Registers

4.11.1.1 Revision Information Registers

- CM_REVISION: Indicates the CM module revision code; it is read-only
- PRM_REVISION: Indicates the PRM module revision code; it is read-only

4.11.1.2 PRCM Configuration Registers

- CM_SYSCONFIG: Holds an AUTOIDLE bit to control the CM internal clock autogating feature
- PRM_SYSCONFIG: Holds an AUTOIDLE bit to control the PRM internal clock autogating feature

4.11.1.3 Interrupt Configuration Registers

The PRM can interrupt the MPU as a result of several events:

- PRM internal event (event generator, sleep transition, wake-up transition)
- Peripheral wake-up event for a peripheral with interrupt capability (GPTIMER[1..12], GPIO[1..6], McBSP[1..5], UART[1..3], IPSS, I2C[1..3], McSPI[1..4], MMC[1,2])
- Module/device-level event not associated with any interrupt (DPLL recalibration, I/O wakeup)

The PRM interrupt is enabled by programming the PRM_IRQENABLE_<processor_name> register; the interrupt status can be read from the PRM_IRQSTATUS_<processor_name> register.

The device has the following processor interrupt registers:

- PRM_IRQSTATUS_MPU
- PRM_IRQENABLE_MPU

4.11.1.3.1 MPU Interrupt Event Sources

The MPU interrupt registers correspond to the interrupt sources connected to the interrupt line mapped to the MPU interrupt controller.

Multiple events can activate this interrupt line:

- MPU peripheral group wake-up event
- Event-generator module end-of-on time and end-of-off time events
- Sleep or wake-up transition (SGX, USBHOST, PER, DSS, NEON, EMU domains)
- DPLL1/DPLL3/DPLL4/DPLL5 recalibration request
- I/O pad wake-up event

The end-of-on time period and end-of-off time period events of the event generator module are sources of interrupts to the MPU processor (the corresponding bits in the PRM_IRQENABLE_MPU are set to 1). The end-of-OFF time period is the source of wake-up events on the MPU domain.

The MPU can force a sleep or wake-up transition on some domains (SGX, USBHOST, PER, DSS, NEON, EMU). The PRM triggers the MPU interrupt when the domain enters a power state. The software must clear the CM_CLKSTCTRL_<domain_name> register only after getting the interrupt. If the software clears this bit before getting the interrupt, the interrupt never occurs, regardless of whether the transition occurs.

An interrupt for a peripheral with wake-up capability is enabled when the wake-up enable bit (PM_WKEN_<domain_name> register type) and group select bit (PM_<processor_name>GRPSEL_<domain_name> type of register) are set to 1.

The PRM triggers the MPU interrupt as long as the DPLL recalibration flag is set and the corresponding interrupt enable bit in the PRM_IRQENABLE_<processor_name> register is set to 1. The recalibration flag is set by the DPLL and remains active if the DPLL is not reinitialized.

4.11.1.3.2 MPU Interrupt Registers

4.11.1.3.2.1 PRM_IRQENABLE_MPU (MPU Interrupt Enable Register)

The MPU interrupt enable register allows independent masking/unmasking of each MPU internal interrupt source.

NOTE: If the following interrupts are enabled and the MPU domain is idled, then when the event occurs, the PRCM sets the interrupt that wakes up the domain:

- DPLL1/DPLL3/DPLL4/DPLL5 recalibration event

4.11.1.3.2.2 PRM_IRQSTATUS_MPU (MPU Interrupt Status Register)

The MPU interrupt status register provides the status of all PRCM internal events that can generate an MPU interrupt. Software must read this register to identify the interrupt cause, and then clear the pending interrupt by setting the corresponding bit to 1.

4.11.1.4 Event Generator Control Registers

For details about the event generator module, see the public *ARM® Cortex™-A8 Technical Reference Manual*.

In the PRCM, three event generator registers allow configuration of the event generator module:

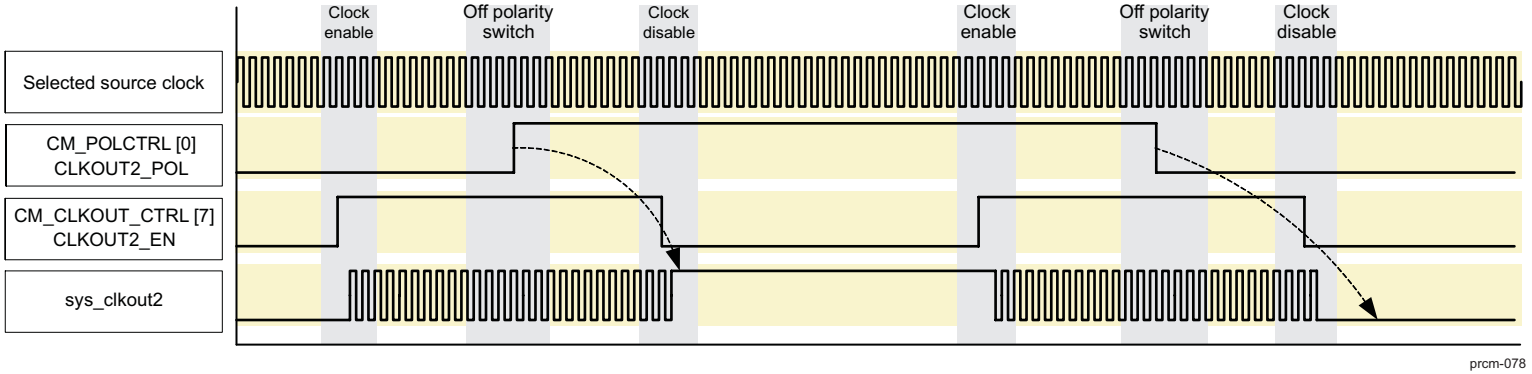
- PM_EVGENCTRL_MPU (event generator control register): Event-generator settings
- PM_EVGENONTIM_MPU (event generator on-time register): On-time duration setting
- PM_EVGENOFFTIM_MPU (event generator off-time register): Off-time duration setting

4.11.1.5 Output Signal Polarity Control Registers

4.11.1.5.1 CM_POLCTRL (CM Polarity Control Register)

The CM_POLCTRL register allows the setting of the polarity of sys_clkout2 when gated (disabled). sys_clkout2 can be gated to a low level or a high level, depending on the software programming of this register. Figure 4-60 shows the normal behavior of sys_clkout2 when gated.

Figure 4-60. sys_clkout2 Gating Polarity Control



4.11.1.5.2 PRM_POLCTRL (PRM Polarity Control Register)

The PRM polarity control register allows the setting of the polarity of the external signals controlled by the PRM. It contains the following polarity control bits:

- CLKOUT_POL: Polarity of sys_clkout1
- CLKREQ_POL: Polarity of sys_clkreq
- EXTVOL_POL: Polarity of both sys_nvmode signals: sys_nvmode1 and sys_nvmode2

At device power up, CLKREQ_POL = 1. Depending on the system hardware implementation, these signals may or may not be used during device power up. For details about using these signals, see Section 4.7, *Clock Manager Functional Description*,

4.11.2 Clock Management Registers

4.11.2.1 System Clock Control Registers

4.11.2.1.1 PRM_CLKSRC_CTRL (Clock Source Control Register)

The clock source control register is dedicated to the clock source controls of the device. It contains the following bit fields:

- SYSCLKSEL: Indicates the oscillator mode (oscillator/bypass) and is automatically updated at power up
- AUTOEXTCLKMODE: Enables autogating for the system clock, depending on the device power state. The clock can be configured to be automatically gated when all domains are in idle state. When the gating condition is satisfied, the internal oscillator is turned off, if it is used. Otherwise, sys_clkreq is asserted to notify the external clock source to turn off.

NOTE: Power consumption is reduced with the SYS_CLK off; however, wake-up latency is increased because of oscillator stabilization time after the clock is turned on again.

- SYSCLKDIV: Input divider (1, 2) of the system clock,

4.11.2.1.2 PRM_CLKSETUP (Source-Clock Setup Register)

The source-clock setup register allows the setting of the setup time of the oscillator system clock, based on the number of 32-kHz clock cycles. This duration corresponds to the time required by the oscillator to stabilize before propagating the system clock.

Because the reset lasts long enough for oscillator stabilization, this register is not used at power-on reset of the device. This is ensured either by the external reset source (power IC) or by the reset extension in the PRCM (RSTTIME0 timer). The PRM_CLKSETUP register is cleared on cold reset only.

4.11.2.1.3 PRM_CLKSEL (Source-Clock Selection Register)

The PRCM.PRM_CKSEL[2:0] SYS_CLKIN_SEL bit field is used to select the input frequency of the system clock (12, 13, 16.8, 19.2, 26 or 38.4 MHz).

4.11.2.2 External Clock Output Control Registers

4.11.2.2.1 PRM_CLKOUT_CTRL (Clock Out Control Register)

The PRM clock out control register provides control to enable and disable the gating of the sys_clkout1 output clock.

4.11.2.2.2 CM_CLKOUT_CTRL (Clock Out Control Register)

The CM clock out control register provides control over the device output clock sys_clkout2, which can be used externally for functional or test purposes. The register allows the following:

- Selection of the source clock for sys_clkout2:
  - CORE_CLK
  - SYS_CLK
  - 96-MHz clock
  - 54-MHz clock
- Dividing-down the selected source clock by 1, 2, 4, 8, or 16
- Enabling/disabling of the gating of sys_clkout2

4.11.2.3 DPLL Clock Control Registers

A set of registers controls the clock features of the five DPLLs (DPLL1, DPLL3, DPLL4, and DPLL5):

- CM_CLKSELn_PLL_<processor_name>
- CM_CLKSELn_PLL
- CM_CLKEN_PLL_processor_name>
- CM_CLKEN_PLL
- CM_AUTOIDLE_PLL_processor_name>
- CM_AUTOIDLE_PLL
- CM_IDLEST_PLL_<processor_name>
- CM_IDLEST_CKGEN
- CM_IDLEST2_CKGEN

The following sections describe the purposes of these registers.

4.11.2.3.1 CM_CLKSELn_PLL_<processor_name> (Processor DPLL Clock Selection Register)

The processor DPLL clock selection register controls the clock configuration of DPLL1, including the following features:

- The multiplier (M) and divider (N) values of the DPLL
- Selection of the fast bypass clock (CORE_CLK or CORE_CLK/2) in bypass mode

- Configuration of DPLL output clock divider values (M2 factor)

The device has two DPLL clock selection registers for DPLL1:

- CM_CLKSEL1_PLL_MPU: DPLL1 multiplier, divider, and fast bypass clock selection
- CM_CLKSEL2_PLL_MPU: DPLL1 output clock divider selection

4.11.2.3.2 CM_CLKSELn_PLL (DPLL Clock Selection Register)

The DPLL clock selection register controls the clock configuration for DPLL3, DPLL4, and DPLL5, including the following features:

- The multiplier (M) and divider (N) values of the DPLL
- Configuration of DPLL output clock divider values

The device has following DPLL clock selection registers for DPLL3, DPLL4, and DPLL5:

- CM_CLKSEL1_PLL: DPLL3 multiplier, divider, and output clock division configuration. Source selection for the 54-MHz and 48-MHz clock (48M_FCLK) between DPLL4 output and sys_altclk.
- CM_CLKSEL2_PLL: DPLL4 multiplier and divider configuration
- CM_CLKSEL3_PLL: Divider configuration for 96-MHz clock (96M_FCLK)
- CM_CLKSEL4_PLL: DPLL5 multiplier and divider configuration
- CM_CLKSEL5_PLL: Divider configuration for 120-MHz clock (120M_FCLK)

4.11.2.3.3 CM_CLKEN_PLL_<processor_name> (Processor DPLL Clock Enable Register)

The processor DPLL clock enable register allows the enabling or disabling of DPLL1. It allows an immediate setting of the DPLL.

The device clock enable registers for DPLL1 is CM_CLKEN_PLL_MPU: DPLL1.

The CM_CLKEN_PLL_<processor_name> register allows programmable control of the following DPLL features:

- Low-power bypass mode/lock mode selection
- Automatic recalibration enable/disable
- Programmable internal frequency range of the DPLL
- Frequency ramping feature disabled and enabled with programmable step duration (4 s, 20 s, or 40 s)
- The DPLL LP mode can be enabled or disabled. However, switching between LP and normal mode is effective only when the DPLL has performed a recalibration; therefore, the DPLL must lock or relock. Also, the LP mode control is considered only during the following transitions:
  - From bypass to lock
  - From stop mode to lock
  - From lock to relock

NOTE: DPLL1 enters its internal power state (MNBYPASS) after being released from reset or when a multiplier value of 0 or 1 is loaded into the DPLL. Therefore, even if the CM_CLKEN_PLL_MPU[2:0] EN_MPU_DPLL bit field is reset to the low-power bypass mode, DPLL1 automatically transitions to MNBYPASS mode, because the multiplier value (the CM_CLKSEL1_PLL_MPU[18:8] MPU_DPLL_MULT bit field) resets to 0.

4.11.2.3.4 CM_CLKEN_PLL (DPLL Enable Register)

The DPLL enable register allows control of DPLL3 DPLL4, and DPLL5. It allows an immediate setting of the DPLLs. This register controls the following features of the two DPLLs:

- DPLL operation mode:
  - Low-power bypass, fast-relock bypass, and lock modes for DPLL3
  - Low-power bypass and lock modes for DPLL4
- Programmable automatic recalibration
- Programmable internal frequency range for the DPLL
- Frequency ramp-up feature disable/enable with programmable step duration
- The DPLL3 output M3X2 and the DPLL4 outputs M2X2, M3X2, M4X2, M5X2, and M6X2 clock paths can be powered down. However, the setting takes effect only when the output clock is gated. It is also powered down whenever the DPLL is in stop mode, regardless of the software settings.
- The DPLL LP mode can be enabled or disabled. However, switching between LP and normal mode is effective only when the DPLL has performed a recalibration; therefore, the DPLL must lock or relock. Also, the LP mode control is considered only during the following transition:
  - From bypass to lock
  - From stop mode to lock
  - From lock to relock

4.11.2.3.5 CM_AUTOIDLE_PLL_<processor_name> (Processor DPLL Autoidle Register)

The processor DPLL autoidle register allows the enabling/disabling of the automatic processor DPLL activity control. In automatic mode, the DPLL automatically enters low-power stop mode when the DPLL clock is not required. It is also restarted automatically.

The following is the device DPLL auto-control register:

- CM_AUTOIDLE_PLL_MPU: DPLL1 autoidle mode control.

4.11.2.3.6 CM_AUTOIDLE_PLL (DPLL Autoidle Register)

The DPLL autoidle register allows the enabling/disabling of the automatic mode-switching control for DPLL3 and DPLL4. This automatic mode takes effect only when the DPLLs are locked.

DPLL3 can be configured to automatically switch to either low-power bypass mode or low-power stop mode when the CORE domain clock is not required.

DPLL4 can be configured to automatically switch to low-power stop mode when the PER domain clock is not required.

4.11.2.3.7 CM_AUTOIDLE1_PLL (DPLL5 Autoidle Register)

The DPLL5 autoidle register allows the enabling/disabling of the automatic mode-switching control. This automatic mode takes effect only when the DPLL is locked.

When enabled, DPLL5 is automatically switched to low-power stop mode whenever the 120-MHz output clock is gated.

4.11.2.3.8 CM_IDLEST_CKGEN (Source-Clock Idle-Status Register)

The source-clock idle-status register provides a status of DPLL3 and DPLL4 clock activity. It also provides the status of the functional clocks derived from DPLL4 output.

The activity status for DPLL3 and DPLL4 can be:

- DPLL is bypassed.
- DPLL is locked.

The activity status for the functional clocks can be:

- The functional 96-MHz clock is active, or not.
- The functional 48-MHz clock is active, or not.
- The functional 12-MHz clock is active, or not.
- The functional 54-MHz clock is active, or not.

The functional 96-MHz clock and all other functional clocks derived from it are active only when DPLL4 is locked and the clock is required.

The functional 48-MHz and 12-MHz clocks are qualified as active only if the clock is required and when the external alternate clock is selected as the source clock.

The functional 54-MHz clock (DSS_TV_CLK) is active only when DPLL4 is locked, selected as the source clock, and required.

4.11.2.3.9 CM_IDLEST2_CKGEN (DPLL5 Source-Clock Idle-Status Register)

The source-clock idle-status register provides the status of DPLL5 clock activity. It also provides the status of the functional clocks derived from DPLL5 output.

The activity status for DPLL5 can be:

- DPLL is bypassed.
- DPLL is locked.

The activity status for the functional clocks can be:

- The 120-MHz functional clock is active, or not.
- The output stage of the 120-MHz functional clock is active, or not.

4.11.2.3.10 CM_IDLEST_PLL_<processor_name> (Processor DPLL Idle-Status Register)

The processor DPLL idle-status register indicates the status of the processor DPLL: whether it is in locked or bypass mode.

The device DPLL idle-status register is:

- CM_IDLEST_PLL_MPU: DPLL1 activity status.

4.11.2.4 Power-Domain Clock Control Registers

An identical set of registers controls the clock features of the domains in the device:

- CM_CLKSEL_<domain_name>
- CM_FCLKEN_<domain_name>
- CM_ICLKEN_<domain_name>
- CM_AUTOIDLE_<domain_name>
- CM_IDLEST_<domain_name>
- CM_CLKSTCTRL_<domain_name>
- CM_CLKSTST_<domain_name>
- CM_SLEEPDEP_<domain_name>

The following sections describe the purposes of these registers.

4.11.2.4.1 CM_CLKSEL_<domain_name> (Clock Select Register)

The clock select register controls the selection of the module or subsystem input clock frequency (except for the processor modules). Therefore, it deals only with modules or subsystems for which the frequency is scalable or selectable (the others have fixed and nonprogrammable frequencies). Both functional and interface clocks can be scaled. In most cases, their value is a divided value from the DPLL3 (CORE) or the DPLL4 (PER) output clock.

The device includes the following clock select registers:

- CM_CLKSEL_CORE: L3 and L4 interconnects and GPTIMER10 and 11 functional clocks
- CM_CLKSEL_DSS: DSS functional clock 1 and TV functional clock
- CM_CLKSEL_PER: GPTIMER2, 3, 4, 5, 6, 7, 8, and 9 functional clocks
- CM_CLKSEL_SGX: SGX subsystem functional clock
- CM_CLKSEL_WKUP: GPTIMER1 functional clock and reset manager counter clock and USIMOCP functional clock

The functional clock of the GPTIMER1 is selectable between the always-on 32K_FCLK and the system clock (SYS_CLK), but it is used with the 32-kHz clock.

The CM_CLKSEL_SGX register controls the SGX functional clock divider ratio, which is divided from the L3_ICLK source clock. Table 4-69 summarizes L3_ICLK ratio for different configurations of the register field.

Table 4-69. SGX Functional Clock Ratio Settings

CM_CLKSEL_SGX.CLKSEL_SGX	SGX_L3_FCLK
0x1	L3_ICLK
0x2	L3_ICLK/2
0x3	L3_ICLK/3
0x4	L3_ICLK/4

4.11.2.4.2 CM_FCLKEN_<domain_name> (Functional Clock Enable Register)

The functional clock enable register allows control of the functional clock activity of each module or subsystem. All module functional clocks are controllable by software, except for the MPU, interconnect, and memory subsystems, for which the clocks are automatically controlled by the PRCM.

The device has the following functional clock control registers:

- CM_FCLKEN1_CORE and CM_FCLKEN3_CORE: CORE domain nonsecure peripherals set
- CM_FCLKEN_DSS: DSS subsystem
- CM_FCLKEN_PER: PER domain peripherals set
- CM_FCLKEN_SGX: SGX subsystem
- CM_FCLKEN_WKUP: WKUP domain peripherals set
- CM_FCLKEN_USBHOST: HS USB Host subsystem

The software effect is immediate and direct. The functional clock is turned on as soon as the bit is set, and turned off if the bit is cleared and the clock is not required by any module. On module wakeup, the functional clock can be automatically restarted.

NOTE: The functional clock supplies the functional part of a module or subsystem, which is not operational without its functional clock. In some cases, a module or a subsystem may require multiple functional clocks.

4.11.2.4.3 CM_ICLKEN_<domain_name> (Interface Clock Enable Register)

The interface clock enable register allows control of the interface clock activity of each module or subsystem. This register provides control over each module in the device.

The device has following interface clock control registers:

- CM_ICLKEN1_CORE, CM_ICLKEN2_CORE, and CM_ICLKEN3_CORE: CORE domain peripherals set
- CM_ICLKEN_DSS: DSS subsystem
- CM_ICLKEN_PER: PER domain peripherals set
- CM_ICLKEN_SGX: SGX subsystem
- CM_ICLKEN_WKUP: WKUP domain peripherals set
- CM_ICLKEN_USBHOST: HS USB Host subsystem

This register has an immediate effect, causing the source of the interface clock to be effectively cut (if the appropriate bit is cleared and no module requires this clock) or activated (if the appropriate bit is set).

Independent functional and interface clock control registers provide potential power savings for each module. For example, because the configuration port and interface of the module are still active, disabling a functional clock of a module while keeping its interface clock active (leaving the module inactive but allowing access to its registers) reduces power consumption.

When the interface clock is disabled, the module cannot communicate with the reset of the device; therefore, it is in idle mode. For example, the interface clock of a peripheral can be disabled while its functional clock is active; it is then idled, from the device standpoint, while it can detect any external event. This configuration typically allows a main part of the device to go into idle mode while keeping a peripheral active and ready to wake up from an external event.

Because a module may or may not be able to function without its functional or interface clocks, power-management strategies must be adapted accordingly. This relation is programmable and is defined in the CLOCKACTIVITY bits of the module SYSCONFIG register; therefore, it requires consistent programming of the CM_FCLKEN and CM_ICLKEN registers.

NOTE: The interface clock ensures proper communication between any module and the interconnect (L3 or L4), in most cases supplying the module interface and registers.

4.11.2.4.4 CM_AUTOIDLE_<domain_name> (Autoidle Register)

The autoidle register holds an AUTOIDLE bit per module that belongs to the related domain. Each AUTOIDLE bit enables/disables automatic (hardware) gating of a module interface clock.

When AUTOIDLE and ICLKEN are set for a module, the module interface clock is managed automatically (that is, by hardware control) according to the domain clock activity; for example, stopped before a domain sleep transition and reenabled on wakeup. Table 4-70 lists the possible autoidle settings for the interface clock.

Table 4-70. Interface Clock Autoidle Settings

CM_AUTOIDLE.AUTO_<module>	CM_ICLKEN.EN_<module>	Interface Clock
0	0	Disabled
0	1	Enabled
1	0	Disabled
1	1	Automatic enabling/disabling

The device has the following autoidle control registers:

- CM_AUTOIDLE1_CORE, CM_AUTOIDLE2_CORE, and CM_AUTOIDLE3_CORE: CORE domain peripherals set
- CM_AUTOIDLE_DSS: DSS subsystem
- CM_AUTOIDLE_PER: PER domain peripherals set
- CM_AUTOIDLE_WKUP: WKUP domain peripherals set
- CM_AUTOIDLE_USBHOST: HS USB Host subsystem

NOTE: For the SGX module, the automatic idle mode is always enabled, and is not software-controllable.

4.11.2.4.5 CM_IDLEST_<domain_name> (Idle-Status Register)

The idle-status register allows checking whether a target module is in idle mode or if an initiator module is in standby mode. The software should not access to a target module in idle mode. A target access, in this state, can lead to an error.

The idle mode of any module can depend on the configuration of the CM_FCLKEN_<domain_name> and CM_ICLKEN_<domain_name> registers, or may be controlled automatically by hardware, depending on the configuration of the CM_AUTOIDLE_<domain_name> registers.

The device has the following idle status registers:

- CM_IDLEST_MPU: MPU subsystem
- CM_IDLEST1_CORE, CM_IDLEST2_CORE, and CM_IDLEST3_CORE: CORE domain peripherals set
- CM_IDLEST_DSS: DSS
- CM_IDLEST_PER: Peripheral domain peripherals set
- CM_IDLEST_NEON: NEON subsystem
- CM_IDLEST_SGX SGX subsystem
- CM_IDLEST_WKUP: WKUP domain peripherals set

- CM_IDLEST_USBHOST: HS USB Host subsystem

4.11.2.4.6 CM_CLKSTCTRL_<domain_name>(Clock State Control Register)

The clock state control register holds a CLKTRCTRL_<clock domain> bit field for each clock domain in the domain. It controls the hardware- and software-supervised state transitions between active and inactive states. Table 4-71 lists the clock state transition settings.

Table 4-71. Clock State Transition Settings

CM_CLKSTCTRL_<pwr domain>. CLKTRCTRL_<clk domain>	Description
0x0	The automatic hardware-supervised mode is disabled. The clocks in a clock domain cannot be cut automatically. This prevents any power transition on the domain.
0x1	Starts software-supervised (forced) sleep transition on the domain. All clocks in the clock domain are automatically cut whenever the initiators in the modules are in standby mode.
0x2	Starts software-supervised (forced) wake-up transition on the domain. The clocks in the clock domain are restarted.
0x3	The automatic hardware-supervised mode is enabled, and the clocks in the clock domain are automatically cut whenever the modules and subsystem in the clock domain are in idle or standby mode and the domain dependencies are met.

The hardware-supervised mode and the software-supervised (forced) sleep mode are mutually exclusive. It is a software decision to program one mode or another, and the software programs the PRCM accordingly.

The hardware-supervised mode is coupled to the sleep and wake-up dependencies programmed in the PRCM, whereas the software-supervised mode must be independent of those dependencies. Therefore, the sleep and wake-up dependencies must be disabled before the software forces a sleep transition on a domain; otherwise, the forced-domain will be wakened immediately because of the wake-up dependency.

The device has the following clock state control registers:

- CM_CLKSTCTRL_MPU: MPU subsystem clock domain
- CM_CLKSTCTRL_CORE: L3 and L4 clock domains
- CM_CLKSTCTRL_DSS: DSS clock domain
- CM_CLKSTCTRL_PER: Peripherals clock domain
- CM_CLKSTCTRL_NEON: NEON clock domain
- CM_CLKSTCTRL_SGX: Graphics clock domain
- CM_CLKSTCTRL_EMU: Emulation clock domain
- CM_CLKSTCTRL_USBHOST: HS USB Host clock domain

The CORE domain has three clock domains (L3 and L4); it does not have forced sleep and forced wake-up ability.

Although the sleep transition in the MPU domain cannot be initiated by software using this register, a software-initiated forced wake-up capability exists. This can be used to wake up the MPU domain if it does not wake up when the CORE domain wakes up (the MPU wake-up-dependency PM_WKDEP_MPU[0] EN_CORE bit is set to 0).

If the hardware-supervised mode is enabled, the following occur:

- The MPU domain clock is automatically cut if the MPU executes the wait-for-interrupt instruction.
- The L3 domain clock is automatically cut when all initiators are in standby mode and slave ports are idled.
- The L4 domain clock is automatically cut when all peripherals and slave ports are idled.
- The DSS interface clock is automatically cut when it stops fetching data from the frame buffer (provided the MPU is also in standby mode, if sleep dependency in the MPU domain is enabled). The display functional clock is controlled only by the CM_FCLKEN_DSS register.
- The PER domain clock is automatically cut when all initiators are in standby mode and all slave ports

are in idle (provided the MPU is standby mode and the CORE domain is INACTIVE, if sleep dependency in their domains is enabled).

- Because of the hardwired sleep dependency between NEON and the MPU domain, NEON can go into idle only if the MPU goes into standby mode. The MPU domain must also be configured in automatic hardware supervised mode for the NEON domain idle transition to occur.
- The USBHOST domain clock is automatically cut whenever the USBHOST is in standby mode (provided MPU is also in standby mode, if sleep dependency in the MPU domain is enabled).

NOTE: Hardware-supervised mode must not be used for the SGX domain when smart-standby is enabled.

4.11.2.4.7 CM_CLKSTST_<domain_name> (Clock State Status Register)

The clock state status register logs the activity status of the domain clock. This includes the activity of the interface clocks running only on the domain.

The device has following clock state status registers:

- CM_CLKSTST_MPU: MPU subsystem clock activity
- CM_CLKSTST_CORE: L3 clock domain activity and L4 clock domain activity
- CM_CLKSTST_DSS: DSS clock activity
- CM_CLKSTST_PER: PER clock domain activity
- CM_CLKSTST_SGX: Graphics subsystem clock activity
- CM_CLKSTST_EMU: Emulation clock activity
- CM_CLKSTST_USBHOST: USBHOST clock activity

4.11.2.4.8 CM_SLEEPDEP_<domain_name> (Sleep Dependency Control Register)

The sleep dependency control register allows the enabling or disabling of the sleep transition dependency of a domain with respect to other domains.

The device has following sleep dependency registers:

- CM_SLEEPDEP_DSS: Display domain sleep dependency with the MPU domain
- CM_SLEEPDEP_PER: PER domain sleep dependencies with the MPU and CORE domains
- CM_SLEEPDEP_SGX: SGX domain sleep dependency with the MPU domain
- CM_SLEEPDEP_USBHOST: USBHOST domain sleep dependency with the MPU domain

Although the CORE domain is composed of L3 and L4 clock domains, the sleep dependency with the L3 clock domain is always enabled (not programmable by software); there is no sleep dependency between the PER domain and the CORE_L4 clock domain.

The dependencies listed in Table 4-72 can be enabled or disabled by software.

Table 4-72. Sleep Dependency Settings

Dependent Domain	Parent Domain	MPU	CORE-L3
DSS		Software controlled	
PER		Software controlled	Always enabled
SGX		Software controlled	
USBHOST		Software controlled	

4.11.2.5 Domain Wake-Up Control Registers

The following modules have programmable wake-up control:

- CORE domain:
  - USBTLL

  - McBSP 1 and 5
  - GPTIMER[10, 11]
  - UART[1,2]
  - I2C[1..3]
  - McSPI[1..4]
  - MMC[1,2, 3]
- PER domain:
  - McBSP[2..4]
  - GPTIMER[2..9]
  - UART 3
  - WDTIMER3
  - GPIO[2..6]
- WKUP domain:
  - GPTIMER[1, 12]
  - GPIO1
  - I/O pad
  - USIM OCP
- DSS domain:
  - DSS subsystem
- USBHOST domain:
  - HS USB Host subsystem

The registers in the following sections control the wake-up settings.

4.11.2.5.1 PM_WKEN_<domain_name> (Wake-Up Enable Register)

The wake-up enable register applies only to modules that can generate wake-up events. It allows the enabling or disabling of the wakeup of the related domain on a module or subsystem wake-up event. Each EN_<module> bit in the register enables/disables the wake-up event from the module to the PRCM.

The device has the following wake-up enable registers:

- PM_WKEN1_CORE and PM_WKEN3_CORE: CORE domain modules wake-up control
- PM_WKEN_DSS: Display subsystem (DSS) wake-up control
- PM_WKEN_PER: Peripheral domain modules wake-up control
- PM_WKEN_WKUP: WKUP domain modules wake-up control
- PM_WKEN_USBHOST: HS USB Host subsystem wake-up control

NOTE:

- The wake-up capability of GPTIMER12 (the secure timer) is not programmable, but is always enabled.
- The wake-up signals issued from the MPU INTC are nonmaskable in the PRCM. However, they cannot be generated if the corresponding interrupts are masked in the INTC.

4.11.2.5.2 PM_WKST_<domain_name> (Wake-Up Status Register)

The wake-up status register logs wake-up events from all modules. Each ST_<module> bit of this register logs a given module-generated wake-up event.

The device has the following wake-up status registers:

- PM_WKST1_CORE and PM_WKST3_CORE: CORE domain modules wake-up event status
- PM_WKST_PER: Peripheral domain modules wake-up event status

- PM_WKST_WKUP: WKUP domain modules wake-up event status
- PM_WKST_USBHOST: HS USB Host subsystem wake-up control

The functional clock of the module causing the wakeup automatically restarts on the wakeup, but the software must enable the functional clock (CM_FCLKEN_<domain_name> register) before clearing the wake-up status bit.

NOTE:

- Software must clear this register before a new sleep transition request; otherwise, the register prevents the sleep transition from occurring.
- If the PRM interrupt is enabled on a module wakeup, the PM_WKST_<domain_name> must be cleared before clearing the PRM_IRQSTATUS_<processor_name> interrupt status register.

4.11.2.5.3 PM_WKDEP_<domain_name> (Wake-Up Dependency Register)

The wake-up dependency register allows the enabling/disabling of a wake-up dependency between domains. When a domain wakes up, it can wake up another domain.

The device has the following wake-up dependency registers:

- PM_WKDEP_MPU: MPU domain wake-up dependency with the following domains:
  - CORE
  - WKUP
  - DSS
  - PER
- PM_WKDEP _DSS: DSS domain wake-up dependency with the following domains:
  - MPU
  - WKUP
- PM_WKDEP _PER: The PER domain has programmable wake-up dependency with the following domains:
  - MPU
  - WKUP
  - CORE

  The PER domain can be wakened only by a wakeup of the L3 clock domain, not by an L4 clock domain.
- PM_WKDEP _NEON: The NEON domain has programmable wake-up dependency with the following domain:
  - MPU
- PM_WKDEP _SGX: The SGX domain has programmable wake-up dependency with the following domains:
  - MPU
  - WKUP
- PM_WKDEP _USBHOST: The USBHOST domain has programmable wake-up dependency with the following domains:
  - MPU
  - WKUP
  - CORE

There is no wake-up dependency control for the WKUP domain, because it is always on.

Because the CORE domain wake-up dependencies are not programmable, the CORE domain always wakes up on a wake-up event on the following domains:

- MPU

- WKUP
- DSS
- SGX
- PER
- USBHOST

4.11.2.5.4 PM_<processor_name>GRPSEL_<domain_name> (Processor Group Selection Register)

The processor group selection register allows defining the group of modules in the domain that can wake up the MPU domain. This bit is effective only if the module wake-up capability is enabled (PM_WKEN_<domain_name> register).

The device has the following processor group selection registers:

- PM_MPUGRPSEL1_CORE and PM_MPUGRPSEL3_CORE: CORE domain modules MPU wake-up control
- PM_MPUGRPSEL_PER: PER domain modules MPU wake-up control
- PM_MPUGRPSEL_WKUP: WKUP domain modules MPU wake-up control
- PM_MPUGRPSEL_USBHOST: HS USB Host subsystem MPU wake-up control

NOTE:

- The wake-up capability of GPTIMER12 (the secure timer) is always attached to the MPU group.

4.11.3 Reset Management Registers

4.11.3.1 Reset Control

4.11.3.1.1 PRM_RSTTIME (Reset Time Register)

The reset time register provides control over reset duration. Two durations are configurable:

- RSTIME1: Minimum duration (by default, two cycles of the 32-kHz clock) of the global warm reset (sys_nreswarm) assertion for external devices, such as flash memories. At power-up reset, DPLLs are reset.
- RSTIME2: Minimum duration (by default, 16 cycles of the RM_ICLK clock) to control domain resets when the domain clocks are on.

For global cold and warm resets, the reset is applied for the RSTIME1 + RSTIME2 duration.

RSTIME1 and RSTIME2 can be reprogrammed for different behavior after the initial power up.

4.11.3.1.2 RM_RSTCTRL_<domain_name> (Reset Control Register)

The reset control register provides control over the local domain software reset.

Most device modules include an individual software reset that can be controlled using the related module SYS_CONFIG register.

The RM_RSTCTRL_<domain_name> register is used for specific subsystems or modules that do not support this local reset or that require a specific system control.

The PRM_RSTCTRL register also handles the global software warm reset control.

The device includes the following reset control registers:

- PRM_RSTCTRL: This register allows control of the assertion of the global software reset and the DPLL3 software reset. These bits are automatically cleared. Assertion of the DPLL3 software reset triggers a device global cold reset. The reset condition of this register depends on the bit field:
  - The RST_GS bit is set on any global source of reset (warm or cold).

– The RST_DPLL3 bit is set only on a global cold source of reset.

4.11.3.1.3 RM_RSTST_<domain_name> (Reset Status Register)

The reset status register logs any source that has generated a reset in the related domain. Depending on the domain, several causes of reset can be logged:

- Global device cold reset
- Global device warm reset
- domain transition (INACTIVE to ACTIVE)
- Software reset
- Processor emulation reset

The PRM_RSTST register logs the source of the global reset:

- External warm reset
- Secure watchdog reset
- MPU watchdog reset
- Security violation reset
- Global software reset and DPLL3 software reset
- Global cold reset

The device includes the following reset status registers:

- RM_RSTST_MPU
- RM_RSTST_CORE
- RM_RSTST_DSS
- RM_RSTST_PER
- RM_RSTST_NEON
- RM_RSTST_SGX
- RM_RSTST_EMU
- PRM_RSTST
- RM_RSTST_USBHOST

Only the RM_RSTST_CORE and PRM_RSTST registers log software sources of reset.

Each bit of these registers is set on the effective release of the respective reset signal.

4.11.4 Power Management Registers

4.11.4.1 PM_PWSTST_<domain_name> (Power State Status Register)

The power state status register provides the status of the power state transition of the domain.

This register may hold the following bit fields:

- POWERSTATESTATUS: Indicates the current power state of the domain (ON or INACTIVE)
- INTRANSITION: Indicates an ongoing power state transition from ON power state to INACTIVE and from INACTIVE to ON power state

The device has the following power state status registers:

- PM_PWSTST_MPU: MPU domain power state status
- PM_PWSTST_CORE: CORE domain power state status
- PM_PWSTST_SGX: SGX domain power state status
- PM_PWSTST_DSS: DSS domain power state status
- PM_PWSTST_PER: PER domain power state status
- PM_PWSTST_NEON: NEON domain power state status
- PM_PWSTST_EMU: EMU domain power state status
- PM_PWSTST_USBHOST: USBHOST domain power state status

The MPU domain power state status register indicates the current domain state.

The CORE domain power state status register indicates the current domain state.

The SGX, DSS, PER, NEON, USBHOST, and EMU domain power state status registers indicate the current domain (ON or INACTIVE) power state or that the domain transition is in progress.

4.11.5 Generic Programming Examples

4.11.5.1 Clock Control

The module clock management is controlled through three programmable steps:

- Enabling or disabling the functional clocks
- Enabling or disabling the interface clocks
- Enabling or disabling the automatic idle mode for the interface clocks

4.11.5.1.1 Enabling and Disabling the Functional Clocks

The flow chart in Figure 4-61 shows how to enable or disable a functional clock.

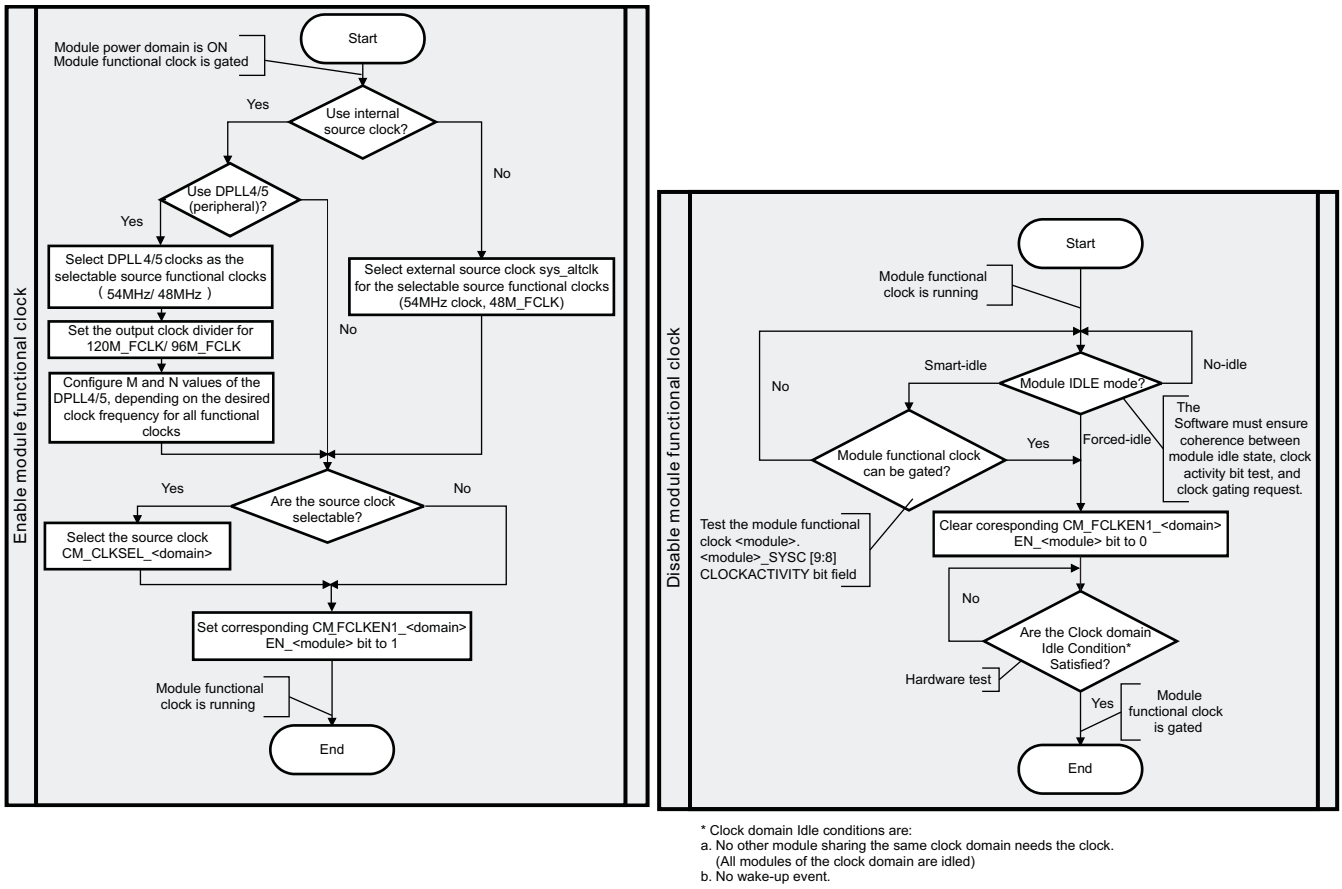
The first step before enabling a functional clock is to select the proper source clock using the corresponding clock selection register (CM_CLKSEL_<domain_name>). It can be either an external source clock or an internal clock; that is, sys_altclk or DPLL4_M3X2_CLK for the DSS_TV_FCLK functional clock of the DSS.

If the source clock is a DPLL3 or DPLL4 output clock, the DPLL multiplier, divider, and output clock ratios are set in the CM_CLKSELn_PLL register, where n is from 1 to 3. The DPLL operating mode is set in the CM_CLKEN_PLL register.

The functional clock is enabled or disabled by writing the dedicated bit in the CM_FCLKEN_<domain_name> register. This bit has a direct effect on the clock activity:

- The functional clock is turned on if the bit is enabled and the clock is not yet active.
- The functional clock is turned off if the bit is disabled and the clock is not required by any other module.

Figure 4-61. Functional Clock Basic Programming Model



* Clock domain Idle conditions are:
a. No other module sharing the same clock domain needs the clock. (All modules of the clock domain are idled)
b. No wake-up event.

prcm-085

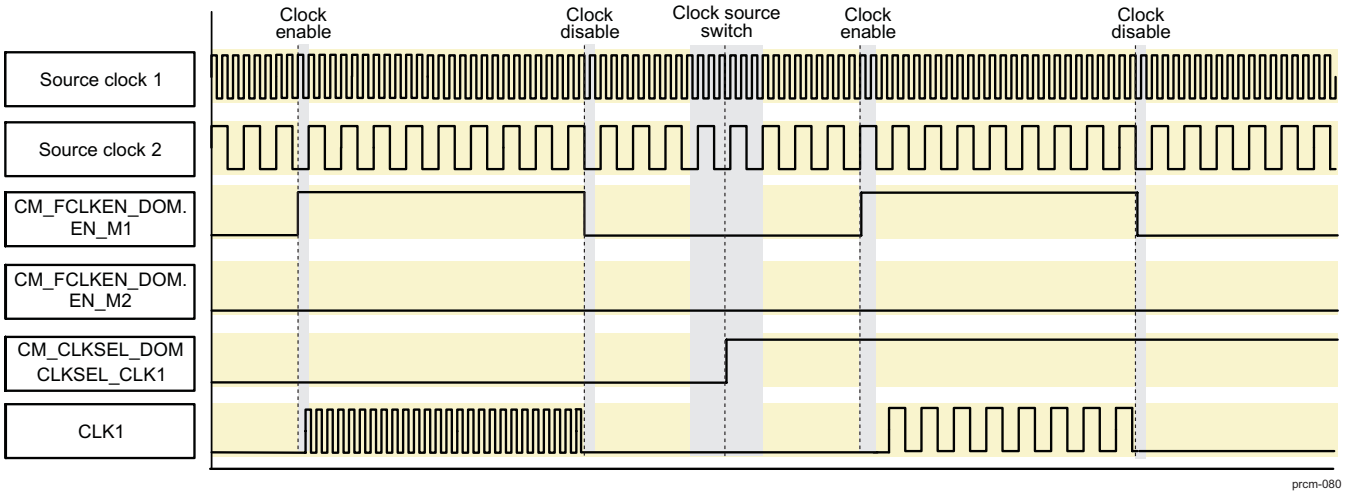
The functional clock must be disabled before switching or scaling its source clock. This means that all the modules using the particular functional clock must not be active during clock switching. To switch a source clock, perform the following sequence:

1. Disable the functional clock by setting the CM_FCLKEN_<domain>EN_<module> bits to 0.
2. Modify the CM_CLKSEL_<domain>CLKSEL_<clock> bits to select the new clock source or clock divider.
3. Enable the functional clock by setting the CM_FCLKEN_<domain>EN_<module> bits to 1.

The timing diagram in Figure 4-62 is a generic example of this sequencing for a functional clock (CLK1). The source clock can be switched between source clock 1 and source clock 2 using the CM_CLKSEL_DOM.CLKSEL_CLK1 bit (source clock 1 is selected when the bit is set to 0; otherwise, source clock 2 is selected). CLK1 can be requested by two modules, M1 and M2. The CM_FCLKEN_DOM.EN_M1 and CM_FCLKEN_DOM.EN_M2 bits control the functional clock enable for the two modules.

NOTE: The activation or deactivation of the clock is implementation-dependent, not one cycle of the source clock, as shown in Figure 4-62.

Figure 4-62. Functional Clock Switching



The following functional clocks require this switching sequence:

- sys_clkout2
- 48M_FCLK (and all gated versions of it)
- 12M_FCLK (and all gated versions of it)
- DSS_TV_CLK
- GPTx_FCLK (with x = 1, 10, and 11)
- GPTx_ALWON_FCLK (with x = 2 up to 9)

4.11.5.1.2 Enabling and Disabling the Interface Clocks

The flow chart in Figure 4-63 shows the enable/disable sequence of the interface clock.

The first step before enabling an interface clock is to select the proper source clock using the corresponding clock selection register (CM_CLKSEL_<domain>). This register allows selection of the interconnect frequency (L3_ICLK, L4_ICLK) from among several divider ratios.

The interface clock is enabled or disabled by writing the dedicated bit in the CM_ICLKEN_<domain> register. This bit has a direct effect on the clock activity:

- The interface clock is turned on if the bit is enabled and the clock is not yet active.
- The interface clock is turned off if the bit is disabled and the clock is not required by any other module.

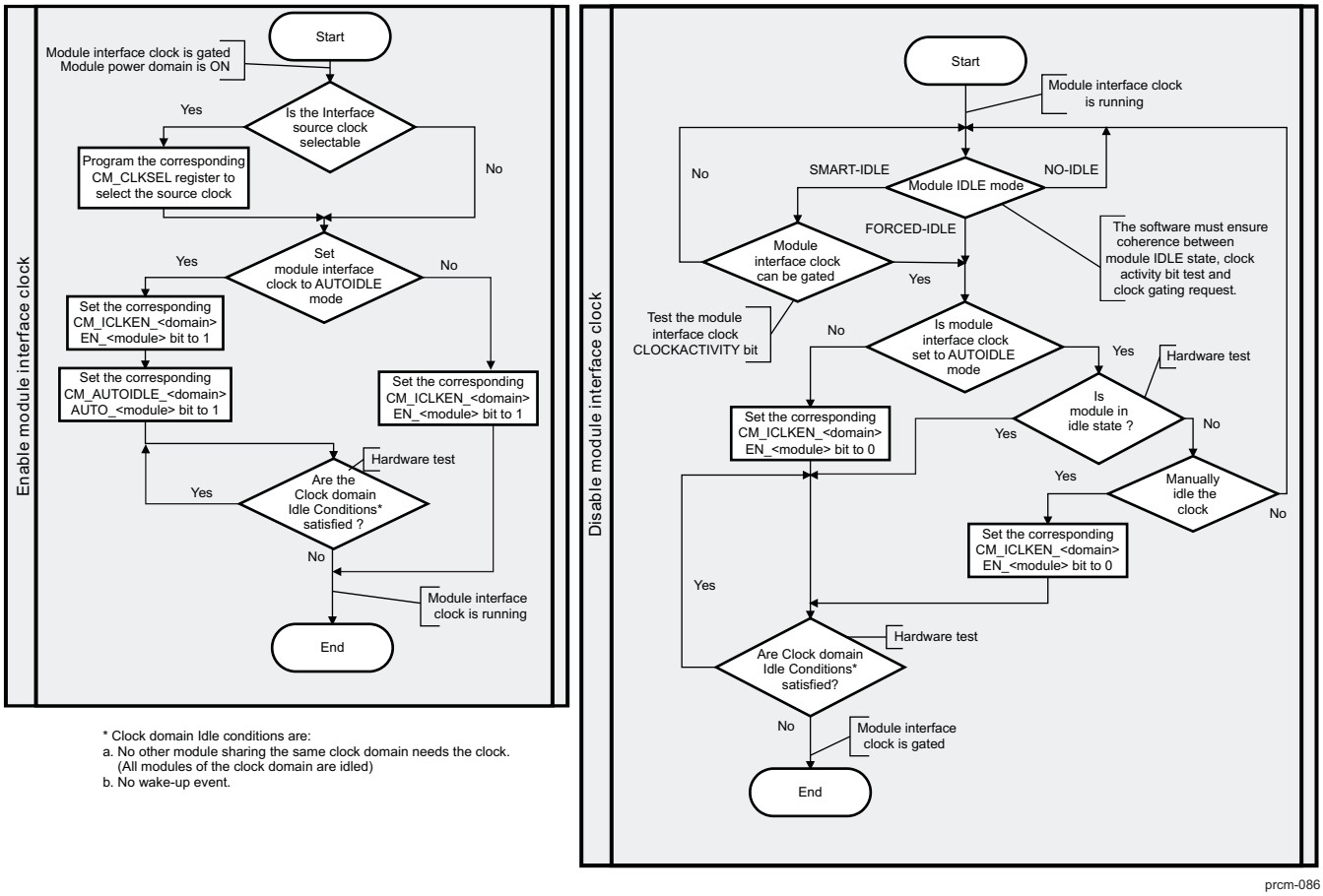
The interface clock can be automatically enabled or disabled by the PRCM, based on hardware conditions. This automatic clock activity control mode is enabled by writing the corresponding bit in the CM_AUTOIDLE_<domain > register. It takes effect only if the interface clock is enabled (the corresponding bit in the CM_ICLKEN_<domain > is set to 1).

The hardware conditions for automatic gating (deactivation) of the clock are as follows:

- The module activity; that is, the module is inactive.
- The domain activity; that is, all modules in the domain are inactive.

The software can read the idle status register CM_IDLESTAT_<domain> at any time to know whether the module is accessible. A module is inaccessible if its idle status bit is set. Accessing an idle module generates an error (if the interface clock is still running) or a time-out (if the interface clock is cut).

Figure 4-63. Interface Clock Basic Programming Model



The frequency ratio between the CORE_CLK, the L3_ICLK, and the L4_ICLK is configured by setting the corresponding CM_CLKSEL_CORE register bit fields. This configuration must be done before switching DPLL3 to lock mode. In this way, the clock ratio is switched while DPLL3 is operating at system clock frequency, and then only DPLL3 is switched to high-frequency locked mode.

If the configuration of the interface clock needs to be changed, first put DPLL3 in bypass mode, select the new configuration, and then relock DPLL3.

NOTE: When performing frequency scaling, the clock division can be done directly by programming the DPLL output divider. In this case, there is no need to change the configuration of the interface clock.

4.11.5.1.3 Enabling and Disabling the INACTIVE State

The flow chart in Figure 4-64 shows how to put a domain into INACTIVE state.

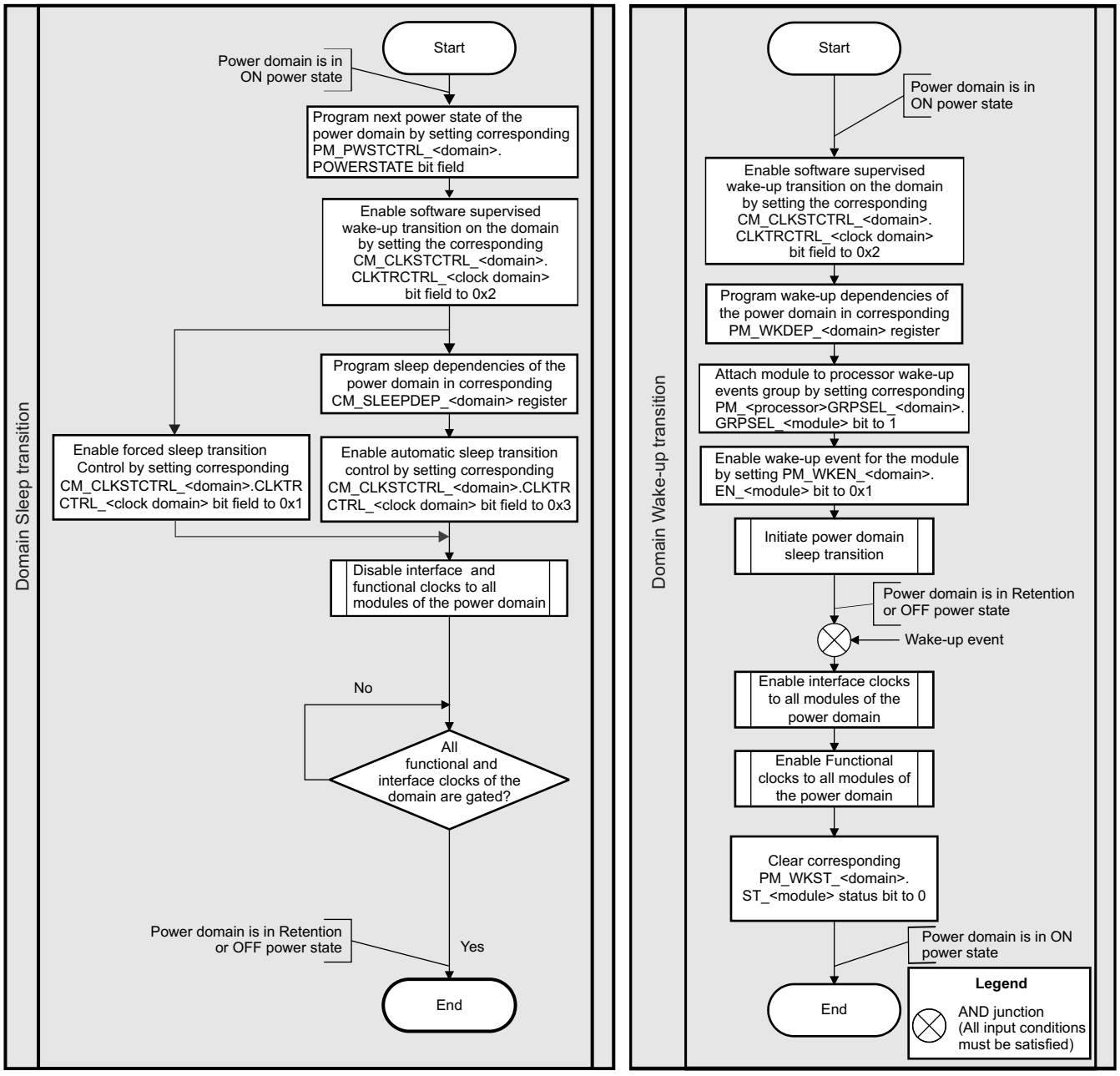
A domain is said to be in INACTIVE state if:

- All the functional and interface clocks of the domain are gated (deactivated).
- All the initiator modules are in standby mode.
- All the dependent domains have reached their mute state. The sleep dependency between the domains is configured by programming the CM_SLEEPDEP_<domain> register.

The domain transition from ACTIVE state to INACTIVE state is effective only if the CM_CLKSTCTRL_<domain> register is programmed for hardware-supervised state transition.

The CM_CLKSTST_<domain> register identifies whether a domain or a clock domain within the domain is accessible. A domain is inaccessible if its corresponding bit in the CM_CLKSTST_<domain > register is set to 1.

Figure 4-64. Domain INACTIVE STATE Basic Programming Model



prcm-087

4.11.5.1.4 Processor Clock Control

The flow chart in Figure 4-65 shows the control sequence of the processor clock.

The processor source clock is generated by the dedicated processor DPLL (DPLL1). After power up, the processor DPLL is in bypass or stop mode. This means the processor clock is either the system clock or is shut off.

The first step is to program the multiplier and divider ratios of the processor DPLL. Those two values are written in the CM_CLKSEL1_PLL_processor> register.

For frequency scaling, the processor DPLL integrates an additional divider to scale down the synthesized clock. The value of this second divider is written in the CM_CLKSEL2_PLL_processor> register. This divider can be configured dynamically (while the processor executes instructions), and the DPLL output clock is scaled without any glitches.

The processor DPLL must be locked at a desired frequency, provided the clock frequency is higher than the system clock. It can then be set to low-power bypass mode when the high-frequency clock is not required. The DPLL operating mode (locked or bypassed) is controlled by programming the CM_CLKEN_PLL_processor> register.

The processor clock can be switched automatically, based on hardware conditions, between the processor DPLL synthesized clock and a bypass clock. This mode is enabled by programming the corresponding mode in the CM_AUTOIDLE_PLL_processor> register. It takes effect only if the processor DPLL is locked (the CM_CLKEN_PLL_processor> register is set in lock mode).

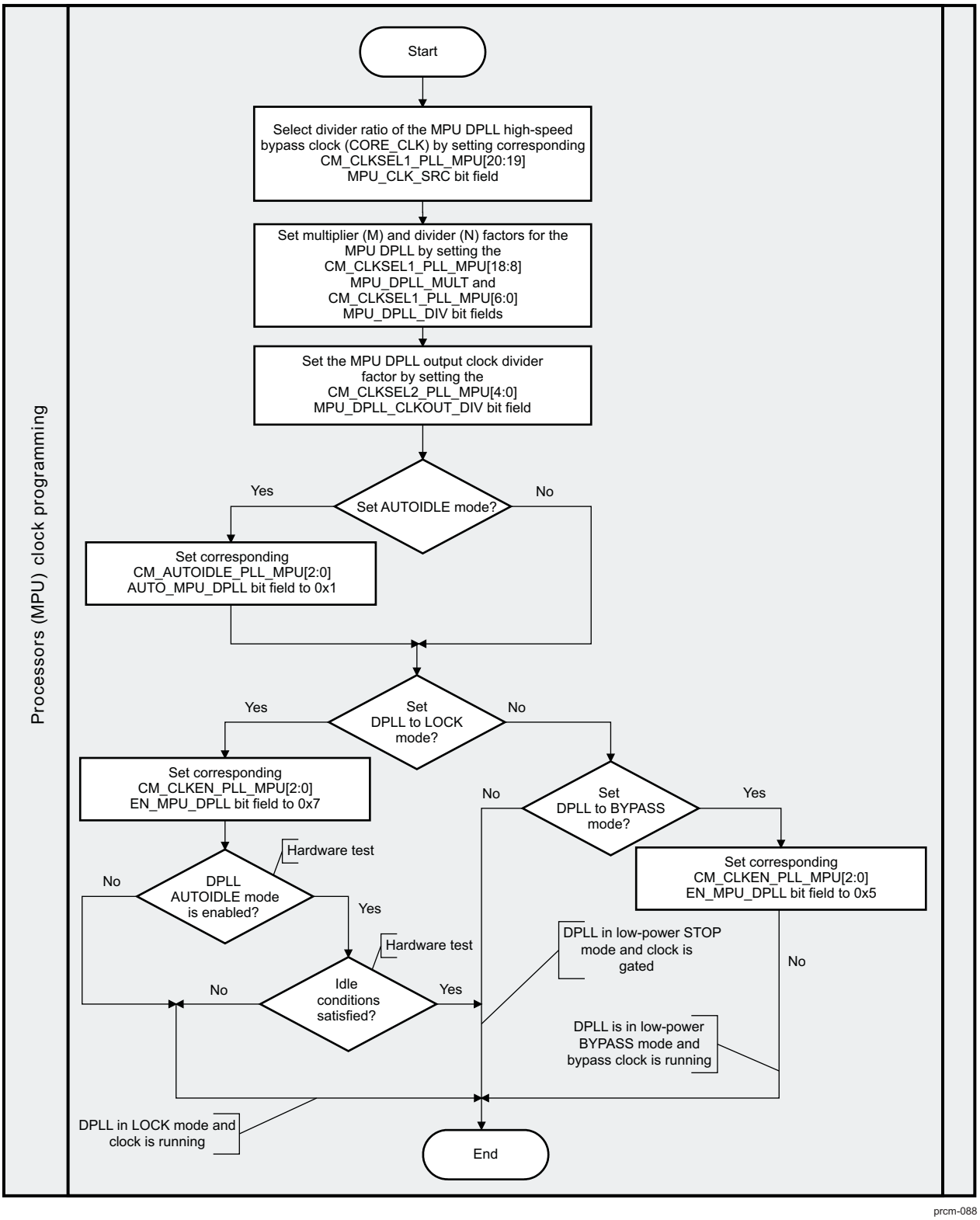
The HS bypass clock for the processor DPLL can be:

- The DPLL3 output clock (CORE_CLK)
- The DPLL3 output clock (CORE_CLK) divided by 2

The HS bypass clock is selected by programming the CM_CLKSEL1_PLL_processor > register.

The software can read the CM_IDLEST_PLL_<processor > register at any time to determine whether the DPLL is in lock mode (DPLL output is a high-frequency synthesized clock) or bypass mode (DPLL output is not the synthesized clock; the DPLL output is the bypass clock, or the DPLL is in transitioning state).

Figure 4-65. Processor Clock Basic Programming Model



4.11.5.2 Reset Management

The reset sequence is hardware-driven. On power on, once all the reset sources have been released, the PRM holds the entire device under reset long enough to ensure the stabilization of the power IC voltages and the oscillator system clock frequency. This reset delay is programmed in the PRM_RSTTIME register.

A domain reset status register (PRM_RSTST_domain>) identifies the source of the current reset applied to the domain. The software must clear this status bit after reset.

A global reset status register (PRM_RSTST) provides information on the global source of resets. All sources of warm reset are logged separately in this register, and all sources of cold reset are logged in a common status bit.

4.11.5.3 Wake-Up Control

The flow chart in Figure 4-66 shows the control sequence of the module wake-up event.

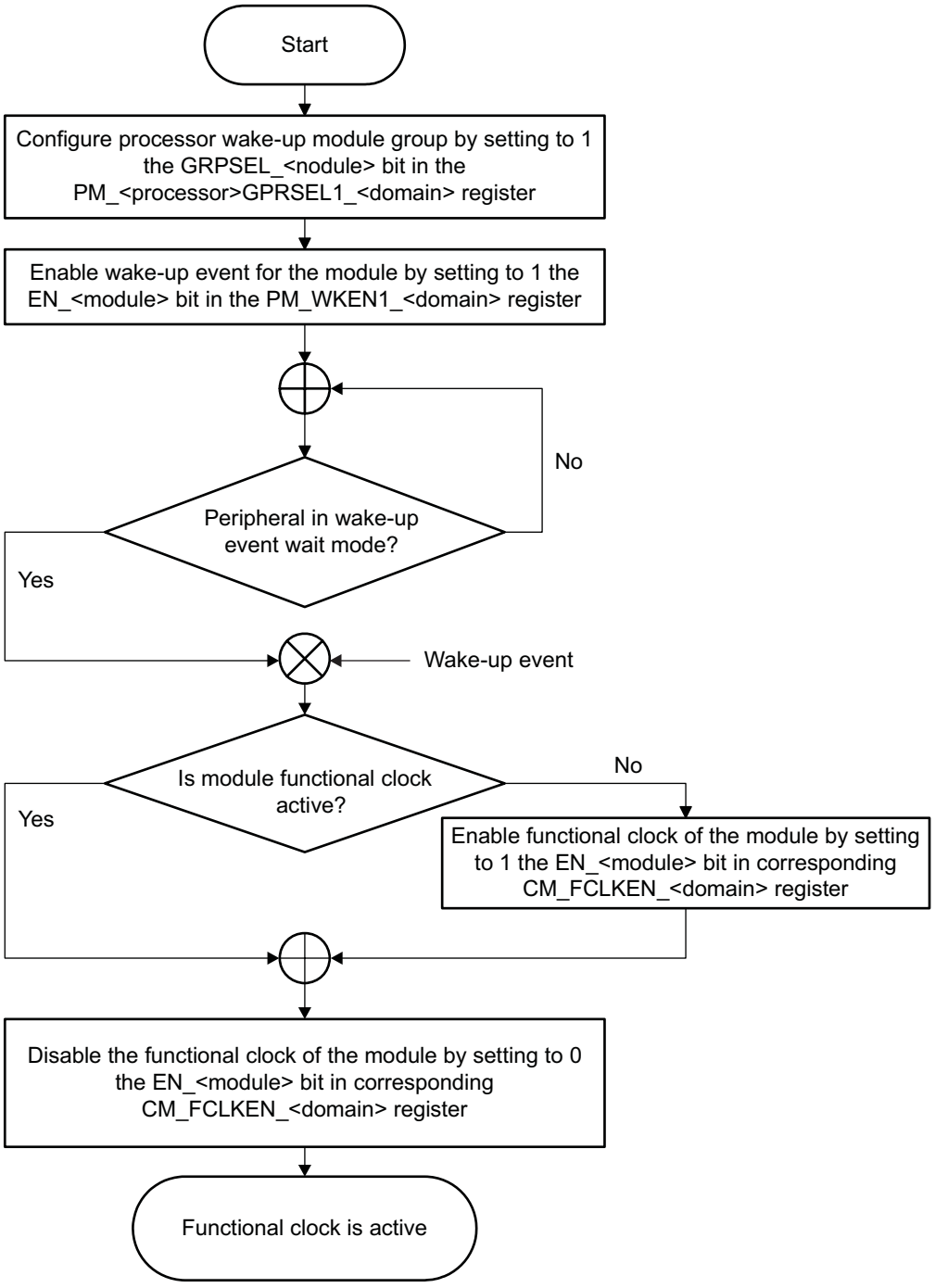
This procedure consists of the following steps:

1. Program the PRCM to consider the wake-up event.
2. Switch to idle mode and wait for the wake-up event.
3. Wake up on the wake-up event and activate the module functional clock.
4. Acknowledge the wake-up event.

The peripheral that can generate a wake-up event must be attached to a group of wake-up event generating modules for one or both processors by programming the PM_<processor>GRPSEL register. Writing 1 to this register allows the corresponding processor to be wakened on a peripheral wake-up event, assuming that the peripheral wake-up capability has been enabled by programming the register PM_WKEN_<domain>.

After this is configured, the PRCM initiates a wake-up procedure on receiving the peripheral wake-up event. The peripheral functional clock must be reenabled by programming the CM_FCLKEN_<domain > register, and then the wake-up event can be acknowledged by clearing the PM_WKST_<domain > register.

Figure 4-66. Wake-Up Basic Programming Model



A domain A can have a functional dependency on a domain B. Thus, when domain A wakes up, it may be necessary to wake up domain B. This wake-up dependency is hardcoded for the CORE domain, but is programmable for the other domains through the PM_WKDEP_<domain> register.

4.11.5.4 Event Generator Programming Examples

The event generator feature allows the MPU domain to be switched between on and idled modes. This is intended to implement an efficient activity modulation of the MPU power state with minimum software support.

When the event generator is activated, the PRCM ensures that the CORE domain always follows the MPU domain activity.

The PRCM.PM_EVGENONTIM_MPU and PRCM.PM_EVGENOFFTIM_MPU registers of the event generator counter allow the configuration of the on and off durations (the number of system clock cycles) for the MPU domain. The PRCM.PM_EVGENCTRL_MPU register allows the enabling/disabling of the event generator feature and control of the way on and off values are loaded in the counter.

There are three ways to load the counter with the next counting value:

- Load on update of the corresponding PRCM.PM_EVGENONTIM_MPU or PRCM.PM_EVGENOFFTIM_MPU register.
- Load the on-time value when the MPU domain wakes up, and the off-time value when the MPU domain starts the sleep transition (the MPU executes the WFI instruction).
- Automatically load the on-time value when the off-time value expires, and automatically load the off-time value when the on-time value expires.

When the counter times out at the end of the on/off time, it triggers the interrupt/wake-up transition, respectively. The software can use the on-time interrupt to trigger the WFI processor instruction to idle the processor clock. Similarly, the wake-up event at the end of the off time restarts the processor clock and interrupts the processor. To enable the corresponding interrupts, the MPU interrupts mask must be set in the PRCM.PRM_IRQENABLE_MPU register.

4.12 PRCM Registers

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. For power savings considerations, ensure that unavailable features and unused modules have clocks properly gated.

Table 4-73 lists the physical addresses of the CM modules. Table 4-74 through Table 4-85 provide register mapping summaries of the CM registers and describe the bits in the individual registers.

Table 4-165 lists the physical addresses of the PRM modules. Table 4-166 through Table 4-170 provide register mapping summaries of the PRM registers and describe the bits in the individual registers.

4.12.1 CM Module Registers

Table 4-73. CM Instance Summary

Module Name	Base Address (hex)	Size	Section
Reserved	0x4800 4000	8192 bytes	
OCP_System_Reg_CM	0x4800 4800	8192 bytes	Section 4.12.1.2
MPU_CM	0x4800 4900	8192 bytes	Section 4.12.1.3
CORE_CM	0x4800 4A00	8192 bytes	Section 4.12.1.4
SGX_CM	0x4800 4B00	8192 bytes	Section 4.12.1.5
WKUP_CM	0x4800 4C00	8192 bytes	Section 4.12.1.6
Clock_Control_Reg_CM	0x4800 4D00	8192 bytes	Section 4.12.1.7
DSS_CM	0x4800 4E00	8192 bytes	Section 4.12.1.8
Reserved	0	8192 bytes	
PER_CM	0x4800 5000	8192 bytes	Section 4.12.1.9
EMU_CM	0x4800 5100	8192 bytes	Section 4.12.1.10
Global_Reg_CM	0x4800 5200	8192 bytes	Section 4.12.1.11
NEON_CM	0x4800 5300	8192 bytes	Section 4.12.1.12
USBHOST_CM	0x4800 5400	8192 bytes	Section 4.12.1.13

4.12.1.1 CM Module Registers Mapping Summary

Table 4-74. OCP_System_Reg_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_REVISION	R	32	0x0000 0000	0x4800 4800	C	Section 4.12.1.2.1
CM_SYSCONFIG	RW	32	0x0000 0010	0x4800 4810	W	Section 4.12.1.2.2

Table 4-75. MPU_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_CLKEN_PLL_MPU	RW	32	0x0000 0004	0x4800 4904	W	Section 4.12.1.3.1
CM_IDLEST_MPU	R	32	0x0000 0020	0x4800 4920	C	Section 4.12.1.3.2
CM_IDLEST_PLL_MPU	R	32	0x0000 0024	0x4800 4924	C	Section 4.12.1.3.3
CM_AUTOIDLE_PLL_MPU	RW	32	0x0000 0034	0x4800 4934	W	Section 4.12.1.3.4
CM_CLKSEL1_PLL_MPU	RW	32	0x0000 0040	0x4800 4940	W	Section 4.12.1.3.5
CM_CLKSEL2_PLL_MPU	RW	32	0x0000 0044	0x4800 4944	W	Section 4.12.1.3.6
CM_CLKSTCTRL_MPU	RW	32	0x0000 0048	0x4800 4948	W	Section 4.12.1.3.7
CM_CLKSTST_MPU	R	32	0x0000 004C	0x4800 494C	C	Section 4.12.1.3.8

Table 4-76. CORE_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_FCLKEN1_CORE	RW	32	0x0000 0000	0x4800 4A00	W	Section 4.12.1.4.1
CM_FCLKEN3_CORE	RW	32	0x0000 0008	0x4800 4A08	W	Section 4.12.1.4.2
CM_ICLKEN1_CORE	RW	32	0x0000 0010	0x4800 4A10	W	Section 4.12.1.4.3
CM_ICLKEN2_CORE	RW	32	0x0000 0014	0x4800 4A14	W	Section 4.12.1.4.4
CM_ICLKEN3_CORE	RW	32	0x0000 0018	0x4800 4A18	W	Section 4.12.1.4.5
CM_IDLEST1_CORE	R	32	0x0000 0020	0x4800 4A20	C	Section 4.12.1.4.6
CM_IDLEST2_CORE	R	32	0x0000 0024	0x4800 4A24	C	Section 4.12.1.4.7
CM_IDLEST3_CORE	R	32	0x0000 0028	0x4800 4A28	C	Section 4.12.1.4.8
CM_AUTOIDLE1_CORE	RW	32	0x0000 0030	0x4800 4A30	W	Section 4.12.1.4.9
CM_AUTOIDLE2_CORE	RW	32	0x0000 0034	0x4800 4A34	W	Section 4.12.1.4.10
CM_AUTOIDLE3_CORE	RW	32	0x0000 0038	0x4800 4A38	W	Section 4.12.1.4.11
CM_CLKSEL_CORE	RW	32	0x0000 0040	0x4800 4A40	W	Section 4.12.1.4.12
CM_CLKSTCTRL_CORE	RW	32	0x0000 0048	0x4800 4A48	W	Section 4.12.1.4.13
CM_CLKSTST_CORE	R	32	0x0000 004C	0x4800 4A4C	C	Section 4.12.1.4.14

Table 4-77. SGX_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_FCLKEN_SGX	RW	32	0x0000 0000	0x4800 4B00	W	Section 4.12.1.5.1
CM_ICLKEN_SGX	RW	32	0x0000 0010	0x4800 4B10	W	Section 4.12.1.5.2
CM_IDLEST_SGX	R	32	0x0000 0020	0x4800 4B20	C	Section 4.12.1.5.3
CM_CLKSEL_SGX	RW	32	0x0000 0040	0x4800 4B40	W	Section 20.2.6.6.12
CM_SLEEPDEP_SGX	RW	32	0x0000 0044	0x4800 4B44	W	Section 4.12.1.5.5
CM_CLKSTCTRL_SGX	RW	32	0x0000 0048	0x4800 4B48	W	Section 4.12.1.5.6
CM_CLKSTST_SGX	R	32	0x0000 004C	0x4800 4B4C	C	Section 4.12.1.5.7

Table 4-78. WKUP_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_FCLKEN_WKUP	RW	32	0x0000 0000	0x4800 4C00	W	Section 4.12.1.6.1
CM_ICLKEN_WKUP	RW	32	0x0000 0010	0x4800 4C10	W	Section 4.12.1.6.2
CM_IDLEST_WKUP	R	32	0x0000 0020	0x4800 4C20	C	Section 4.12.1.6.3
CM_AUTOIDLE_WKUP	RW	32	0x0000 0030	0x4800 4C30	W	Section 4.12.1.6.4
CM_CLKSEL_WKUP	RW	32	0x0000 0040	0x4800 4C40	W	Section 4.12.1.6.5

Table 4-79. Clock_Control_Reg_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_CLKEN_PLL	RW	32	0x0000 0000	0x4800 4D00	W	Section 4.12.1.7.1
CM_CLKEN2_PLL	RW	32	0x0000 0004	0x4800 4D04	W	Section 4.12.1.7.2
CM_IDLEST_CKGEN	R	32	0x0000 0020	0x4800 4D20	C	Section 4.12.1.7.3
CM_IDLEST2_CKGEN	R	32	0x0000 0024	0x4800 4D24	C	Section 4.12.1.7.4

Table 4-79. Clock_Control_Reg_CM Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_AUTOIDLE_PLL	RW	32	0x0000 0030	0x4800 4D30	W	Section 4.12.1.7.5
CM_AUTOIDLE2_PLL	RW	32	0x0000 0034	0x4800 4D34	W	Section 4.12.1.7.6
CM_CLKSEL1_PLL	RW	32	0x0000 0040	0x4800 4D40	W	Section 4.12.1.7.7
CM_CLKSEL2_PLL	RW	32	0x0000 0044	0x4800 4D44	W	Section 4.12.1.7.8
CM_CLKSEL3_PLL	RW	32	0x0000 0048	0x4800 4D48	W	Section 4.12.1.7.9
CM_CLKSEL4_PLL	RW	32	0x0000 004C	0x4800 4D4C	W	Section 4.12.1.7.10
CM_CLKSEL5_PLL	RW	32	0x0000 0050	0x4800 4D50	W	Section 4.12.1.7.11
CM_CLKOUT_CTRL	RW	32	0x0000 0070	0x4800 4D70	C	Section 4.12.1.7.12

Table 4-80. DSS_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_FCLKEN_DSS	RW	32	0x0000 0000	0x4800 4E00	W	Section 4.12.1.8.1
CM_ICLKEN_DSS	RW	32	0x0000 0010	0x4800 4E10	W	Section 4.12.1.8.2
CM_IDLEST_DSS	R	32	0x0000 0020	0x4800 4E20	C	Section 4.12.1.8.3
CM_AUTOIDLE_DSS	RW	32	0x0000 0030	0x4800 4E30	W	Section 4.12.1.8.4
CM_CLKSEL_DSS	RW	32	0x0000 0040	0x4800 4E40	W	Section 4.12.1.8.5
CM_SLEEPDEP_DSS	RW	32	0x0000 0044	0x4800 4E44	W	Section 4.12.1.8.6
CM_CLKSTCTRL_DSS	RW	32	0x0000 0048	0x4800 4E48	W	Section 4.12.1.8.7
CM_CLKSTST_DSS	R	32	0x0000 004C	0x4800 4E4C	C	Section 4.12.1.8.8

Table 4-81. PER_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_FCLKEN_PER	RW	32	0x0000 0000	0x4800 5000	W	Section 4.12.1.9.1
CM_ICLKEN_PER	RW	32	0x0000 0010	0x4800 5010	W	Section 4.12.1.9.2
CM_IDLEST_PER	R	32	0x0000 0020	0x4800 5020	C	Section 4.12.1.9.3
CM_AUTOIDLE_PER	RW	32	0x0000 0030	0x4800 5030	W	Section 4.12.1.9.4
CM_CLKSEL_PER	RW	32	0x0000 0040	0x4800 5040	W	Section 4.12.1.9.5
CM_SLEEPDEP_PER	RW	32	0x0000 0044	0x4800 5044	W	Section 4.12.1.9.6
CM_CLKSTCTRL_PER	RW	32	0x0000 0048	0x4800 5048	W	Section 4.12.1.9.7
CM_CLKSTST_PER	R	32	0x0000 004C	0x4800 504C	C	Section 4.12.1.9.8

Table 4-82. EMU_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_CLKSEL1_EMU	RW	32	0x0000 0040	0x4800 5140	W	Section 4.12.1.10.1
CM_CLKSTCTRL_EMU	RW	32	0x0000 0048	0x4800 5148	C	Section 4.12.1.10.2
CM_CLKSTST_EMU	R	32	0x0000 004C	0x4800 514C	C	Section 4.12.1.10.3
CM_CLKSEL2_EMU	RW	32	0x0000 0050	0x4800 5150	W	Section 4.12.1.10.4
CM_CLKSEL3_EMU	RW	32	0x0000 0054	0x4800 5154	W	Section 4.12.1.10.5

Table 4-83. Global_Reg_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_POLCTRL	RW	32	0x0000 009C	0x4800 529C	C	Section 4.12.1.11.1

Table 4-84. NEON_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_IDLEST_NEON	R	32	0x0000 0020	0x4800 5320	C	Section 4.12.1.12.1
CM_CLKSTCTRL_NEON	RW	32	0x0000 0048	0x4800 5348	W	Section 4.12.1.12.2

Table 4-85. USBHOST_CM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
CM_FCLKEN_USBHOST	RW	32	0x0000 0000	0x4800 5400	W	Section 4.12.1.13.1
CM_ICLKEN_USBHOST	RW	32	0x0000 0010	0x4800 5410	W	Section 4.12.1.13.2
CM_IDLEST_USBHOST	R	32	0x0000 0020	0x4800 5420	C	Section 4.12.1.13.3
CM_AUTOIDLE_USBHOST	RW	32	0x0000 0030	0x4800 5430	W	Section 4.12.1.13.4
CM_SLEEPDEP_USBHOST	RW	32	0x0000 0044	0x4800 5444	W	Section 4.12.1.13.5
CM_CLKSTCTRL_USBHOST	RW	32	0x0000 0048	0x4800 5448	W	Section 4.12.1.13.6
CM_CLKSTST_USBHOST	R	32	0x0000 004C	0x4800 544C	C	Section 4.12.1.13.7

4.12.1.2 OCP_System_Reg_CM Register Descriptions

4.12.1.2.1 CM_REVISION

Table 4-86. CM_REVISION

Address Offset	0x0000 0000		
Physical Address	0x4800 4800	**Instance**	OCP_System_Reg_CM
Description	This register contains the IP revision code for the CM part of the PRCM		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	0x10

4.12.1.2.2 CM_SYSCONFIG

Table 4-87. CM_SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x4800 4810	**Instance**	OCP_System_Reg_CM
Description	This register controls the various parameters of the interface clock		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	AUTOIDLE	Internal clock gating strategy (for the CM part of the PRCM) 0x0: Interface clock is free-running 0x1: Automatic clock gating strategy is enabled, based on the interface activity.	RW	0x1

4.12.1.3 MPU_CM Register Descriptions

4.12.1.3.1 CM_CLKEN_PLL_MPU

Table 4-88. CM_CLKEN_PLL_MPU

Address Offset	0x0000 0004		
Physical Address	0x4800 4904	**Instance**	MPU_CM
Description	This register allows controlling the DPLL1 modes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					EN_MPU_DPLL_LPMODE	MPU_DPLL_RAMPTIME		MPU_DPLL_FREQSEL				EN_MPU_DPLL_DRIFTGUARD	EN_MPU_DPLL		

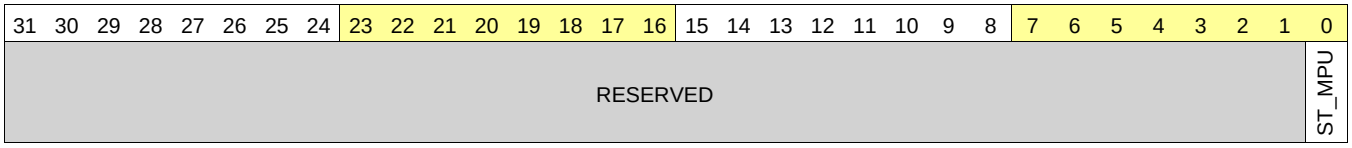
Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Reserved: keep at reset value.	R	0x000000
10	EN_MPU_DPLL_LPMODE	This bit allows to enable or disable the LP mode of the MPU DPLL. Writing this bit to switch the mode between LP or normal mode will take effect only when the DPLL will have transition into the bypass or stop state, followed by a lock or re-lock of the DPLL. 0x0: Disables the DPLL LP mode to re-enter the normal mode at the following lock or re-lock sequence. 0x1: Enables the DPLL LP mode to enter the LP mode at the following lock or re-lock sequence.	RW	0x0
9:8	MPU_DPLL_RAMPTIME	The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and relock. The frequency ramping occurs in a maximum of four steps in frequency before the DPLL frequency lock indicator is asserted. The ramp step size is specified in the number of DPLL internal reference clock cycles. The internal DPLL reference clock cycle is given by: Fint = DPLL input reference clock frequency/(N + 1), where N is the value in the CM_CLKSEL1_PLL_MPU[6:0] MPU_DPLL_DIV bit field.This register is used to enable and disable the DPLL ramping feature. If enabled, it is also used to select a range of ramp step sizes. The step size used per range depends on the DPLL internal reference clock frequency: the lower the frequency, the smaller the ramp step size. 0x0: Disables the frequency ramping feature. 0x1: Ramp step size range is 2–40 Fint cycles. 0x2: Ramp step size range is 4–80 Fint cycles. 0x3: Ramp step size range is 12–240 Fint cycles.	RW	0x0

Bits	Field Name	Description	Type	Reset
7:4	MPU_DPLL_FREQSEL	This bit field allows selecting the proper range of the DPLL1 internal frequency depending on the DPLL reference clock and the N divider. 0x3: 0.75 MHz—1.0 MHz 0x4: 1.0 MHz—1.25 MHz 0x5: 1.25 MHz—1.5 MHz 0x6: 1.5 MHz—1.75 MHz 0x7: 1.75 MHz—2.1 MHz 0xB: 7.5 MHz—10 MHz 0xC: 10 MHz—12.5 MHz 0xD: 12.5 MHz—15 MHz 0xE: 15 MHz—17.5 MHz 0xF: 17.5 MHz—21 MHz	RW	0x1
3	EN_MPU_DPLL_DRIFTGUARD	This bit allows to enable or disable the automatic recalibration feature of the MPU DPLL. The DPLL1 will automatically start a recalibration process upon assertion of the recal flag if this bit is set. 0x0: Disables the DPLL1 automatic recalibration mode 0x1: Enables the DPLL1 automatic recalibration mode	RW	0x0
2:0	EN_MPU_DPLL	DPLL1 control; Other enums: Reserved 0x5: Put the DPLL1 in low power bypass mode 0x7: Enables the DPLL1 in lock mode	RW	0x5

4.12.1.3.2 CM_IDLEST_MPU

Table 4-89. CM_IDLEST_MPU

Address Offset	0x0000 0020		
Physical Address	0x4800 4920	**Instance**	MPU_CM
Description	Modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

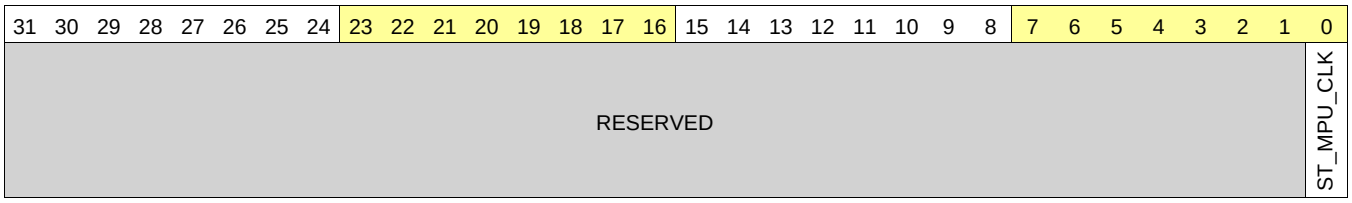


Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	ST_MPU	MPU standby status. 0x0: MPU is active. 0x1: MPU is in standby mode.	R	0x1

4.12.1.3.3 CM_IDLEST_PLL_MPU

Table 4-90. CM_IDLEST_PLL_MPU

Address Offset	0x0000 0024		
Physical Address	0x4800 4924	**Instance**	MPU_CM
Description	This register allows monitoring the master clock activity. This register is read only and automatically updated.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	ST_MPU_CLK	MPU_CLK activity 0x0: DPLL1 is bypassed 0x1: DPLL1 is locked	R	0x0

4.12.1.3.4 CM_AUTOIDLE_PLL_MPU

Table 4-91. CM_AUTOIDLE_PLL_MPU

Address Offset	0x0000 0034		
Physical Address	0x4800 4934	**Instance**	MPU_CM
Description	This register provides automatic control over the DPLL1 activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														AUTO_MPU_DPLL	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2:0	AUTO_MPU_DPLL	DPLL1 automatic control; Other enums: Reserved	RW	0x0
		0x0: Auto control disabled		
		0x1: DPLL1 is automatically put in low power stop mode when the MPU clock is not required anymore. It is also restarted automatically.		

4.12.1.3.5 CM_CLKSEL1_PLL_MPU

Table 4-92. CM_CLKSEL1_PLL_MPU

Address Offset	0x0000 0040		
Physical Address	0x4800 4940	**Instance**	MPU_CM
Description	This register provides controls over the MPU DPLL.		
Type	RW		

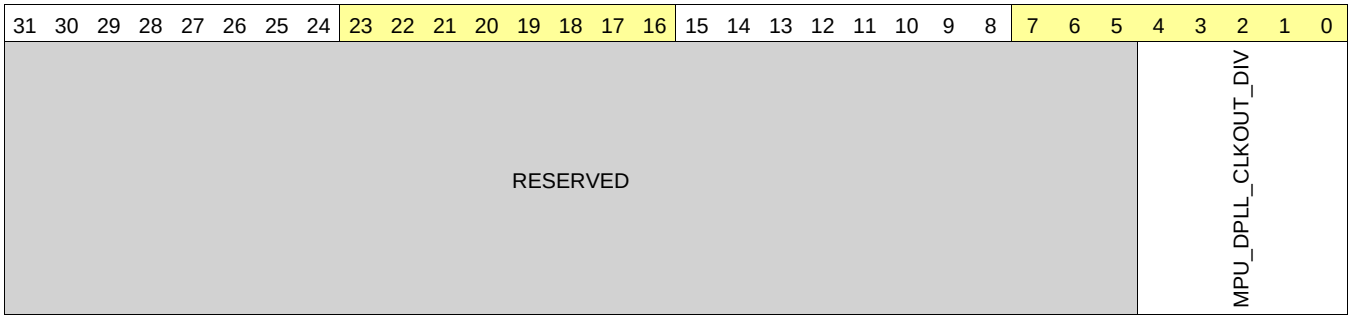
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										MPU_CLK_SRC			MPU_DPLL_MULT											RESERVED	MPU_DPLL_DIV						

Bits	Field Name	Description	Type	Reset
31:22	RESERVED	Reserved: keep at reset value.	R	0x000
21:19	MPU_CLK_SRC	Selects the DPLL1 bypass source clock; Other enums: Reserved 0x1: DPLL1_FCLK is CORE_CLK divided by 1 0x2: DPLL1_FCLK is CORE_CLK divided by 2 0x4: DPLL1_FCLK is CORE_CLK divided by 4	RW	0x1
18:8	MPU_DPLL_MULT	DPLL1 multiplier factor (0 to 2047)	RW	0x000
7	RESERVED	Reserved: keep at reset value.	R	0x0
6:0	MPU_DPLL_DIV	DPLL1 divider factor (0 to 127)	RW	0x00

4.12.1.3.6 CM_CLKSEL2_PLL_MPU

Table 4-93. CM_CLKSEL2_PLL_MPU

Address Offset	0x0000 0044		
Physical Address	0x4800 4944	**Instance**	MPU_CM
Description	This register provides controls over the MPU DPLL.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4:0	MPU_DPLL_CLKOUT_DIV	DPLL1 output clock divider factor (1 up to 16); Other enums: Reserved 0x1: DPLL1 CLKOUTX2 divided by 1 0x2: DPLL1 CLKOUTX2 divided by 2 0x3: DPLL1 CLKOUTX2 divided by 3 0x4: DPLL1 CLKOUTX2 divided by 4 0x5: DPLL1 CLKOUTX2 divided by 5 0x6: DPLL1 CLKOUTX2 divided by 6 0x7: DPLL1 CLKOUTX2 divided by 7 0x8: DPLL1 CLKOUTX2 divided by 8 0x9: DPLL1 CLKOUTX2 divided by 9 0xA: DPLL1 CLKOUTX2 divided by 10 0xB: DPLL1 CLKOUTX2 divided by 11 0xC: DPLL1 CLKOUTX2 divided by 12 0xD: DPLL1 CLKOUTX2 divided by 13 0xE: DPLL1 CLKOUTX2 divided by 14 0xF: DPLL1 CLKOUTX2 divided by 15 0x10: DPLL1 CLKOUTX2 divided by 16	RW	0x01

4.12.1.3.7 CM_CLKSTCTRL_MPU

Table 4-94. CM_CLKSTCTRL_MPU

Address Offset	0x0000 0048		
Physical Address	0x4800 4948	**Instance**	MPU_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		

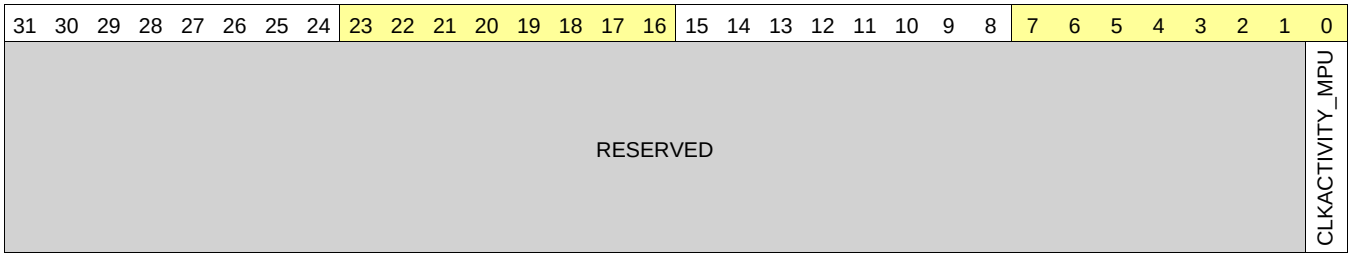
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														CLKTRCTRL_MPU	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_MPU	Controls the clock state transition of the MPU clock domain. 0x0: Automatic transition is disabled 0x1: Reserved 0x2: Start a software supervised wake-up transition on the domain 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0

4.12.1.3.8 CM_CLKSTST_MPU

Table 4-95. CM_CLKSTST_MPU

Address Offset	0x0000 004C		
Physical Address	0x4800 494C	**Instance**	MPU_CM
Description	This register provides a status on the clock activity in the domain (MPU DPLL output clock).		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	CLKACTIVITY_MPU	Clock activity status 0x0: No domain clock activity 0x1: Domain clock is active	R	0x0

4.12.1.4 CORE_CM Register Descriptions

4.12.1.4.1 CM_FCLKEN1_CORE

Table 4-96. CM_FCLKEN1_CORE

Address Offset	0x0000 0000		
Physical Address	0x4800 4A00	**Instance**	CORE_CM
Description	Controls the module functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	EN_MMC3	RESERVED				EN_MMC2	EN_MMC1	RESERVED	EN_HDQ	EN_MCSPI4	EN_MCSPI3	EN_MCSPI2	EN_MCSPI1	EN_I2C3	EN_I2C2	EN_I2C1	EN_UART2	EN_UART1	EN_GPT11	EN_GPT10	EN_MCBSP5	EN_MCBSP1	RESERVED								

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved: keep at reset value.	RW	0x0
30	EN_MMC3	MMC3 functional clock control. 0x0: MMC3 functional clock is disabled 0x1: MMC3 functional clock is enabled	RW	0x0
29:26	RESERVED	Reserved: keep at reset value.	R	0x0
25	EN_MMC2	MMC2 functional clock control. 0x0: MMC2 functional clock is disabled 0x1: MMC2 functional clock is enabled	RW	0x0
24	EN_MMC1	MMC1 functional clock control. 0x0: MMC 1 functional clock is disabled 0x1: MMC 1 functional clock is enabled	RW	0x0
23	RESERVED	Reserved: keep at reset value.	R	0x0
22	EN_HDQ	HDQ-1 wire functional clock control. 0x0: HDQ functional clock is disabled 0x1: HDQ functional clock is enabled	RW	0x0
21	EN_MCSPI4	McSPI 4 functional clock control. 0x0: McSPI 4 functional clock is disabled 0x1: McSPI 4 functional clock is enabled	RW	0x0
20	EN_MCSPI3	McSPI 3 functional clock control. 0x0: McSPI 3 functional clock is disabled 0x1: McSPI 3 functional clock is enabled	RW	0x0
19	EN_MCSPI2	McSPI 2 functional clock control. 0x0: McSPI 2 functional clock is disabled 0x1: McSPI 2 functional clock is enabled	RW	0x0
18	EN_MCSPI1	McSPI 1 functional clock control. 0x0: McSPI 1 functional clock is disabled 0x1: McSPI 1 functional clock is enabled	RW	0x0
17	EN_I2C3	I2C 3 functional clock control. 0x0: I2C 3 functional clock is disabled 0x1: I2C 3 functional clock is enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
16	EN_I2C2	I2C 2 functional clock control. 0x0: I2C 2 functional clock is disabled 0x1: I2C 2 functional clock is enabled	RW	0x0
15	EN_I2C1	I2C 1 functional clock control. 0x0: I2C 1 functional clock is disabled 0x1: I2C 1 functional clock is enabled	RW	0x0
14	EN_UART2	UART 2 functional clock control. 0x0: UART 2 functional clock is disabled 0x1: UART 2 functional clock is enabled	RW	0x0
13	EN_UART1	UART 1 functional clock control. 0x0: UART 1 functional clock is disabled 0x1: UART 1 functional clock is enabled	RW	0x0
12	EN_GPT11	GPTIMER 11 functional clock control. 0x0: GPTIMER 11 functional clock is disabled 0x1: GPTIMER 11 functional clock is enabled	RW	0x0
11	EN_GPT10	GPTIMER 10 functional clock control. 0x0: GPTIMER 10 functional clock is disabled 0x1: GPTIMER 10 functional clock is enabled	RW	0x0
10	EN_MCBSP5	McBSP 5 functional clock control. 0x0: McBSP 5 functional clock is disabled 0x1: McBSP 5 functional clock is enabled	RW	0x0
9	EN_MCBSP1	McBSP 1 functional clock control. 0x0: McBSP 1 functional clock is disabled 0x1: McBSP 1 functional clock is enabled	RW	0x0
8:0	RESERVED	Reserved: keep at reset value.	R	0x000

4.12.1.4.2 CM_FCLKEN3_CORE

Table 4-97. CM_FCLKEN3_CORE

Address Offset	0x0000 0008		
Physical Address	0x4800 4A08	**Instance**	CORE_CM
Description	Controls the module functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													EN_USBTLL	EN_TS	EN_CPEFUSE

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	EN_USBTLL	USB TLL functional clock control. 0x0: USB TLL functional clock is disabled 0x1: USB TLL functional clock is enabled	RW	0x0
1	EN_TS	Temperature Sensors functional clock control. 0x0: Temperature Sensors functional clock is disabled (for both BandGap) 0x1: Temperature Sensors functional clock is enabled (for both BandGap)	RW	0x0
0	EN_CPEFUSE	CPEFUSE functional clock control. 0x0: CPEFUSE functional clock is disabled 0x1: CPEFUSE functional clock is enabled	RW	0x0

4.12.1.4.3 CM_ICLKEN1_CORE

Table 4-98. CM_ICLKEN1_CORE

Address Offset	0x0000 0010		
Physical Address	0x4800 4A10	**Instance**	CORE_CM
Description	Controls the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	EN_MMC3	RESERVED				EN_MMC2	EN_MMC1	EN_UART4	EN_HDQ	EN_MCSPI4	EN_MCSPI3	EN_MCSPI2	EN_MCSPI1	EN_I2C3	EN_I2C2	EN_I2C1	EN_UART2	EN_UART1	EN_GPT11	EN_GPT10	EN_MCBSP5	EN_MCBSP1	RESERVED	EN_SCMCTRL	RESERVED	EN_IPSS	RESERVED		EN_SDRC	RESERVED

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved: keep at reset value.	R	0x0
30	EN_MMC3	MMC SDIO 3 interface clock control. 0x0: MMC 3 interface clock is disabled 0x1: MMC 3 interface clock is enabled	RW	0x0
29:26	RESERVED	Reserved: keep at reset value.	R	0x0
25	EN_MMC2	MMC SDIO 2 interface clock control. 0x0: MMC 2 interface clock is disabled 0x1: MMC 2 interface clock is enabled	RW	0x0
24	EN_MMC1	MMC SDIO 1 interface clock control. 0x0: MMC 1 interface clock is disabled 0x1: MMC 1 interface clock is enabled	RW	0x0
23	EN_UART4	UART4 interface clock control. 0x0: UART4 interface clock is disabled 0x1: UART4 interface clock is enabled	RW	0x0
22	EN_HDQ	HDQ-wire interface clock control. 0x0: HDQ interface clock is disabled 0x1: HDQ interface clock is enabled	RW	0x0
21	EN_MCSPI4	McSPI 4 interface clock control. 0x0: McSPI 4 interface clock is disabled 0x1: McSPI 4 interface clock is enabled	RW	0x0
20	EN_MCSPI3	McSPI 3 interface clock control. 0x0: McSPI 3 interface clock is disabled 0x1: McSPI 3 interface clock is enabled	RW	0x0
19	EN_MCSPI2	McSPI 2 interface clock control. 0x0: McSPI 2 interface clock is disabled 0x1: McSPI 2 interface clock is enabled	RW	0x0
18	EN_MCSPI1	McSPI 1 interface clock control. 0x0: McSPI 1 interface clock is disabled 0x1: McSPI 1 interface clock is enabled	RW	0x0
17	EN_I2C3	I2C 3 interface clock control. 0x0: I2C 3 interface clock is disabled 0x1: I2C 3 interface clock is enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
16	EN_I2C2	I2C 2 interface clock control. 0x0: I2C 2 interface clock is disabled 0x1: I2C 2 interface clock is enabled	RW	0x0
15	EN_I2C1	I2C 1 interface clock control. 0x0: I2C 1 interface clock is disabled 0x1: I2C 1 interface clock is enabled	RW	0x0
14	EN_UART2	UART 2 interface clock control. 0x0: UART 2 interface clock is disabled 0x1: UART 2 interface clock is enabled	RW	0x0
13	EN_UART1	UART 1 interface clock control. 0x0: UART 1 interface clock is disabled 0x1: UART 1 interface clock is enabled	RW	0x0
12	EN_GPT11	GPTIMER 11 interface clock control. 0x0: GPTIMER 11 interface clock is disabled 0x1: GPTIMER 11 interface clock is enabled	RW	0x0
11	EN_GPT10	GPTIMER 10 interface clock control. 0x0: GPTIMER 10 interface clock is disabled 0x1: GPTIMER 10 interface clock is enabled	RW	0x0
10	EN_MCBSP5	McBSP 5 interface clock control. 0x0: McBSP 5 interface clock is disabled 0x1: McBSP 5 interface clock is enabled	RW	0x0
9	EN_MCBSP1	McBSP 1 interface clock control. 0x0: McBSP 1 interface clock is disabled 0x1: McBSP 1 interface clock is enabled	RW	0x0
8:7	RESERVED	Reserved: keep at reset value.	R	0x0
6	EN_SCMCTRL	System Control Module interface clock control 0x0: System Control Module interface clock is disabled 0x1: S.M. interface clock is enabled	RW	0x1
5	RESERVED	Reserved: keep at reset value.	R	0x0
4	EN_IPSS	IPSS interface clock control. 0x0: IPSS interface clock is disabled 0x1: IPSS interface clock is enabled	RW	0x0
3:2	RESERVED	Reserved: keep at reset value.	R	0x2
1	EN_SDRC	SDRC interface clock control. 0x0: SDRC interface clock is disabled 0x1: SDRC interface clock is enabled	RW	0x1
0	RESERVED	Reserved: keep at reset value.	RW	0x0

4.12.1.4.4 CM_ICLKEN2_CORE

Table 4-99. CM_ICLKEN2_CORE

Address Offset	0x0000 0014		
Physical Address	0x4800 4A14	**Instance**	CORE_CM
Description	This register is Reserved for future use. Configuring this register would result in undefined behavior.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															

Bits	Field Name	Description	Type	Reset
31:0	RESERVED	Reserved: keep at reset value.	R	0x00000000

4.12.1.4.5 CM_ICLKEN3_CORE

Table 4-100. CM_ICLKEN3_CORE

Address Offset	0x0000 0018		
Physical Address	0x4800 4A18	**Instance**	CORE_CM
Description	Controls the module interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													EN_USBTLL	RESERVED	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	EN_USBTLL	USB TLL interface clock control. 0x0: USB TLL interface clock is disabled 0x1: USB TLL interface clock is enabled	RW	0x0
1:0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.1.4.6 CM_IDLEST1_CORE

Table 4-101. CM_IDLEST1_CORE

Address Offset	0x0000 0020		
Physical Address	0x4800 4A20	**Instance**	CORE_CM
Description	CORE modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	ST_MMC3	RESERVED				ST_MMC2	ST_MMC1	ST_UART4	ST_HDQ	ST_MCSPI4	ST_MCSPI3	ST_MCSPI2	ST_MCSPI1	ST_I2C3	ST_I2C2	ST_I2C1	ST_UART2	ST_UART1	ST_GPT11	ST_GPT10	ST_MCBSP5	ST_MCBSP1	RESERVED	ST_SCMCTRL	ST_IPSS_IDLE	RESERVED			ST_SDMA	ST_EMIF4	RESERVED

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved	R	0x1
30	ST_MMC3	MMC 3 idle status. 0x0: MMC 3 can be accessed. 0x1: MMC 3 cannot be accessed. Any access may return an error.	R	0x1
29:26	RESERVED	Reserved	R	0xF
25	ST_MMC2	MMC 2 idle status. 0x0: MMC 2 can be accessed. 0x1: MMC 2 cannot be accessed. Any access may return an error.	R	0x1
24	ST_MMC1	MMC SDIO 1 idle status. 0x0: MMC 1 can be accessed. 0x1: MMC 1 cannot be accessed. Any access may return an error.	R	0x1
23	ST_UART4	UART4 idle status. 0x0: UART4 can be accessed. 0x1: UART4 cannot be accessed. Any access may return an error.	R	0x1
22	ST_HDQ	HDQ-1 wire idle status. 0x0: HDQ can be accessed. 0x1: HDQ cannot be accessed. Any access may return an error.	R	0x1
21	ST_MCSPI4	McSPI 4 idle status. 0x0: McSPI 4 can be accessed. 0x1: McSPI 4 cannot be accessed. Any access may return an error.	R	0x1
20	ST_MCSPI3	McSPI 3 idle status. 0x0: McSPI 3 can be accessed. 0x1: McSPI 3 cannot be accessed. Any access may return an error.	R	0x1
19	ST_MCSPI2	McSPI 2 idle status. 0x0: McSPI 2 can be accessed. 0x1: McSPI 2 cannot be accessed. Any access may return an error.	R	0x1

Bits	Field Name	Description	Type	Reset
18	ST_MCSPI1	McSPI 1 idle status. 0x0: McSPI 1 can be accessed. 0x1: McSPI 1 cannot be accessed. Any access may return an error.	R	0x1
17	ST_I2C3	I2C 3 idle status. 0x0: I2C 3 can be accessed. 0x1: I2C 3 cannot be accessed. Any access may return an error.	R	0x1
16	ST_I2C2	I2C 2 idle status. 0x0: I2C 2 can be accessed. 0x1: I2C 2 cannot be accessed. Any access may return an error.	R	0x1
15	ST_I2C1	I2C 1 idle status. 0x0: I2C 1 can be accessed. 0x1: I2C 1 cannot be accessed. Any access may return an error.	R	0x1
14	ST_UART2	UART 2 idle status. 0x0: UART 2 can be accessed. 0x1: UART 2 cannot be accessed. Any access may return an error.	R	0x1
13	ST_UART1	UART 1 idle status. 0x0: UART 1 can be accessed. 0x1: UART 1 cannot be accessed. Any access may return an error.	R	0x1
12	ST_GPT11	GPTIMER 11 idle status. 0x0: GPTIMER 11 can be accessed. 0x1: GPTIMER 11 cannot be accessed. Any access may return an error.	R	0x1
11	ST_GPT10	GPTIMER 10 idle status. 0x0: GPTIMER 10 can be accessed. 0x1: GPTIMER 10 cannot be accessed. Any access may return an error.	R	0x1
10	ST_MCBSP5	McBSP 5 idle status. 0x0: McBSP 5 can be accessed. 0x1: McBSP 5 cannot be accessed. Any access may return an error.	R	0x1
9	ST_MCBSP1	McBSP 1 idle status. 0x0: McBSP 1 can be accessed. 0x1: McBSP 1 cannot be accessed. Any access may return an error.	R	0x1
8:7	RESERVED	Reserved	R	0x3
6	ST_SCMCTRL	System Control Module idle status 0x0: System Control Module can be accessed. 0x1: System Control Module cannot be accessed. Any access may return an error.	R	0x1
5	ST_IPSS_IDLE	IPSS idle status. 0x0: IPSS can be accessed. 0x1: IPSS cannot be accessed. Any access may return an error.	R	0x1
4:3	RESERVED	Reserved	R	0x3
2	ST_SDMA	System DMA standby status. 0x0: System DMA is active. 0x1: System DMA is in standby mode.	R	0x1

Bits	Field Name	Description	Type	Reset
1	ST_EMIF4	EMIF4 idle status. 0x0: EMIF4 can be accessed. 0x1: EMIF4 cannot be accessed. Any access may return an error.	R	0x1
0	RESERVED	Reserved	R	0x1

4.12.1.4.7 CM_IDLEST2_CORE

Table 4-102. CM_IDLEST2_CORE

Address Offset	0x0000 0024		
Physical Address	0x4800 4A24	**Instance**	CORE_CM
Description	CORE modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															

Bits	Field Name	Description	Type	Reset
31:0	RESERVED	Reserved	R	0x0000001F

4.12.1.4.8 CM_IDLEST3_CORE

Table 4-103. CM_IDLEST3_CORE

Address Offset	0x0000 0028		
Physical Address	0x4800 4A28	**Instance**	CORE_CM
Description	CORE modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													ST_USBTLL	RESERVED	ST_CPEFUSE

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved	R	0x00000001
2	ST_USBTLL	USB TLL idle status. 0x0: USB TLL can be accessed. 0x1: USB TLL cannot be accessed. Any access may return an error.	R	0x1
1	RESERVED	Reserved	R	0x0
0	ST_CPEFUSE	CPEFUSE idle status. 0x0: CPEFUSE can be accessed. 0x1: CPEFUSE cannot be accessed. Any access may return an error.	R	0x1

4.12.1.4.9 CM_AUTOIDLE1_CORE

Table 4-104. CM_AUTOIDLE1_CORE

Address Offset	0x0000 0030		
Physical Address	0x4800 4A30	**Instance**	CORE_CM
Description	This register controls the automatic control of the CORE modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	AUTO_MMC3	RESERVED				AUTO_MMC2	AUTO_MMC1	AUTO_UART4	AUTO_HDQ	AUTO_MCSPI4	AUTO_MCSPI3	AUTO_MCSPI2	AUTO_MCSPI1	AUTO_I2C3	AUTO_I2C2	AUTO_I2C1	AUTO_UART2	AUTO_UART1	AUTO_GPT11	AUTO_GPT10	AUTO_MCBSP5	AUTO_MCBSP1	RESERVED	AUTO_SCMCTRL	RESERVED	AUTO_IPSS	RESERVED				

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved: keep at reset value.	RW	0x0
30	AUTO_MMC3	MMC SDIO 3 auto clock control. 0x0: MMC 3 interface clock is unrelated to the domain state transition. 0x1: MMC 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
29:26	RESERVED	Reserved: keep at reset value.	RW	0x0
25	AUTO_MMC2	MMC SDIO 2 auto clock control. 0x0: MMC 2 interface clock is unrelated to the domain state transition. 0x1: MMC 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
24	AUTO_MMC1	MMC SDIO 1 auto clock control. 0x0: MMC 1 interface clock is unrelated to the domain state transition. 0x1: MMC 1 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
23	AUTO_UART4	UART4 auto clock control. 0x0: UART4 interface clock is unrelated to the domain state transition. 0x1: UART4 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
22	AUTO_HDQ	HDQ-1 wire auto clock control. 0x0: HDQ interface clock is unrelated to the domain state transition. 0x1: HDQ interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
21	AUTO_MCSPI4	McSPI 4 auto clock control. 0x0: McSPI 4 interface clock is unrelated to the domain state transition. 0x1: McSPI 4 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
20	AUTO_MCSPI3	McSPI 3 auto clock control. 0x0: McSPI 3 interface clock is unrelated to the domain state transition. 0x1: McSPI 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
19	AUTO_MCSPI2	McSPI 2 auto clock control. 0x0: McSPI 2 interface clock is unrelated to the domain state transition. 0x1: McSPI 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0

Bits	Field Name	Description	Type	Reset
18	AUTO_MCSPI1	McSPI 1 auto clock control. 0x0: McSPI 1 interface clock is unrelated to the domain state transition. 0x1: McSPI 1 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
17	AUTO_I2C3	I2C 3 auto clock control. 0x0: I2C 3 interface clock is unrelated to the domain state transition. 0x1: I2C 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
16	AUTO_I2C2	I2C 2 auto clock control. 0x0: I2C 2 interface clock is unrelated to the domain state transition. 0x1: I2C 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
15	AUTO_I2C1	I2C 1 auto clock control. 0x0: I2C 1 interface clock is unrelated to the domain state transition. 0x1: I2C 1 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
14	AUTO_UART2	UART 2 auto clock control. 0x0: UART 2 interface clock is unrelated to the domain state transition. 0x1: UART 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
13	AUTO_UART1	UART 1 auto clock control. 0x0: UART 1 interface clock is unrelated to the domain state transition. 0x1: UART 1 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
12	AUTO_GPT11	GPTIMER 11 auto clock control. 0x0: GPTIMER 11 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 11 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
11	AUTO_GPT10	GPTIMER 10 auto clock control. 0x0: GPTIMER 10 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 10 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
10	AUTO_MCBSP5	McBSP 5 auto clock control. 0x0: McBSP 5 interface clock is unrelated to the domain state transition. 0x1: McBSP 5 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
9	AUTO_MCBSP1	McBSP 1 auto clock control. 0x0: McBSP 1 interface clock is unrelated to the domain state transition. 0x1: McBSP 1 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
8:7	RESERVED	Reserved: keep at reset value.	R	0x0
6	AUTO_SCMCTRL	System Control Module auto clock control 0x0: System Control Module interface clock is unrelated to the domain state transition. 0x1: System Control Module interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
5	RESERVED	Reserved: keep at reset value.	R	0x0
4	AUTO_IPSS	IPSS auto clock control. 0x0: IPSS interface clock is unrelated to the domain state transition. 0x1: IPSS interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
3:0	RESERVED	Reserved: keep at reset value.	R	0x8

4.12.1.4.10 CM_AUTOIDLE2_CORE

Table 4-105. CM_AUTOIDLE2_CORE

Address Offset	0x0000 0034		
Physical Address	0x4800 4A34	**Instance**	CORE_CM
Description	This register controls the automatic control of the CORE modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															

Bits	Field Name	Description	Type	Reset
31:0	RESERVED	Reserved: keep at reset value.	R	0x00000000

4.12.1.4.11 CM_AUTOIDLE3_CORE

Table 4-106. CM_AUTOIDLE3_CORE

Address Offset	0x0000 0038		
Physical Address	0x4800 4A38	**Instance**	CORE_CM
Description	This register controls the automatic control of the CORE modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													AUTO_USBTLL	RESERVED	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	AUTO_USBTLL	USB TLL auto clock control. 0x0: USB TLL interface clock is unrelated to the domain state transition. 0x1: USB TLL interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
1:0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.1.4.12 CM_CLKSEL_CORE

Table 4-107. CM_CLKSEL_CORE

Address Offset	0x0000 0040		
Physical Address	0x4800 4A40	**Instance**	CORE_CM
Description	CORE modules clock selection.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CLKSEL_GPT11	CLKSEL_GPT10	RESERVED		CLKSEL_L4		CLKSEL_L3	

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000001
7	CLKSEL_GPT11	Selects GPTIMER 11 source clock 0x0: source is CM_32K_CLK 0x1: source is CM_SYS_CLK	RW	0x0
6	CLKSEL_GPT10	Selects GPTIMER 10 source clock 0x0: source is CM_32K_CLK 0x1: source is CM_SYS_CLK	RW	0x0
5:4	RESERVED	Reserved: keep at reset value.	R	0x0
3:2	CLKSEL_L4	Selects Peripherals interconnect clock (L4_CLK); Other enums: Reserved 0x1: L4_CLK is L3_CLK divided by 1 (boot mode only) 0x2: L4_CLK is L3_CLK divided by 2	RW	0x1
1:0	CLKSEL_L3	Selects L3 interconnect clock (L3_CLK); Other enums: Reserved 0x1: L3_CLK is CORE_CLK divided by 1 0x2: L3_CLK is CORE_CLK divided by 2	RW	0x1

4.12.1.4.13 CM_CLKSTCTRL_CORE

Table 4-108. CM_CLKSTCTRL_CORE

Address Offset	0x0000 0048		
Physical Address	0x4800 4A48	**Instance**	CORE_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		

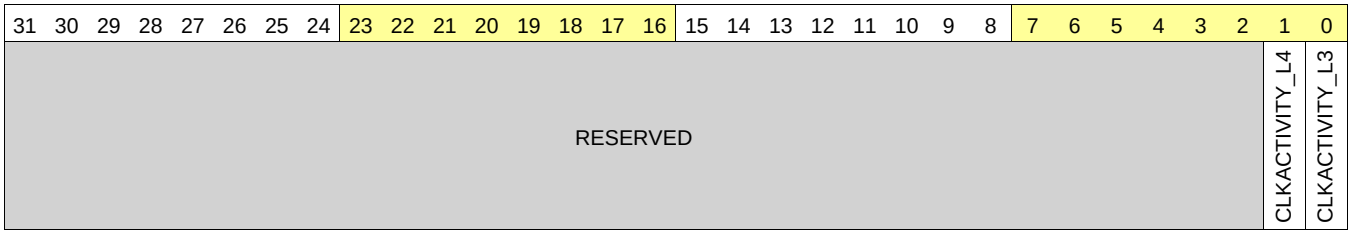
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												CLKTRCTRL_L4		CLKTRCTRL_L3	

Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved: keep at reset value.	R	0x0000000
3:2	CLKTRCTRL_L4	Controls the clock state transition of the L4 clock domain. 0x0: Automatic transition is disabled 0x1: Reserved 0x2: Reserved 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0
1:0	CLKTRCTRL_L3	Controls the clock state transition of the L3 clock domain. 0x0: Automatic transition is disabled 0x1: Reserved 0x2: Reserved 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0

4.12.1.4.14 CM_CLKSTST_CORE

Table 4-109. CM_CLKSTST_CORE

Address Offset	0x0000 004C		
Physical Address	0x4800 4A4C	**Instance**	CORE_CM
Description	This register provides a status on the interface clock activity in the domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved	R	0x00000000
1	CLKACTIVITY_L4	L4_ICLK interface clock activity status 0x0: No domain interface clock activity 0x1: Domain interface clock is active	R	0x0
0	CLKACTIVITY_L3	L3_ICLK interface clock activity status 0x0: No domain interface clock activity 0x1: Domain interface clock is active	R	0x0

4.12.1.5 SGX_CM Register Descriptions

4.12.1.5.1 CM_FCLKEN_SGX

Table 4-110. CM_FCLKEN_SGX

Address Offset	0x0000 0000		
Physical Address	0x4800 4B00	**Instance**	SGX_CM
Description	Controls the Graphic engine functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_SGX	RESERVED

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_SGX	SGX functional clock enable 0x0: SGX_FCLK is disabled 0x1: SGX_FCLK is enabled	RW	0x0
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.1.5.2 CM_ICLKEN_SGX

Table 4-111. CM_ICLKEN_SGX

Address Offset	0x0000 0010		
Physical Address	0x4800 4B10	**Instance**	SGX_CM
Description	Controls the Graphic engine interface clock activity.		
Type	RW		

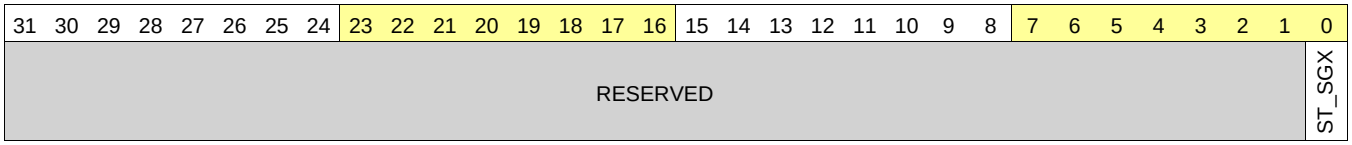
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															EN_SGX

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value	R	0x00000000
0	EN_SGX	SGX interface clock control 0x0: SGX_L3_ICLK is disabled 0x1: SGX_L3_ICLK is enabled	RW	0x0

4.12.1.5.3 CM_IDLEST_SGX

Table 4-112. CM_IDLEST_SGX

Address Offset	0x0000 0020		
Physical Address	0x4800 4B20	**Instance**	SGX_CM
Description	SGX standby status. This register is read only and automatically updated.		
Type	R		

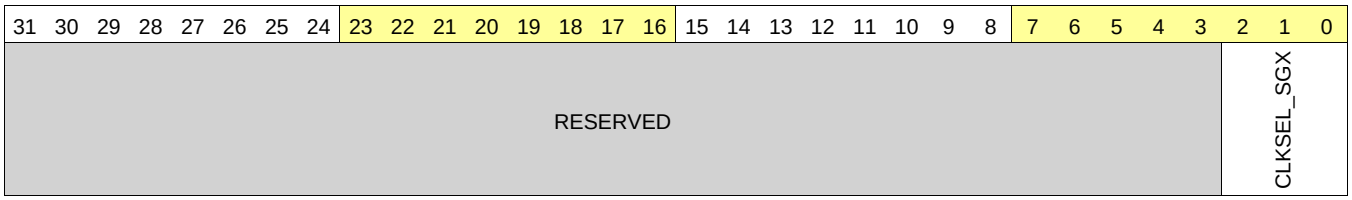


Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	ST_SGX	SGX standby status. 0x0: SGX subsystem is active. 0x1: SGX subsystem is in standby mode.	R	0x1

4.12.1.5.4 CM_CLKSEL_SGX

Table 4-113. CM_CLKSEL_SGX

Address Offset	0x0000 0040		
Physical Address	0x4800 4B40	**Instance**	SGX_CM
Description	SGX clock selection.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value	R	0x00000000
2:0	CLKSEL_SGX	Selects SGX functional clock; Other enums: Reserved. See device specific Data Manual for more details. 0x0: SGX_FCLK is CORE_CLK divided by 3 0x1: SGX_FCLK is CORE_CLK divided by 4 0x2: SGX_FCLK is CORE_CLK divided by 6 0x3: SGX_FCLK clock is CM_96M_FCLK clock	RW	0x0

4.12.1.5.5 CM_SLEEPDEP_SGX

Table 4-114. CM_SLEEPDEP_SGX

Address Offset	0x0000 0044		
Physical Address	0x4800 4B44	**Instance**	SGX_CM
Description	This register allows enabling or disabling the sleep transition dependency of SGX domain with respect to other domain.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_MPU	RESERVED

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_MPU	MPU domain dependency 0x0: SGX domain sleep dependency with MPU domain is disabled. 0x1: SGX domain sleep dependency with MPU domain is enabled.	RW	0x0
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.1.5.6 CM_CLKSTCTRL_SGX

Table 4-115. CM_CLKSTCTRL_SGX

Address Offset	0x0000 0048		
Physical Address	0x4800 4B48	**Instance**	SGX_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		

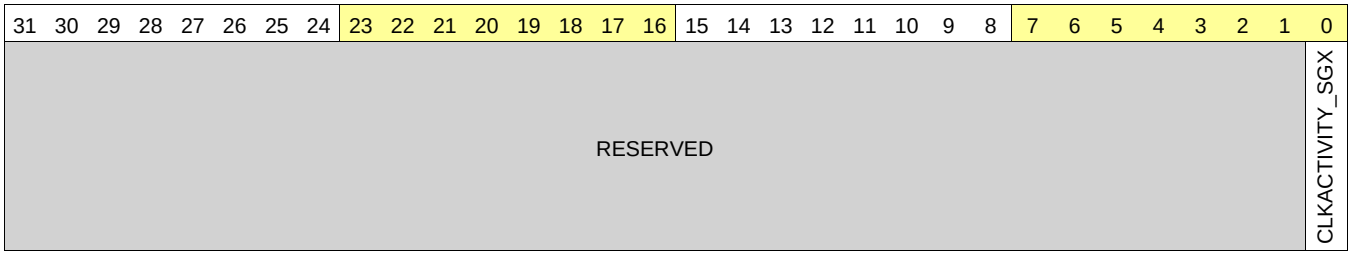
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														CLKTRCTRL_SGX	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_SGX	Controls the clock state transition of the SGX clock domain. 0x0: Automatic transition is disabled 0x1: Start a software supervised sleep transition on the domain 0x2: Start a software supervised wake-up transition on the domain 0x3: Automatic transition is enabled. Transition is supervised by the hardware. **NOTE:** This mode must not be used when smart-standby is enabled.	RW	0x0

4.12.1.5.7 CM_CLKSTST_SGX

Table 4-116. CM_CLKSTST_SGX

Address Offset	0x0000 004C		
Physical Address	0x4800 4B4C	**Instance**	SGX_CM
Description	This register provides a status on the interface clock activity in the domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	CLKACTIVITY_SGX	Interface clock activity status 0x0: No domain interface clock activity 0x1: Domain interface clock is active	R	0x0

4.12.1.6 WKUP_CM Register Descriptions

4.12.1.6.1 CM_FCLKEN_WKUP

Table 4-117. CM_FCLKEN_WKUP

Address Offset	0x0000 0000		
Physical Address	0x4800 4C00	**Instance**	WKUP_CM
Description	Controls the modules functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										EN_WDT2	RESERVED	EN_GPIO1	RESERVED		EN_GPT1

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved: keep at reset value.	R	0x0000000
5	EN_WDT2	WDTIMER 2 functional clock control. 0x0: WDTIMER 2 functional clock is disabled 0x1: WDTIMER 2 functional clock is enabled	RW	0x0
4	RESERVED	Reserved: keep at reset value.	R	0x0
3	EN_GPIO1	GPIO 1 clock control 0x0: GPIO 1 functional clock is disabled 0x1: GPIO 1 functional clock is enabled	RW	0x0
2:1	RESERVED	Reserved: keep at reset value.	R	0x0
0	EN_GPT1	GPTIMER 1 clock control 0x0: GPTIMER 1 functional clock is disabled 0x1: GPTIMER 1 functional clock is enabled	RW	0x0

4.12.1.6.2 CM_ICLKEN_WKUP

Table 4-118. CM_ICLKEN_WKUP

Address Offset	0x0000 0010		
Physical Address	0x4800 4C10	**Instance**	WKUP_CM
Description	Controls the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										EN_WDT2	EN_WDT1	EN_GPIO1	EN_32KSYNC	EN_GPT12	EN_GPT1

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved: keep at reset value.	R	0x0000000
5	EN_WDT2	WDTIMER 2 interface clock 0x0: WDTIMER 2 interface clock is disabled 0x1: WDTIMER 2 interface clock is enabled	RW	0x0
4	EN_WDT1	WDTIMER 1 (Secure) interface clock control. 0x0: WDTIMER 1 (Secure) interface clock is disabled 0x1: WDTIMER 1 (Secure) interface clock is enabled	RW	0x0
3	EN_GPIO1	GPIO 1 interface clock control 0x0: GPIO 1 interface clock is disabled 0x1: GPIO 1 interface clock is enabled	RW	0x0
2	EN_32KSYNC	32 kHz Sync Timer interface clock control 0x0: 32k Sync Timer interface clock is disabled 0x1: 32k Sync Timer interface clock is enabled	RW	0x0
1	EN_GPT12	GPTIMER 12 interface clock control. 0x0: GPTIMER 12 interface clock is disabled 0x1: GPTIMER 12 interface clock is enabled	RW	0x0
0	EN_GPT1	GPTIMER 1 interface clock control 0x0: GPTIMER 1 interface clock is disabled 0x1: GPTIMER 1 interface clock is enabled	RW	0x0

4.12.1.6.3 CM_IDLEST_WKUP

Table 4-119. CM_IDLEST_WKUP

Address Offset	0x0000 0020		
Physical Address	0x4800 4C20	**Instance**	WKUP_CM
Description	WAKEUP domain modules access monitoring. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										ST_WDT2	ST_WDT1	ST_GPIO1	ST_32KSYNC	ST_GPT12	ST_GPT1

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved	R	0x000000B
5	ST_WDT2	WDTIMER 2 idle status 0x0: WDTIMER 2 can be accessed. 0x1: WDTIMER 2 cannot be accessed. Any access may return an error.	R	0x1
4	ST_WDT1	WDTIMER 1 (Secure) idle status. 0x0: WDTIMER 1 (Secure) can be accessed. 0x1: WDTIMER 1 (Secure) cannot be accessed. Any access may return an error.	R	0x1
3	ST_GPIO1	GPIO 1 idle status 0x0: GPIO 1 can be accessed. 0x1: GPIO 1 cannot be accessed. Any access may return an error.	R	0x1
2	ST_32KSYNC	32 kHz Sync Timer idle status 0x0: 32k Sync Timer can be accessed. 0x1: 32k Sync Timer cannot be accessed. Any access may return an error.	R	0x1
1	ST_GPT12	GPTIMER 12 idle status. 0x0: GPTIMER 12 can be accessed. 0x1: GPTIMER 12 cannot be accessed. Any access may return an error.	R	0x1
0	ST_GPT1	GPTIMER 1 idle status 0x0: GPTIMER 1 can be accessed 0x1: GPTIMER 1 cannot be accessed. Any access may return an error.	R	0x1

4.12.1.6.4 CM_AUTOIDLE_WKUP

Table 4-120. CM_AUTOIDLE_WKUP

Address Offset	0x0000 0030		
Physical Address	0x4800 4C30	**Instance**	WKUP_CM
Description	This register controls the automatic control of the WAKEUP modules interface clock activity. This activity is related to CORE domain activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										AUTO_WDT2	AUTO_WDT1	AUTO_GPIO1	AUTO_32KSYNC	AUTO_GPT12	AUTO_GPT1

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved: keep at reset value.	R	0x0000000
5	AUTO_WDT2	WDTIMER 2 autoidle control 0x0: WDTIMER 2 interface clock is unrelated to the domain activity. 0x1: WDTIMER 2 interface clock is automatically enabled or disabled according to the domain activity.	RW	0x0
4	AUTO_WDT1	WDTIMER 1 (Secure) auto clock control. 0x0: WDTIMER 1 interface clock is unrelated to the domain state transition. 0x1: WDTIMER 1 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
3	AUTO_GPIO1	GPIO 1 autoidle control 0x0: GPIO 1 interface clock is unrelated to the domain activity. 0x1: GPIO 1 interface clock is automatically enabled / disabled according to the domain activity.	RW	0x0
2	AUTO_32KSYNC	32 kHz Sync Timer autoidle control 0x0: 32k Sync Timer interface clock is unrelated to the domain activity. 0x1: 32k Sync Timer interface clock is automatically enabled or disabled according to the domain activity.	RW	0x0
1	AUTO_GPT12	GPTIMER 12 auto clock control. 0x0: GPTIMER 12 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 12 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
0	AUTO_GPT1	GPTIMER 1 autoidle control 0x0: GPTIMER 1 interface clock is unrelated to the domain activity. 0x1: GPTIMER 1 interface clock is automatically enabled or disabled according to the domain activity.	RW	0x0

4.12.1.6.5 CM_CLKSEL_WKUP

Table 4-121. CM_CLKSEL_WKUP

Address Offset	0x0000 0040		
Physical Address	0x4800 4C40	**Instance**	WKUP_CM
Description	WAKEUP domain modules source clock selection.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													CLKSEL_RM		CLKSEL_GPT1

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000002
2:1	CLKSEL_RM	Selects the Reset Manager clock; Other enums: Reserved 0x1: RM_ICLK is L4_CLK divided by 1 0x2: RM_ICLK is L4_CLK divided by 2	RW	0x1
0	CLKSEL_GPT1	Selects GPTIMER 1 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0

4.12.1.7 Clock_Control_Reg_CM Register Descriptions

4.12.1.7.1 CM_CLKEN_PLL

Table 4-122. CM_CLKEN_PLL

Address Offset	0x0000 0000		
Physical Address	0x4800 4D00	**Instance**	Clock_Control_Reg_CM
Description	This register allows controlling the DPLL3 and DPLL4 modes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWRDN_EMU_PERIPH	RESERVED	PWRDN_DSS1	PWRDN_TV	PWRDN_96M	EN_PERIPH_DPLL_LPMODE	PERIPH_DPLL_RAMPTIME		PERIPH_DPLL_FREQSEL				EN_PERIPH_DPLL_DRIFTGUARD		EN_PERIPH_DPLL		RESERVED			PWRDN_EMU_CORE	RESERVED	EN_CORE_DPLL_LPMODE	CORE_DPLL_RAMPTIME		CORE_DPLL_FREQSEL				EN_CORE_DPLL_DRIFTGUARD		EN_CORE_DPLL	

Bits	Field Name	Description	Type	Reset
31	PWRDN_EMU_PERIPH	This bit allows to power-down or not the DPLL4_M6X2_CLK HSDIVIDER path. 0x0: Power-up the DPLL4_M6X2_CLK HSDIVIDER path. 0x1: Power-down the DPLL4_M6X2_CLK HSDIVIDER path. Writing this bit to 1 will take effect immediately.	RW	0x0
30	RESERVED	Reserved: keep at reset value.	R	0x0
29	PWRDN_DSS1	This bit allows to power-down or not the DPLL4_M4X2_CLK HSDIVIDER path. 0x0: Power-up the DPLL4_M4X2_CLK HSDIVIDER path. 0x1: Power-down the DPLL4_M4X2_CLK HSDIVIDER path. Writing this bit to 1 will take effect immediately.	RW	0x0
28	PWRDN_TV	This bit allows to power-down or not the DPLL4_M3X2_CLK HSDIVIDER path. 0x0: Power-up the DPLL4_M3X2_CLK HSDIVIDER path. 0x1: Power-down the DPLL4_M3X2_CLK HSDIVIDER path. Writing this bit to 1 will take effect immediately.	RW	0x0
27	PWRDN_96M	This bit allows to power-down or not the DPLL4_M2X2_CLK path. 0x0: Power-up the DPLL4_M2X2_CLK path. 0x1: Power-down the DPLL4_M2X2_CLK path. Writing this bit to 1 will take effect immediately.	RW	0x0
26	EN_PERIPH_DPLL_LPMODE	This bit allows to enable or disable the LP mode of the DPLL4. Writing this bit to switch the mode between LP or normal mode will take effect only when the DPLL will have transition into the bypass or stop state, followed by a lock or re-lock of the DPLL. 0x0: Disables the DPLL LP mode to re-enter the normal mode at the following lock or re-lock sequence. 0x1: Enables the DPLL LP mode to enter the LP mode at the following lock or re-lock sequence.	RW	0x0

Bits	Field Name	Description	Type	Reset
25:24	PERIPH_DPLL_RAMPTIME	The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and relock. The frequency ramping occurs in a maximum of four steps in frequency before the DPLL frequency lock indicator is asserted. The ramp step size is specified in the number of DPLL internal reference clock cycles. The internal DPLL reference clock cycle is given by: Fint = DPLL input reference clock frequency/(N + 1), where N is the value in the CM_CLKSEL2_PLL[6:0] PERIPH_DPLL_DIV bit field. This register is used to enable and disable the DPLL ramping feature. If enabled, it is also used to select a range of ramp step sizes. The step size used per range depends on the DPLL internal reference clock frequency: the lower the frequency, the smaller the ramp step size. 0x0: Disables the frequency ramping feature. 0x1: Ramp step size range is 2–40 Fint cycles. 0x2: Ramp step size range is 4–80 Fint cycles. 0x3: Ramp step size range is 12–240 Fint cycles.	RW	0x0
23:20	PERIPH_DPLL_FREQSEL	This bit field allows selecting the proper range of the DPLL4 internal frequency depending on the DPLL reference clock and the N divider. 0x3: 0.75 MHz—1.0 MHz 0x4: 1.0 MHz—1.25 MHz 0x5: 1.25 MHz—1.5 MHz 0x6: 1.5 MHz—1.75 MHz 0x7: 1.75 MHz—2.1 MHz 0xB: 7.5 MHz—10 MHz 0xC: 10 MHz—12.5 MHz 0xD: 12.5 MHz—15 MHz 0xE: 15 MHz—17.5 MHz 0xF: 17.5 MHz—21 MHz	RW	0x1
19	EN_PERIPH_DPLL_DRIFTGUARD	This bit allows to enable or disable the automatic recalibration feature of the DPLL4. The DPLL4 will automatically start a recalibration process upon assertion of the recal flag if this bit is set. 0x0: Disables the DPLL4 automatic recalibration mode 0x1: Enables the DPLL4 automatic recalibration mode	RW	0x0
18:16	EN_PERIPH_DPLL	DPLL4 control; Other enums: Reserved 0x1: Put the DPLL4 in low power stop mode 0x7: Enables the DPLL4 in lock mode	RW	0x1
15:13	RESERVED	Reserved: keep at reset value.	R	0x0
12	PWRDN_EMU_CORE	This bit allows to power-down or not the DPLL3_M3X2 HSDIVIDER path. 0x0: Power-up the DPLL3_M3X2 HSDIVIDER path. 0x1: Power-down the DPLL3_M3X2 HSDIVIDER path. Writing this bit to 1 will take effect immediately.	RW	0x0
11	RESERVED	Reserved: keep at reset value.	R	0x0
10	EN_CORE_DPLL_LPMODE	This bit allows to enable or disable the LP mode of the DPLL3. Writing this bit to switch the mode between LP or normal mode will take effect only when the DPLL will have transition into the bypass or stop state, followed by a lock or re-lock of the DPLL. 0x0: Disables the DPLL LP mode to re-enter the normal mode at the following lock or re-lock sequence. 0x1: Enables the DPLL LP mode to enter the LP mode at the following lock or re-lock sequence.	RW	0x0

Bits	Field Name	Description	Type	Reset
9:8	CORE_DPLL_RAMPTIME	The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and relock. The frequency ramping occurs in a maximum of four steps in frequency before the DPLL frequency lock indicator is asserted. The ramp step size is specified in the number of DPLL internal reference clock cycles. The internal DPLL reference clock cycle is given by: Fint = DPLL input reference clock frequency/(N + 1), where N is the value in the CM_CLKSEL1_PLL[14:8] CORE_DPLL_DIV bit field. This register is used to enable and disable the DPLL ramping feature. If enabled, it is also used to select a range of ramp step sizes. The step size used per range depends on the DPLL internal reference clock frequency: the lower the frequency, the smaller the ramp step size. 0x0: Disables the frequency ramping feature. 0x1: Ramp step size range is 2–40 Fint cycles. 0x2: Ramp step size range is 4–80 Fint cycles. 0x3: Ramp step size range is 12–240 Fint cycles.	RW	0x0
7:4	CORE_DPLL_FREQSEL	This bit field allows selecting the proper range of the DPLL3 internal frequency depending on the DPLL reference clock and the N divider. 0x3: 0.75 MHz—1.0 MHz 0x4: 1.0 MHz—1.25 MHz 0x5: 1.25 MHz—1.5 MHz 0x6: 1.5 MHz—1.75 MHz 0x7: 1.75 MHz—2.1 MHz 0xB: 7.5 MHz—10 MHz 0xC: 10 MHz—12.5 MHz 0xD: 12.5 MHz—15 MHz 0xE: 15 MHz—17.5 MHz 0xF: 17.5 MHz—21 MHz	RW	0x1
3	EN_CORE_DPLL_DRIFTGUARD	This bit allows to enable or disable the automatic recalibration feature of the DPLL3. The DPLL3 will automatically start a recalibration process upon assertion of the recal flag if this bit is set. 0x0: Disables the DPLL3 automatic recalibration mode 0x1: Enables the DPLL3 automatic recalibration mode	RW	0x0
2:0	EN_CORE_DPLL	DPLL3 control; Other enums: Reserved 0x5: Put the DPLL3 in low power bypass 0x6: Put the DPLL3 in fast relock bypass 0x7: Enables the DPLL3 in lock mode	RW	0x5

4.12.1.7.2 CM_CLKEN2_PLL

Table 4-123. CM_CLKEN2_PLL

Address Offset	0x0000 0004		
Physical Address	0x4800 4D04	**Instance**	Clock_Control_Reg_CM
Description	This register allows controlling the DPLL5 modes.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					EN_PERIPH2_DPLL_LPMODE	PERIPH2_DPLL_RAMPTIME			PERIPH2_DPLL_FREQSEL				EN_PERIPH2_DPLL_DRIFTGUARD	EN_PERIPH2_DPLL		

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Reserved: keep at reset value.	R	0x000000
10	EN_PERIPH2_DPLL_LPMODE	This bit allows to enable or disable the LP mode of the DPLL5. Writing this bit to switch the mode between LP or normal mode will take effect only when the DPLL will have transition into the bypass or stop state, followed by a lock or re-lock of the DPLL. 0x0: Disables the DPLL LP mode to re-enter the normal mode at the following lock or re-lock sequence. 0x1: Enables the DPLL LP mode to enter the LP mode at the following lock or re-lock sequence.	RW	0x0
9:8	PERIPH2_DPLL_RAMPTIME	The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and relock. The frequency ramping occurs in a maximum of four steps in frequency before the DPLL frequency lock indicator is asserted. The ramp step size is specified in the number of DPLL internal reference clock cycles. The internal DPLL reference clock cycle is given by: Fint = DPLL input reference clock frequency/(N + 1), where N is the value in the CM_CLKSEL4_PLL[6:0] PERIPH2_DPLL_DIV bit field. This register is used to enable and disable the DPLL ramping feature. If enabled, it is also used to select a range of ramp step sizes. The step size used per range depends on the DPLL internal reference clock frequency: the lower the frequency, the smaller the ramp step size. 0x0: Disables the frequency ramping feature. 0x1: Ramp step size range is 2–40 Fint cycles. 0x2: Ramp step size range is 4–80 Fint cycles. 0x3: Ramp step size range is 12–240 Fint cycles.	RW	0x0

Bits	Field Name	Description	Type	Reset
7:4	PERIPH2_DPLL_FREQSEL	This bit field allows selecting the proper range of the second PERIPHERAL DPLL internal frequency depending on the DPLL reference clock and the N divider. 0x3: 0.75 MHz—1.0 MHz 0x4: 1.0 MHz—1.25 MHz 0x5: 1.25 MHz—1.5 MHz 0x6: 1.5 MHz—1.75 MHz 0x7: 1.75 MHz—2.1 MHz 0xB: 7.5 MHz—10 MHz 0xC: 10 MHz—12.5 MHz 0xD: 12.5 MHz—15 MHz 0xE: 15 MHz—17.5 MHz 0xF: 17.5 MHz—21 MHz	RW	0x1
3	EN_PERIPH2_DPLL_DRIFTGUARD	This bit allows to enable or disable the automatic recalibration feature of the DPLL5. The DPLL5 will automatically start a recalibration process upon assertion of the recal flag if this bit is set. 0x0: Disables the DPLL5 automatic recalibration mode 0x1: Enables the DPLL5 automatic recalibration mode	RW	0x0
2:0	EN_PERIPH2_DPLL	DPLL5 control; Other enums: Reserved 0x1: Put the second DPLL5 in low power stop mode 0x7: Enables the DPLL5 in lock mode	RW	0x1

4.12.1.7.3 CM_IDLEST_CKGEN

Table 4-124. CM_IDLEST_CKGEN

Address Offset	0x0000 0020		
Physical Address	0x4800 4D20	**Instance**	Clock_Control_Reg_CM
Description	This register allows monitoring the master clock activity. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		ST_EMU_PERIPH_CLK	RESERVED	ST_DSS1_CLK	ST_TV_CLK	ST_FUNC96M_CLK	ST_EMU_CORE_CLK	RESERVED		ST_54M_CLK	ST_12M_CLK	ST_48M_CLK	ST_96M_CLK	ST_PERIPH_CLK	ST_CORE_CLK

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Reserved	R	0x00000
13	ST_EMU_PERIPH_CLK	Emulation clock activity at the output stage of the DPLL4 0x0: EMU_PER_ALWON_CLK is not active 0x1: EMU_PER_ALWON_CLK is active	R	0x0
12	RESERVED	Reserved	R	0x0
11	ST_DSS1_CLK	DSS functional clock 1 activity at the output stage of the DPLL4 0x0: DSS1_ALWON_FCLK is not active 0x1: DSS1_ALWON_FCLK is active	R	0x0
10	ST_TV_CLK	TV clock activity at the output stage of the DPLL4 0x0: DPLL4_M3X2_CLK is not active 0x1: DPLL4_M3X2_CLK is active	R	0x0
9	ST_FUNC96M_CLK	96 MHz clock activity at the output stage of the DPLL4 0x0: DPLL4_M2X2_CLK is not active 0x1: DPLL4_M2X2_CLK is active	R	0x0
8	ST_EMU_CORE_CLK	Emulation clock activity at the output stage of the DPLL3 0x0: EMU_CORE_ALWON_CLK is not active 0x1: EMU_CORE_ALWON_CLK is active	R	0x0
7:6	RESERVED	Reserved	R	0x0
5	ST_54M_CLK	Functional clock 54 MHz activity 0x0: 54MHz clock is not active 0x1: 54MHz clock is active	R	0x0
4	ST_12M_CLK	Functional clock 12 MHz activity 0x0: 12M_FCLK is not active 0x1: 12M_FCLK is active	R	0x0
3	ST_48M_CLK	Functional clock 48 MHz activity 0x0: 48M_FCLK is not active 0x1: 48M_FCLK is active	R	0x0
2	ST_96M_CLK	Functional clock 96 MHz activity 0x0: 96M_FCLK is not active 0x1: 96M_FCLK is active	R	0x0

Bits	Field Name	Description	Type	Reset
1	ST_PERIPH_CLK	DPLL4 clock activity 0x0: DPLL4 is bypassed 0x1: DPLL4 is locked	R	0x0
0	ST_CORE_CLK	DPLL3 clock activity 0x0: DPLL3 is bypassed 0x1: DPLL3 is locked	R	0x0

4.12.1.7.4 CM_IDLEST2_CKGEN

Table 4-125. CM_IDLEST2_CKGEN

Address Offset	0x0000 0024		
Physical Address	0x4800 4D24	**Instance**	Clock_Control_Reg_CM
Description	This register allows monitoring the master clock activity. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												ST_FUNC120M_CLK	RESERVED	ST_120M_CLK	ST_PERIPH2_CLK

Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved	R	0x0000000
3	ST_FUNC120M_CLK	120 MHz clock activity at the output stage of the DPLL5 0x0: DPLL5_M2_CLK is not active 0x1: DPLL5_M2_CLK is active	R	0x0
2	RESERVED	Reserved	R	0x0
1	ST_120M_CLK	Functional clock 120 MHz activity 0x0: 120M_FCLK is not active 0x1: 120M_FCLK is active	R	0x0
0	ST_PERIPH2_CLK	DPLL5 clock activity 0x0: DPLL5 is bypassed 0x1: DPLL5 is locked	R	0x0

4.12.1.7.5 CM_AUTOIDLE_PLL

Table 4-126. CM_AUTOIDLE_PLL

Address Offset	0x0000 0030		
Physical Address	0x4800 4D30	**Instance**	Clock_Control_Reg_CM
Description	This register provides automatic control over the DPLL3 and DPLL4 activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										AUTO_PERIPH_DPLL			AUTO_CORE_DPLL		

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved: keep at reset value.	R	0x0000000
5:3	AUTO_PERIPH_DPLL	DPLL4 automatic control; Other enums: Reserved 0x0: Auto control disabled 0x1: DPLL4 is automatically put in low power stop mode when none of the 96 MHz and 54 MHz clocks are required anymore. It is also restarted automatically.	RW	0x0
2:0	AUTO_CORE_DPLL	DPLL3 automatic control; Other enums: Reserved 0x0: Auto control disabled 0x1: DPLL3 is automatically put in low power stop mode when the CORE clock is not required anymore. It is also restarted automatically. 0x5: DPLL3 is automatically put in idle bypass low power mode when the CORE clock is not required anymore. It is also restarted automatically.	RW	0x0

4.12.1.7.6 CM_AUTOIDLE2_PLL

Table 4-127. CM_AUTOIDLE2_PLL

Address Offset	0x0000 0034		
Physical Address	0x4800 4D34	**Instance**	Clock_Control_Reg_CM
Description	This register provides automatic control over the DPLL5 activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														AUTO_PERIPH2_DPLL	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2:0	AUTO_PERIPH2_DPLL	DPLL5 automatic control; Other enums: Reserved 0x0: Auto control disabled 0x1: DPLL5 is automatically put in low power stop mode when the 120 MHz clock is not required anymore. It is also restarted automatically.	RW	0x0

4.12.1.7.7 CM_CLKSEL1_PLL

Table 4-128. CM_CLKSEL1_PLL

Address Offset	0x0000 0040		
Physical Address	0x4800 4D40	**Instance**	Clock_Control_Reg_CM
Description	This register controls the selection of the master clock frequencies.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORE_DPLL_CLKOUT_DIV					CORE_DPLL_MULT											RESERVED	CORE_DPLL_DIV							RESERVED	SOURCE_96M	SOURCE_54M	RESERVED	SOURCE_48M	RESERVED		

Bits	Field Name	Description	Type	Reset
31:27	CORE_DPLL_CLKOUT_DIV	DPLL3 output clock divider factor M2; Other enums: Reserved 0x1: DPLL3 output clock is divided by 1 0x2: DPLL3 output clock is divided by 2 0x3: DPLL3 output clock is divided by 3 0x4: DPLL3 output clock is divided by 4 0x5: DPLL3 output clock is divided by 5 0x6: DPLL3 output clock is divided by 6 0x7: DPLL3 output clock is divided by 7 0x8: DPLL3 output clock is divided by 8 0x9: DPLL3 output clock is divided by 9 0xA: DPLL3 output clock is divided by 10 0xB: DPLL3 output clock is divided by 11 0xC: DPLL3 output clock is divided by 12 0xD: DPLL3 output clock is divided by 13 0xE: DPLL3 output clock is divided by 14 0xF: DPLL3 output clock is divided by 15 0x10: DPLL3 output clock is divided by 16 0x11: DPLL3 output clock is divided by 17 0x12: DPLL3 output clock is divided by 18 0x13: DPLL3 output clock is divided by 19 0x14: DPLL3 output clock is divided by 20 0x15: DPLL3 output clock is divided by 21 0x16: DPLL3 output clock is divided by 22 0x17: DPLL3 output clock is divided by 23 0x18: DPLL3 output clock is divided by 24 0x19: DPLL3 output clock is divided by 25 0x1A: DPLL3 output clock is divided by 26 0x1B: DPLL3 output clock is divided by 27 0x1C: DPLL3 output clock is divided by 28 0x1D: DPLL3 output clock is divided by 29 0x1E: DPLL3 output clock is divided by 30 0x1F: DPLL3 output clock is divided by 31	RW	0x01
26:16	CORE_DPLL_MULT	DPLL3 multiplier factor (0 to 2047)	RW	0x000
15	RESERVED	Reserved: keep at reset value.	R	0x0
14:8	CORE_DPLL_DIV	DPLL3 divider factor (0 to 127)	RW	0x00
7	RESERVED	Reserved: keep at reset value.	R	0x0
6	SOURCE_96M	Selection of 96M_FCLK source 0x0: source is the CM_96M_FCLK 0x1: source is CM_SYS_CLK	RW	0x1
5	SOURCE_54M	Selection of 54MHz clock source 0x0: source is the DPLL4_M3X2_CLK 0x1: source is sys_altclk	RW	0x0
4	RESERVED	Reserved: keep at reset value.	R	0x0
3	SOURCE_48M	Selection of Func_12M_clk and Func_48M_clk source 0x0: source is the CM_96M_FCLK 0x1: source is sys_altclk	RW	0x0
2:0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.1.7.8 CM_CLKSEL2_PLL

Table 4-129. CM_CLKSEL2_PLL

Address Offset	0x0000 0044		
Physical Address	0x4800 4D44	**Instance**	Clock_Control_Reg_CM
Description	This register controls the selection of the master clock frequencies.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								PERIPH_DPLL_MULT																RESERVED	PERIPH_DPLL_DIV						

Bits	Field Name	Description	Type	Reset
31:19	RESERVED	Reserved: keep at reset value.	R	0x0000
18:8	PERIPH_DPLL_MULT	DPLL4 multiplier factor (0 to 2047)	RW	0x000
7	RESERVED	Reserved: keep at reset value.	R	0x0
6:0	PERIPH_DPLL_DIV	DPLL4 divider factor (0 to 127)	RW	0x00

4.12.1.7.9 CM_CLKSEL3_PLL

Table 4-130. CM_CLKSEL3_PLL

Address Offset	0x0000 0048		
Physical Address	0x4800 4D48	**Instance**	Clock_Control_Reg_CM
Description	This register controls the selection of the master clock frequencies.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DIV_96M							

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4:0	DIV_96M	96 MHz clock divider factor M2 (1 up to 16); Other enums: Reserved 0x1: 96 MHz clock is DPLL4 clock divided by 1 0x2: 96 MHz clock is DPLL4 clock divided by 2 0x3: 96 MHz clock is DPLL4 clock divided by 3 0x4: 96 MHz clock is DPLL4 clock divided by 4 0x5: 96 MHz clock is DPLL4 clock divided by 5 0x6: 96 MHz clock is DPLL4 clock divided by 6 0x7: 96 MHz clock is DPLL4 clock divided by 7 0x8: 96 MHz clock is DPLL4 clock divided by 8 0x9: 96 MHz clock is DPLL4 clock divided by 9 0xA: 96 MHz clock is DPLL4 clock divided by 10 0xB: 96 MHz clock is DPLL4 clock divided by 11 0xC: 96 MHz clock is DPLL4 clock divided by 12 0xD: 96 MHz clock is DPLL4 clock divided by 13 0xE: 96 MHz clock is DPLL4 clock divided by 14 0xF: 96 MHz clock is DPLL4 clock divided by 15 0x10: 96 MHz clock is DPLL4 clock divided by 16	RW	0x01

4.12.1.7.10 CM_CLKSEL4_PLL

Table 4-131. CM_CLKSEL4_PLL

Address Offset	0x0000 004C		
Physical Address	0x4800 4D4C	**Instance**	Clock_Control_Reg_CM
Description	This register controls the selection of the master clock frequencies.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													PERIPH2_DPLL_MULT											RESERVED	PERIPH2_DPLL_DIV						

Bits	Field Name	Description	Type	Reset
31:19	RESERVED	Reserved: keep at reset value.	R	0x0000
18:8	PERIPH2_DPLL_MULT	DPLL5 multiplier factor (0 to 2047)	RW	0x000
7	RESERVED	Reserved: keep at reset value.	R	0x0
6:0	PERIPH2_DPLL_DIV	DPLL5 divider factor (0 to 127)	RW	0x00

4.12.1.7.11 CM_CLKSEL5_PLL

Table 4-132. CM_CLKSEL5_PLL

Address Offset	0x0000 0050		
Physical Address	0x4800 4D50	**Instance**	Clock_Control_Reg_CM
Description	This register controls the selection of the master clock frequencies.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DIV_120M							

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4:0	DIV_120M	120 MHz clock divider factor M2 (1 up to 16); Other enums: Reserved 0x1: 120 MHz clock is DPLL5 clock divided by 1 0x2: 120 MHz clock is DPLL5 clock divided by 2 0x3: 120 MHz clock is DPLL5 clock divided by 3 0x4: 120 MHz clock is DPLL5 clock divided by 4 0x5: 120 MHz clock is DPLL5 clock divided by 5 0x6: 120 MHz clock is DPLL5 clock divided by 6 0x7: 120 MHz clock is DPLL5 clock divided by 7 0x8: 120 MHz clock is DPLL5 clock divided by 8 0x9: 120 MHz clock is DPLL5 clock divided by 9 0xA: 120 MHz clock is DPLL5 clock divided by 10 0xB: 120 MHz clock is DPLL5 clock divided by 11 0xC: 120 MHz clock is DPLL5 clock divided by 12 0xD: 120 MHz clock is DPLL5 clock divided by 13 0xE: 120 MHz clock is DPLL5 clock divided by 14 0xF: 120 MHz clock is DPLL5 clock divided by 15 0x10: 120 MHz clock is DPLL5 clock divided by 16	RW	0x01

4.12.1.7.12 CM_CLKOUT_CTRL

Table 4-133. CM_CLKOUT_CTRL

Address Offset	0x0000 0070		
Physical Address	0x4800 4D70	**Instance**	Clock_Control_Reg_CM
Description	This register provides control over the SYS_CLKOUT2 output clock.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CLKOUT2_EN	RESERVED	CLKOUT2_DIV			RESERVED	CLKOUT2SOURCE	

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7	CLKOUT2_EN	This bit controls the external output clock activity 0x0: sys_clkout2 is disabled 0x1: sys_clkout2 is enabled	RW	0x0
6	RESERVED	Reserved: keep at reset value.	R	0x0
5:3	CLKOUT2_DIV	This field controls the external output clock division; Other enums: Reserved 0x0: sys_clkout2 / 1 0x1: sys_clkout2 / 2 0x2: sys_clkout2 / 4 0x3: sys_clkout2 / 8 0x4: sys_clkout2 / 16	RW	0x0
2	RESERVED	Reserved: keep at reset value.	R	0x0
1:0	CLKOUT2SOURCE	This field selects the external output clock source 0x0: source is CORE_CLK 0x1: source is CM_SYS_CLK 0x2: source is CM_96M_FCLK 0x3: source is 54 MHz clock	RW	0x3

4.12.1.8 DSS_CM Register Descriptions

4.12.1.8.1 CM_FCLKEN_DSS

Table 4-134. CM_FCLKEN_DSS

Address Offset	0x0000 0000		
Physical Address	0x4800 4E00	**Instance**	DSS_CM
Description	Controls the modules functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													EN_TV	EN_DSS2	EN_DSS1

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	EN_TV	DSS_TV_FCLK functional clock control 0x0: DSS_TV_FCLK is disabled 0x1: DSS_TV_FCLK is enabled	RW	0x0
1	EN_DSS2	Display Sub-System functional clock 2 control 0x0: DSS2_ALWON_FCLK is disabled 0x1: DSS2_ALWON_FCLK is enabled	RW	0x0
0	EN_DSS1	Display Sub-System functional clock 1 control 0x0: DSS1_ALWON_FCLK is disabled 0x1: DSS1_ALWON_FCLK is enabled	RW	0x0

4.12.1.8.2 CM_ICLKEN_DSS

Table 4-135. CM_ICLKEN_DSS

Address Offset	0x0000 0010		
Physical Address	0x4800 4E10	**Instance**	DSS_CM
Description	Controls the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															EN_DSS

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	EN_DSS	Display sub-system interface clock control 0x0: DSS_L3_ICLK and DSS_L4_ICLK are disabled 0x1: DSS_L3_ICLK and DSS_L4_ICLK are enabled	RW	0x0

4.12.1.8.3 CM_IDLEST_DSS

Table 4-136. CM_IDLEST_DSS

Address Offset	0x0000 0020		
Physical Address	0x4800 4E20	**Instance**	DSS_CM
Description	Modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														ST_DSS_IDLE	ST_DSS_STDBY

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved	R	0x00000000
1	ST_DSS_IDLE	Display Sub-System idle status. 0x0: Display Sub-System is active. 0x1: Display Sub-System is in idle mode and cannot be accessed.	R	0x1
0	ST_DSS_STDBY	Display Sub-System standby status. 0x0: Display Sub-System is active. 0x1: Display Sub-System is in standby mode.	R	0x1

4.12.1.8.4 CM_AUTOIDLE_DSS

Table 4-137. CM_AUTOIDLE_DSS

Address Offset	0x0000 0030		
Physical Address	0x4800 4E30	**Instance**	DSS_CM
Description	This register controls the automatic control of the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															AUTO_DSS

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	AUTO_DSS	Display Sub-System auto clock control. 0x0: Display Sub-System interface clock is unrelated to the domain state transition. 0x1: Display Sub-System interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0

4.12.1.8.5 CM_CLKSEL_DSS

Table 4-138. CM_CLKSEL_DSS

Address Offset	0x0000 0040		
Physical Address	0x4800 4E40	**Instance**	DSS_CM
Description	Modules clock selection.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																			CLKSEL_TV					RESERVED				CLKSEL_DSS1			

Bits	Field Name	Description	Type	Reset
31:13	RESERVED	Reserved: keep at reset value.	R	0x00000
12:8	CLKSEL_TV	TV functional clock divider factor DPLL4 M3 (1 up to 16); Other enums: Reserved 0x1: TV functional clock is DPLL4 clock divided by 1 0x2: TV functional clock is DPLL4 clock divided by 2 0x3: TV functional clock is DPLL4 clock divided by 3 0x4: TV functional clock is DPLL4 clock divided by 4 0x5: TV functional clock is DPLL4 clock divided by 5 0x6: TV functional clock is DPLL4 clock divided by 6 0x7: TV functional clock is DPLL4 clock divided by 7 0x8: TV functional clock is DPLL4 clock divided by 8 0x9: TV functional clock is DPLL4 clock divided by 9 0xA: TV functional clock is DPLL4 clock divided by 10 0xB: TV functional clock is DPLL4 clock divided by 11 0xC: TV functional clock is DPLL4 clock divided by 12 0xD: TV functional clock is DPLL4 clock divided by 13 0xE: TV functional clock is DPLL4 clock divided by 14 0xF: TV functional clock is DPLL4 clock divided by 15 0x10: TV functional clock is DPLL4 clock divided by 16	RW	0x10
7:5	RESERVED	Reserved: keep at reset value.	R	0x0

Bits	Field Name	Description	Type	Reset
4:0	CLKSEL_DSS1	DPLL4 M4 divide factor for DSS1_ALWON_FCLK (1 up to 16); Other enums: Reserved 0x1: DSS1_ALWON_FCLK is DPLL4 clock divided by 1 0x2: DSS1_ALWON_FCLK is DPLL4 clock divided by 2 0x3: DSS1_ALWON_FCLK is DPLL4 clock divided by 3 0x4: DSS1_ALWON_FCLK is DPLL4 clock divided by 4 0x5: DSS1_ALWON_FCLK is DPLL4 clock divided by 5 0x6: DSS1_ALWON_FCLK is DPLL4 clock divided by 6 0x7: DSS1_ALWON_FCLK is DPLL4 clock divided by 7 0x8: DSS1_ALWON_FCLK is DPLL4 clock divided by 8 0x9: DSS1_ALWON_FCLK is DPLL4 clock divided by 9 0xA: DSS1_ALWON_FCLK is DPLL4 clock divided by 10 0xB: DSS1_ALWON_FCLK is DPLL4 clock divided by 11 0xC: DSS1_ALWON_FCLK is DPLL4 clock divided by 12 0xD: DSS1_ALWON_FCLK is DPLL4 clock divided by 13 0xE: DSS1_ALWON_FCLK is DPLL4 clock divided by 14 0xF: DSS1_ALWON_FCLK is DPLL4 clock divided by 15 0x10: DSS1_ALWON_FCLK is DPLL4 clock divided by 16	RW	0x10

4.12.1.8.6 CM_SLEEPDEP_DSS

Table 4-139. CM_SLEEPDEP_DSS

Address Offset	0x0000 0044		
Physical Address	0x4800 4E44	**Instance**	DSS_CM
Description	This register allows enabling or disabling the sleep transition dependency of DSS domain with respect to other domain.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_MPU	EN_CORE

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_MPU	MPU domain dependency 0x0: DSS domain sleep dependency with MPU domain is disabled. 0x1: DSS domain sleep dependency with MPU domain is enabled.	RW	0x0
0	EN_CORE	CORE domain dependency.	RW	0x0

4.12.1.8.7 CM_CLKSTCTRL_DSS

Table 4-140. CM_CLKSTCTRL_DSS

Address Offset	0x0000 0048		
Physical Address	0x4800 4E48	**Instance**	DSS_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		

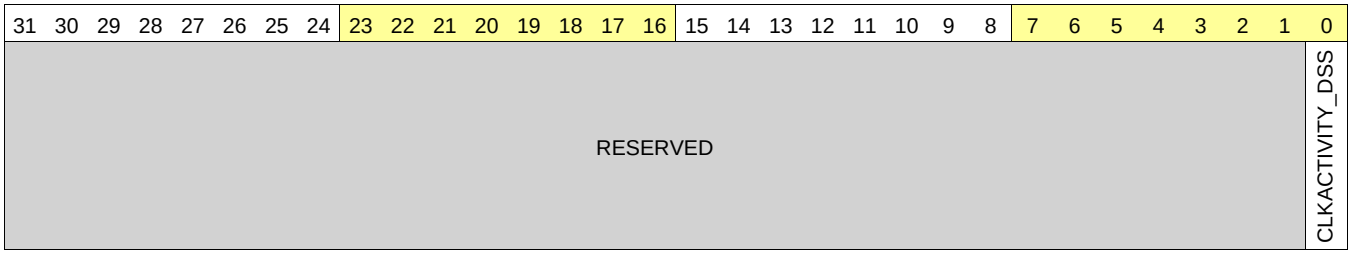
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														CLKTRCTRL_DSS	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_DSS	Controls the clock state transition of the DSS clock domain. 0x0: Automatic transition is disabled 0x1: Start a software supervised sleep transition on the domain 0x2: Start a software supervised wake-up transition on the domain 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0

4.12.1.8.8 CM_CLKSTST_DSS

Table 4-141. CM_CLKSTST_DSS

Address Offset	0x0000 004C		
Physical Address	0x4800 4E4C	**Instance**	DSS_CM
Description	This register provides a status on the OCP interface clock activity in the domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	CLKACTIVITY_DSS	Interface clock activity status 0x0: No domain Interface clock activity 0x1: Domain Interface clock is active	R	0x0

4.12.1.9 PER_CM Register Descriptions

4.12.1.9.1 CM_FCLKEN_PER

Table 4-142. CM_FCLKEN_PER

Address Offset	0x0000 0000		
Physical Address	0x4800 5000	**Instance**	PER_CM
Description	Controls the modules functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														EN_GPIO6	EN_GPIO5	EN_GPIO4	EN_GPIO3	EN_GPIO2	EN_WDT3	EN_UART3	EN_GPT9	EN_GPT8	EN_GPT7	EN_GPT6	EN_GPT5	EN_GPT4	EN_GPT3	EN_GPT2	EN_MCBSP4	EN_MCBSP3	EN_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17	EN_GPIO6	GPIO 6 functional clock control 0x0: GPIO 6 functional clock is disabled 0x1: GPIO 6 functional clock is enabled	RW	0x0
16	EN_GPIO5	GPIO 5 functional clock control 0x0: GPIO 5 functional clock is disabled 0x1: GPIO 5 functional clock is enabled	RW	0x0
15	EN_GPIO4	GPIO 4 functional clock control 0x0: GPIO 4 functional clock is disabled 0x1: GPIO 4 functional clock is enabled	RW	0x0
14	EN_GPIO3	GPIO 3 functional clock control 0x0: GPIO 3 functional clock is disabled 0x1: GPIO 3 functional clock is enabled	RW	0x0
13	EN_GPIO2	GPIO 2 functional clock control 0x0: GPIO 2 functional clock is disabled 0x1: GPIO 2 functional clock is enabled	RW	0x0
12	EN_WDT3	WDTIMER 3 functional clock control. 0x0: WDTIMER 3 functional clock is disabled 0x1: WDTIMER 3 functional clock is enabled	RW	0x0
11	EN_UART3	UART3 functional clock control. 0x0: UART 3 functional clock is disabled 0x1: UART 3 functional clock is enabled	RW	0x0
10	EN_GPT9	GPTIMER 9 functional clock control. 0x0: GPTIMER 9 functional clock is disabled 0x1: GPTIMER 9 functional clock is enabled	RW	0x0
9	EN_GPT8	GPTIMER 8 functional clock control. 0x0: GPTIMER 8 functional clock is disabled 0x1: GPTIMER 8 functional clock is enabled	RW	0x0
8	EN_GPT7	GPTIMER 7 functional clock control. 0x0: GPTIMER 7 functional clock is disabled 0x1: GPTIMER 7 functional clock is enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
7	EN_GPT6	GPTIMER 6 functional clock control. 0x0: GPTIMER 6 functional clock is disabled 0x1: GPTIMER 6 functional clock is enabled	RW	0x0
6	EN_GPT5	GPTIMER 5 functional clock control. 0x0: GPTIMER 5 functional clock is disabled 0x1: GPTIMER 5 functional clock is enabled	RW	0x0
5	EN_GPT4	GPTIMER 4 functional clock control. 0x0: GPTIMER 4 functional clock is disabled 0x1: GPTIMER 4 functional clock is enabled	RW	0x0
4	EN_GPT3	GPTIMER 3 functional clock control. 0x0: GPTIMER 3 functional clock is disabled 0x1: GPTIMER 3 functional clock is enabled	RW	0x0
3	EN_GPT2	GPTIMER 2 functional clock control. 0x0: GPTIMER 2 functional clock is disabled 0x1: GPTIMER 2 functional clock is enabled	RW	0x0
2	EN_MCBSP4	McBSP 4 functional clock control. 0x0: McBSP 4 functional clock is disabled 0x1: McBSP 4 functional clock is enabled	RW	0x0
1	EN_MCBSP3	McBSP3 functional clock control. 0x0: McBSP 3 functional clock is disabled 0x1: McBSP 3 functional clock is enabled	RW	0x0
0	EN_MCBSP2	McBSP 2 functional clock control. 0x0: McBSP 2 functional clock is disabled 0x1: McBSP 2 functional clock is enabled	RW	0x0

4.12.1.9.2 CM_ICLKEN_PER

Table 4-143. CM_ICLKEN_PER

Address Offset	0x0000 0010		
Physical Address	0x4800 5010	**Instance**	PER_CM
Description	Controls the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														EN_GPIO6	EN_GPIO5	EN_GPIO4	EN_GPIO3	EN_GPIO2	EN_WDT3	EN_UART3	EN_GPT9	EN_GPT8	EN_GPT7	EN_GPT6	EN_GPT5	EN_GPT4	EN_GPT3	EN_GPT2	EN_MCBSP4	EN_MCBSP3	EN_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17	EN_GPIO6	GPIO 6 interface clock control 0x0: GPIO 6 interface clock is disabled 0x1: GPIO 6 interface clock is enabled	RW	0x0
16	EN_GPIO5	GPIO 5 interface clock control 0x0: GPIO 5 interface clock is disabled 0x1: GPIO 5 interface clock is enabled	RW	0x0
15	EN_GPIO4	GPIO 4 interface clock control 0x0: GPIO 4 interface clock is disabled 0x1: GPIO 4 interface clock is enabled	RW	0x0
14	EN_GPIO3	GPIO 3 interface clock control 0x0: GPIO 3 interface clock is disabled 0x1: GPIO 3 interface clock is enabled	RW	0x0
13	EN_GPIO2	GPIO 2 interface clock control 0x0: GPIO 2 interface clock is disabled 0x1: GPIO 2 interface clock is enabled	RW	0x0
12	EN_WDT3	WDTIMER 3 interface clock control. 0x0: WDTIMER 3 interface clock is disabled 0x1: WDTIMER 3 interface clock is enabled	RW	0x0
11	EN_UART3	UART3 interface clock control. 0x0: UART 3 interface clock is disabled 0x1: UART 3 interface clock is enabled	RW	0x0
10	EN_GPT9	GPTIMER 9 interface clock control. 0x0: GPTIMER 9 interface clock is disabled 0x1: GPTIMER 9 interface clock is enabled	RW	0x0
9	EN_GPT8	GPTIMER 8 interface clock control. 0x0: GPTIMER 8 interface clock is disabled 0x1: GPTIMER 8 interface clock is enabled	RW	0x0
8	EN_GPT7	GPTIMER 7 interface clock control. 0x0: GPTIMER 7 interface clock is disabled 0x1: GPTIMER 7 interface clock is enabled	RW	0x0
7	EN_GPT6	GPTIMER 6 interface clock control. 0x0: GPTIMER 6 interface clock is disabled 0x1: GPTIMER 6 interface clock is enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
6	EN_GPT5	GPTIMER 5 interface clock control. 0x0: GPTIMER 5 interface clock is disabled 0x1: GPTIMER 5 interface clock is enabled	RW	0x0
5	EN_GPT4	GPTIMER 4 interface clock control. 0x0: GPTIMER 4 interface clock is disabled 0x1: GPTIMER 4 interface clock is enabled	RW	0x0
4	EN_GPT3	GPTIMER 3 interface clock control. 0x0: GPTIMER 3 interface clock is disabled 0x1: GPTIMER 3 interface clock is enabled	RW	0x0
3	EN_GPT2	GPTIMER 2 interface clock control. 0x0: GPTIMER 2 interface clock is disabled 0x1: GPTIMER 2 interface clock is enabled	RW	0x0
2	EN_MCBSP4	McBSP 4 interface clock control. 0x0: McBSP 4 interface clock is disabled 0x1: McBSP 4 interface clock is enabled	RW	0x0
1	EN_MCBSP3	McBSP 3 interface clock control. 0x0: McBSP 3 interface clock is disabled 0x1: McBSP 3 interface clock is enabled	RW	0x0
0	EN_MCBSP2	McBSP 2 interface clock control. 0x0: McBSP 2 interface clock is disabled 0x1: McBSP 2 interface clock is enabled	RW	0x0

4.12.1.9.3 CM_IDLEST_PER

Table 4-144. CM_IDLEST_PER

Address Offset	0x0000 0020		
Physical Address	0x4800 5020	**Instance**	PER_CM
Description	Modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														ST_GPIO6	ST_GPIO5	ST_GPIO4	ST_GPIO3	ST_GPIO2	ST_WDT3	ST_UART3	ST_GPT9	ST_GPT8	ST_GPT7	ST_GPT6	ST_GPT5	ST_GPT4	ST_GPT3	ST_GPT2	ST_MCBSP4	ST_MCBSP3	ST_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved	R	0x0000
17	ST_GPIO6	GPIO 6 idle status 0x0: GPIO 6 can be accessed. 0x1: GPIO 6 cannot be accessed. Any access may return an error.	R	0x1
16	ST_GPIO5	GPIO 5 idle status 0x0: GPIO 5 can be accessed. 0x1: GPIO 5 cannot be accessed. Any access may return an error.	R	0x1
15	ST_GPIO4	GPIO 4 idle status 0x0: GPIO 4 can be accessed. 0x1: GPIO 4 cannot be accessed. Any access may return an error.	R	0x1
14	ST_GPIO3	GPIO 3 idle status 0x0: GPIO 3 can be accessed. 0x1: GPIO 3 cannot be accessed. Any access may return an error.	R	0x1
13	ST_GPIO2	GPIO 2 idle status 0x0: GPIO 2 can be accessed. 0x1: GPIO 2 cannot be accessed. Any access may return an error.	R	0x1
12	ST_WDT3	WDTIMER 3 idle status. 0x0: WDTIMER 3 can be accessed. 0x1: WDTIMER 3 cannot be accessed. Any access may return an error.	R	0x1
11	ST_UART3	UART3 idle status. 0x0: UART 3 can be accessed. 0x1: UART 3 cannot be accessed. Any access may return an error.	R	0x1
10	ST_GPT9	GPTIMER 9 idle status. 0x0: GPTIMER 9 can be accessed. 0x1: GPTIMER 9 cannot be accessed. Any access may return an error.	R	0x1
9	ST_GPT8	GPTIMER 8 idle status. 0x0: GPTIMER 8 can be accessed. 0x1: GPTIMER 8 cannot be accessed. Any access may return an error.	R	0x1
8	ST_GPT7	GPTIMER 7 idle status. 0x0: GPTIMER 7 can be accessed. 0x1: GPTIMER 7 cannot be accessed. Any access may return an error.	R	0x1

Bits	Field Name	Description	Type	Reset
7	ST_GPT6	GPTIMER 6 idle status. 0x0: GPTIMER 6 can be accessed. 0x1: GPTIMER 6 cannot be accessed. Any access may return an error.	R	0x1
6	ST_GPT5	GPTIMER 5 idle status. 0x0: GPTIMER 5 can be accessed. 0x1: GPTIMER 5 cannot be accessed. Any access may return an error.	R	0x1
5	ST_GPT4	GPTIMER 4 idle status. 0x0: GPTIMER 4 can be accessed. 0x1: GPTIMER 4 cannot be accessed. Any access may return an error.	R	0x1
4	ST_GPT3	GPTIMER 3 idle status. 0x0: GPTIMER 3 can be accessed. 0x1: GPTIMER 3 cannot be accessed. Any access may return an error.	R	0x1
3	ST_GPT2	GPTIMER 2 idle status. 0x0: GPTIMER 2 can be accessed. 0x1: GPTIMER 2 cannot be accessed. Any access may return an error.	R	0x1
2	ST_MCBSP4	McBSP 4 idle status. 0x0: McBSP 4 can be accessed. 0x1: McBSP 4 cannot be accessed. Any access may return an error.	R	0x1
1	ST_MCBSP3	McBSP 3 idle status. 0x0: McBSP 3 can be accessed. 0x1: McBSP 3 cannot be accessed. Any access may return an error.	R	0x1
0	ST_MCBSP2	McBSP 2 idle status. 0x0: McBSP 2 can be accessed. 0x1: McBSP 2 cannot be accessed. Any access may return an error.	R	0x1

4.12.1.9.4 CM_AUTOIDLE_PER

Table 4-145. CM_AUTOIDLE_PER

Address Offset	0x0000 0030		
Physical Address	0x4800 5030	**Instance**	PER_CM
Description	This register controls the automatic control of the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														AUTO_GPIO6	AUTO_GPIO5	AUTO_GPIO4	AUTO_GPIO3	AUTO_GPIO2	AUTO_WDT3	AUTO_UART3	AUTO_GPT9	AUTO_GPT8	AUTO_GPT7	AUTO_GPT6	AUTO_GPT5	AUTO_GPT4	AUTO_GPT3	AUTO_GPT2	AUTO_MCBSP4	AUTO_MCBSP3	AUTO_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17	AUTO_GPIO6	GPIO 6 auto clock control 0x0: GPIO 6 interface clock is unrelated to the domain state transition. 0x1: GPIO 6 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
16	AUTO_GPIO5	GPIO 5 auto clock control 0x0: GPIO 5 interface clock is unrelated to the domain state transition. 0x1: GPIO 5 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
15	AUTO_GPIO4	GPIO 4 auto clock control 0x0: GPIO 4 interface clock is unrelated to the domain state transition. 0x1: GPIO 4 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
14	AUTO_GPIO3	GPIO 3 auto clock control 0x0: GPIO 3 interface clock is unrelated to the domain state transition. 0x1: GPIO 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
13	AUTO_GPIO2	GPIO 2 auto clock control 0x0: GPIO 2 interface clock is unrelated to the domain state transition. 0x1: GPIO 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
12	AUTO_WDT3	WDTIMER 3 auto clock control. 0x0: WDTIMER 3 interface clock is unrelated to the domain state transition. 0x1: WDTIMER 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
11	AUTO_UART3	UART3 auto clock control. 0x0: UART 3 interface clock is unrelated to the domain state transition. 0x1: UART 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0

Bits	Field Name	Description	Type	Reset
10	AUTO_GPT9	GPTIMER 9 auto clock control. 0x0: GPTIMER 9 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 9 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
9	AUTO_GPT8	GPTIMER 8 auto clock control. 0x0: GPTIMER 8 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 8 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
8	AUTO_GPT7	GPTIMER 7 auto clock control. 0x0: GPTIMER 7 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 7 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
7	AUTO_GPT6	GPTIMER 6 auto clock control. 0x0: GPTIMER 6 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 6 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
6	AUTO_GPT5	GPTIMER 5 auto clock control. 0x0: GPTIMER 5 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 5 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
5	AUTO_GPT4	GPTIMER 4 auto clock control. 0x0: GPTIMER 4 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 4 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
4	AUTO_GPT3	GPTIMER 3 auto clock control. 0x0: GPTIMER 3 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
3	AUTO_GPT2	GPTIMER 2 auto clock control. 0x0: GPTIMER 2 interface clock is unrelated to the domain state transition. 0x1: GPTIMER 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
2	AUTO_MCBSP4	McBSP 4 auto clock control. 0x0: McBSP 4 interface clock is unrelated to the domain state transition. 0x1: McBSP 4 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
1	AUTO_MCBSP3	McBSP 3 auto clock control. 0x0: McBSP 3 interface clock is unrelated to the domain state transition. 0x1: McBSP 3 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0
0	AUTO_MCBSP2	McBSP 2 auto clock control. 0x0: McBSP 2 interface clock is unrelated to the domain state transition. 0x1: McBSP 2 interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0

4.12.1.9.5 CM_CLKSEL_PER

Table 4-146. CM_CLKSEL_PER

Address Offset	0x0000 0040		
Physical Address	0x4800 5040	**Instance**	PER_CM
Description	PER domain modules source clock selection.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CLKSEL_GPT9	CLKSEL_GPT8	CLKSEL_GPT7	CLKSEL_GPT6	CLKSEL_GPT5	CLKSEL_GPT4	CLKSEL_GPT3	CLKSEL_GPT2

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7	CLKSEL_GPT9	Selects GPTIMER 9 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
6	CLKSEL_GPT8	Selects GPTIMER 8 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
5	CLKSEL_GPT7	Selects GPTIMER 7 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
4	CLKSEL_GPT6	Selects GPTIMER 6 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
3	CLKSEL_GPT5	Selects GPTIMER 5 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
2	CLKSEL_GPT4	Selects GPTIMER 4 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
1	CLKSEL_GPT3	Selects GPTIMER 3 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0
0	CLKSEL_GPT2	Selects GPTIMER 2 source clock 0x0: source is 32K_FCLK 0x1: source is SYS_CLK	RW	0x0

4.12.1.9.6 CM_SLEEPDEP_PER

Table 4-147. CM_SLEEPDEP_PER

Address Offset	0x0000 0044		
Physical Address	0x4800 5044	**Instance**	PER_CM
Description	This register allows enabling or disabling the sleep transition dependency of PER domain with respect to other domain.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_MPU	RESERVED

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_MPU	MPU domain dependency	RW	0x0
		0x0: PER domain sleep dependency with MPU domain is disabled.		
		0x1: PER domain sleep dependency with MPU domain is enabled.		
0	RESERVED	Reserved: keep at reset value.	RW	0x0

4.12.1.9.7 CM_CLKSTCTRL_PER

Table 4-148. CM_CLKSTCTRL_PER

Address Offset	0x0000 0048		
Physical Address	0x4800 5048	**Instance**	PER_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		

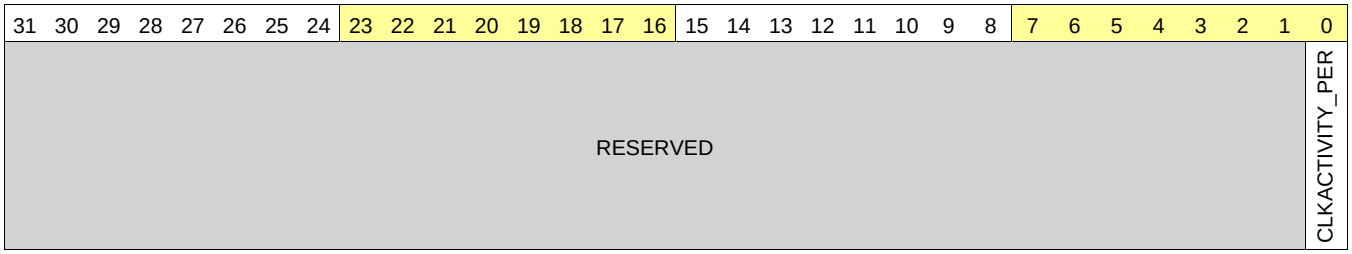
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														CLKTRCTRL_PER	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_PER	Controls the clock state transition of the PERIPHERAL clock domain. 0x0: Automatic transition is disabled 0x1: Start a software supervised sleep transition on the domain 0x2: Start a software supervised wake-up transition on the domain 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0

4.12.1.9.8 CM_CLKSTST_PER

Table 4-149. CM_CLKSTST_PER

Address Offset	0x0000 004C		
Physical Address	0x4800 504C	**Instance**	PER_CM
Description	This register provides a status on the OCP interface clock activity in the domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	CLKACTIVITY_PER	Interface clock activity status 0x0: No domain interface clock activity 0x1: Domain interface clock is active	R	0x0

4.12.1.10 EMU_CM Register Descriptions

4.12.1.10.1 CM_CLKSEL1_EMU

Table 4-150. CM_CLKSEL1_EMU

Address Offset	0x0000 0040		
Physical Address	0x4800 5140	**Instance**	EMU_CM
Description	Modules clock selection.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			DIV_DPLL4					RESERVED			DIV_DPLL3					RESERVED		CLKSEL_TRACECLK			CLKSEL_PCLK			CLKSEL_PCLKX2		CLKSEL_ATCLK		TRACE_MUX_CTRL		MUX_CTRL	

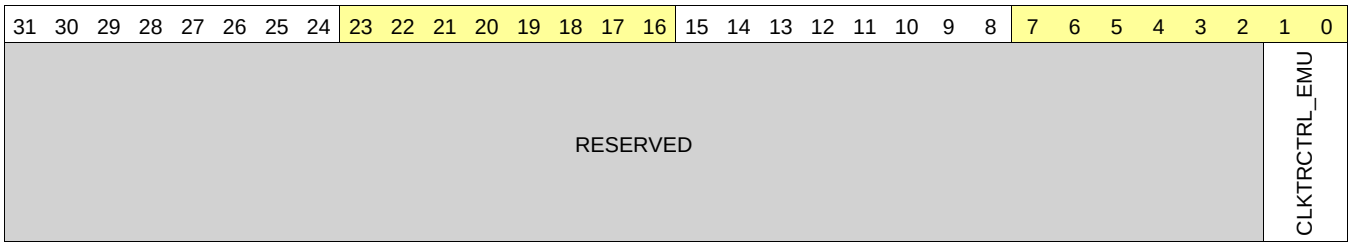
Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Reserved: keep at reset value.	R	0x0
28:24	DIV_DPLL4	DPLL4 M6 clock divider factor (1 up to 16); Other enums: Reserved 0x1: EMU_PER_ALWON_CLK is DPLL4 clock divided by 1 0x2: EMU_PER_ALWON_CLK is DPLL4 clock divided by 2 0x3: EMU_PER_ALWON_CLK is DPLL4 clock divided by 3 0x4: EMU_PER_ALWON_CLK is DPLL4 clock divided by 4 0x5: EMU_PER_ALWON_CLK is DPLL4 clock divided by 5 0x6: EMU_PER_ALWON_CLK is DPLL4 clock divided by 6 0x7: EMU_PER_ALWON_CLK is DPLL4 clock divided by 7 0x8: EMU_PER_ALWON_CLK is DPLL4 clock divided by 8 0x9: EMU_PER_ALWON_CLK is DPLL4 clock divided by 9 0xA: EMU_PER_ALWON_CLK is DPLL4 clock divided by 10 0xB: EMU_PER_ALWON_CLK is DPLL4 clock divided by 11 0xC: EMU_PER_ALWON_CLK is DPLL4 clock divided by 12 0xD: EMU_PER_ALWON_CLK is DPLL4 clock divided by 13 0xE: EMU_PER_ALWON_CLK is DPLL4 clock divided by 14 0xF: EMU_PER_ALWON_CLK is DPLL4 clock divided by 15 0x10: EMU_PER_ALWON_CLK is DPLL4 clock divided by 16	RW	0x10
23:21	RESERVED	Reserved: keep at reset value.	R	0x0

Bits	Field Name	Description	Type	Reset
20:16	DIV_DPLL3	DPLL3_M3X2 clock divider factor (1 up to 16); Other enums: Reserved 0x1: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 1 0x2: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 2 0x3: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 3 0x4: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 4 0x5: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 5 0x6: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 6 0x7: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 7 0x8: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 8 0x9: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 9 0xA: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 10 0xB: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 11 0xC: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 12 0xD: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 13 0xE: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 14 0xF: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 15 0x10: EMU_CORE_ALWON_CLK is DPLL3 clock divided by 16	RW	0x10
15:14	RESERVED	Reserved: keep at reset value.	R	0x0
13:11	CLKSEL_TRACECLK	Selects the TRACE clock; Other enums: Reserved 0x1: TRACECLK.FCLK is the selected TRACE source clock divided by 1 0x2: TRACECLK.FCLK is the selected TRACE source clock divided by 2 0x4: TRACECLK.FCLK is the selected TRACE source clock divided by 4	RW	0x1
10:8	CLKSEL_PCLK	Selects the PCLK clock; Other enums: Reserved 0x2: PCLK.FCLK is the selected PCLK source clock divided by 2 0x3: PCLK.FCLK is the selected PCLK source clock divided by 3 0x4: PCLK.FCLK is the selected PCLK source clock divided by 4 0x6: PCLK.FCLK is the selected PCLK source clock divided by 6	RW	0x2
7:6	CLKSEL_PCLKX2	Selects the PCLKx2 clock; Other enums: Reserved 0x1: PCLKx2.FCLK is the selected PCLK source clock divided by 1 0x2: PCLKx2.FCLK is the selected PCLK source clock divided by 2 0x3: PCLKx2.FCLK is the selected PCLK source clock divided by 3	RW	0x1
5:4	CLKSEL_ATCLK	Selects the ATCLK clock; Other enums: Reserved 0x1: ATCLK.FCLK is the selected ATCLK source clock divided by 1 0x2: ATCLK.FCLK is the selected ATCLK source clock divided by 2	RW	0x1
3:2	TRACE_MUX_CTRL	Selection of TRACECLK.FCLK source clock 0x0: TRACE source clock is SYS_CLK 0x1: TRACE source clock is EMU_CORE_ALWON_CLK 0x2: TRACE source clock is EMU_PER_ALWON clock 0x3: TRACE source clock is EMU_MPU_ALWON clock	RW	0x0
1:0	MUX_CTRL	Selection of ATCLK.FCLK, PCLK.FCLK and PCLKx2.FCLK source clock 0x0: ATCLK, PCLK and PCLKx2 source clock is SYS_CLK 0x1: ATCLK, PCLK and PCLKx2 source clock is EMU_CORE_ALWON_CLK 0x2: ATCLK, PCLK and PCLKx2 source clock is EMU_PER_ALWON clock 0x3: ATCLK, PCLK and PCLKx2 source clock is EMU_MPU_ALWON_CLK	RW	0x0

4.12.1.10.2 CM_CLKSTCTRL_EMU

Table 4-151. CM_CLKSTCTRL_EMU

Address Offset	0x0000 0048		
Physical Address	0x4800 5148	**Instance**	EMU_CM
Description	This register allows to enable or disable software and hardware supervised transition between ACTIVE and INACTIVE states.		
Type	RW		

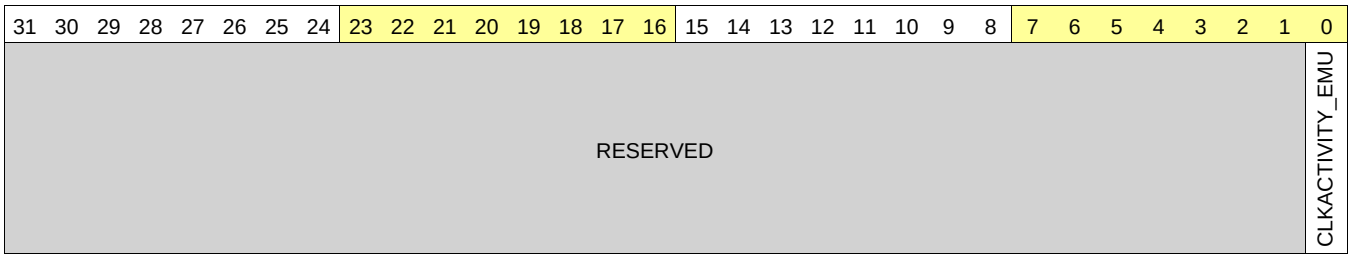


Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_EMU	Controls the clock state transition of the EMULATION clock domain. 0x0: Reserved 0x1: Start a software supervised sleep transition on the domain 0x2: Start a software supervised wake-up transition on the domain or maintain emulation domain active. (force wakeup has to be kept asserted to keep Emulation domain ON) 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x2

4.12.1.10.3 CM_CLKSTST_EMU

Table 4-152. CM_CLKSTST_EMU

Address Offset	0x0000 004C		
Physical Address	0x4800 514C	**Instance**	EMU_CM
Description	This register provides a status on the clock activity in the domain (depends on the selected source clock).		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	CLKACTIVITY_EMU	Clock activity status (depends on the selected source clock) 0x0: No domain clock activity 0x1: Domain clock is active	R	0x0

4.12.1.10.4 CM_CLKSEL2_EMU

Table 4-153. CM_CLKSEL2_EMU

Address Offset	0x0000 0050		
Physical Address	0x4800 5150	**Instance**	EMU_CM
Description	This register provides override controls over the DPLL3.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OVERRIDE_ENABLE	CORE_DPLL_EMU_MULT											RESERVED	CORE_DPLL_EMU_DIV						

Bits	Field Name	Description	Type	Reset
31:20	RESERVED	Reserved: keep at reset value.	R	0x000
19	OVERRIDE_ENABLE	This bit allows to enable or disable the emulation override controls 0x0: The emulation override controls are disabled 0x1: The emulation override controls are enabled	RW	0x0
18:8	CORE_DPLL_EMU_MULT	DPLL3 override multiplier factor (0 to 2047)	RW	0x000
7	RESERVED	Reserved: keep at reset value.	R	0x0
6:0	CORE_DPLL_EMU_DIV	DPLL3 override divider factor (0 to 127)	RW	0x00

4.12.1.10.5 CM_CLKSEL3_EMU

Table 4-154. CM_CLKSEL3_EMU

Address Offset	0x0000 0054		
Physical Address	0x4800 5154	**Instance**	EMU_CM
Description	This register provides override controls over the PERIPHERAL DPLL.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OVERRIDE_ENABLE	PERIPH_DPLL_EMU_MULT											RESERVED	PERIPH_DPLL_EMU_DIV						

Bits	Field Name	Description	Type	Reset
31:20	RESERVED	Reserved: keep at reset value.	R	0x000
19	OVERRIDE_ENABLE	This bit allows to enable or disable the emulation override controls 0x0: The emulation override controls are disabled 0x1: The emulation override controls are enabled	RW	0x0
18:8	PERIPH_DPLL_EMU_MULT	DPLL4 override multiplier factor (0 to 2047)	RW	0x000
7	RESERVED	Reserved: keep at reset value.	R	0x0
6:0	PERIPH_DPLL_EMU_DIV	DPLL4 override divider factor (0 to 127)	RW	0x00

4.12.1.11 Global_Reg_CM Register Descriptions

4.12.1.11.1 CM_POLCTRL

Table 4-155. CM_POLCTRL

Address Offset	0x0000 009C		
Physical Address	0x4800 529C	**Instance**	Global_Reg_CM
Description	This register allows setting the polarity of device outputs control signals.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															CLKOUT2_POL

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	CLKOUT2_POL	Controls the external output clock 2 polarity when disabled 0x0: sys_clkout2 is gated low when inactive 0x1: sys_clkout2 is gated high when inactive	RW	0x0

4.12.1.12 NEON_CM Register Descriptions

4.12.1.12.1 CM_IDLEST_NEON

Table 4-156. CM_IDLEST_NEON

Address Offset	0x0000 0020		
Physical Address	0x4800 5320	**Instance**	NEON_CM
Description	Modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

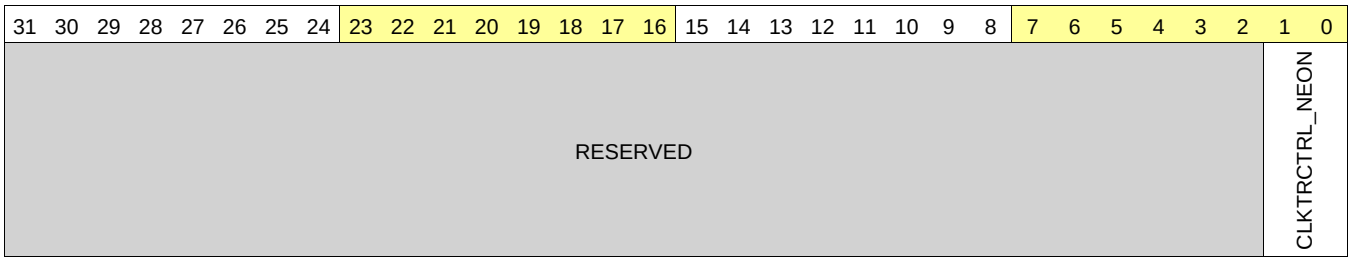
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															ST_NEON

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	ST_NEON	NEON standby status. 0x0: NEON is active 0x1: NEON is in standby mode	R	0x1

4.12.1.12.2 CM_CLKSTCTRL_NEON

Table 4-157. CM_CLKSTCTRL_NEON

Address Offset	0x0000 0048		
Physical Address	0x4800 5348	**Instance**	NEON_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_NEON	Controls the clock state transition of the NEON clock domain. 0x0: Automatic transition is disabled 0x1: Start a software supervised sleep transition on the domain 0x2: Start a software supervised wake-up transition on the domain 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0

4.12.1.13 USBHOST_CM Register Descriptions

4.12.1.13.1 CM_FCLKEN_USBHOST

Table 4-158. CM_FCLKEN_USBHOST

Address Offset	0x0000 0000		
Physical Address	0x4800 5400	**Instance**	USBHOST_CM
Description	Controls the modules functional clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_USBHOST2	EN_USBHOST1

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_USBHOST2	USB HOST 120-MHz functional clock control 0x0: USBHOST_120M_FCLK is disabled 0x1: USBHOST_120M_FCLK is enabled	RW	0x0
0	EN_USBHOST1	USB HOST 48-MHz functional clock control 0x0: USBHOST_48M_FCLK is disabled 0x1: USBHOST_48M_FCLK is enabled	RW	0x0

4.12.1.13.2 CM_ICLKEN_USBHOST

Table 4-159. CM_ICLKEN_USBHOST

Address Offset	0x0000 0010		
Physical Address	0x4800 5410	**Instance**	USBHOST_CM
Description	Controls the modules interface clock activity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															EN_USBHOST

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	EN_USBHOST	USB HOST interface clock control 0x0: USB HOST interface clock is disabled 0x1: USB HOST interface clock is enabled	RW	0x0

4.12.1.13.3 CM_IDLEST_USBHOST

Table 4-160. CM_IDLEST_USBHOST

Address Offset	0x0000 0020		
Physical Address	0x4800 5420	**Instance**	USBHOST_CM
Description	Modules access availability monitoring. This register is read only and automatically updated.		
Type	R		

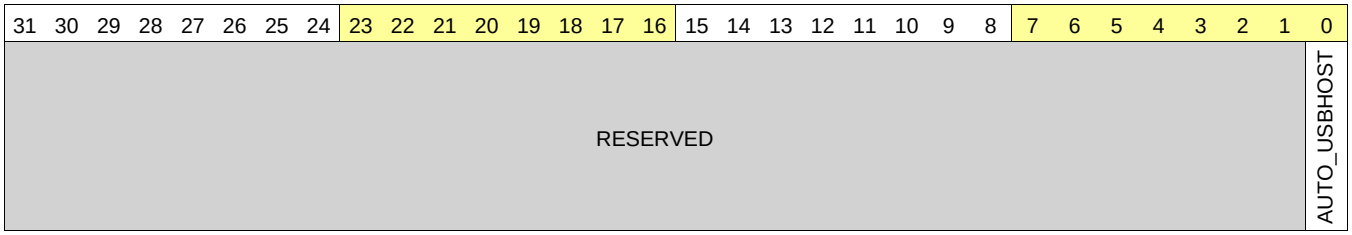
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														ST_USBHOST_IDLE	ST_USBHOST_STDBY

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved	R	0x00000000
1	ST_USBHOST_IDLE	USB HOST idle status. 0x0: USB HOST is active. 0x1: USB HOST is in idle mode and cannot be accessed.	R	0x1
0	ST_USBHOST_STDBY	USB HOST standby status. 0x0: USB HOST is active. 0x1: USB HOST is in standby mode.	R	0x1

4.12.1.13.4 CM_AUTOIDLE_USBHOST

Table 4-161. CM_AUTOIDLE_USBHOST

Address Offset	0x0000 0030		
Physical Address	0x4800 5430	**Instance**	USBHOST_CM
Description	This register controls the automatic control of the modules interface clock activity.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	AUTO_USBHOST	USB HOST auto clock control. 0x0: USB HOST interface clock is unrelated to the domain state transition. 0x1: USB HOST interface clock is automatically enabled or disabled along with the domain state transition.	RW	0x0

4.12.1.13.5 CM_SLEEPDEP_USBHOST

Table 4-162. CM_SLEEPDEP_USBHOST

Address Offset	0x0000 0044		
Physical Address	0x4800 5444	**Instance**	USBHOST_CM
Description	This register allows enabling or disabling the sleep transition dependency of USB HOST domain with respect to other domain.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_MPU	RESERVED

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_MPU	MPU domain dependency 0x0: USB HOST domain sleep dependency with MPU domain is disabled. 0x1: USB HOST domain sleep dependency with MPU domain is enabled.	RW	0x0
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.1.13.6 CM_CLKSTCTRL_USBHOST

Table 4-163. CM_CLKSTCTRL_USBHOST

Address Offset	0x0000 0048		
Physical Address	0x4800 5448	**Instance**	USBHOST_CM
Description	This register enables the domain power state transition. It controls the hardware supervised domain power state transition between ACTIVE and INACTIVE states.		
Type	RW		

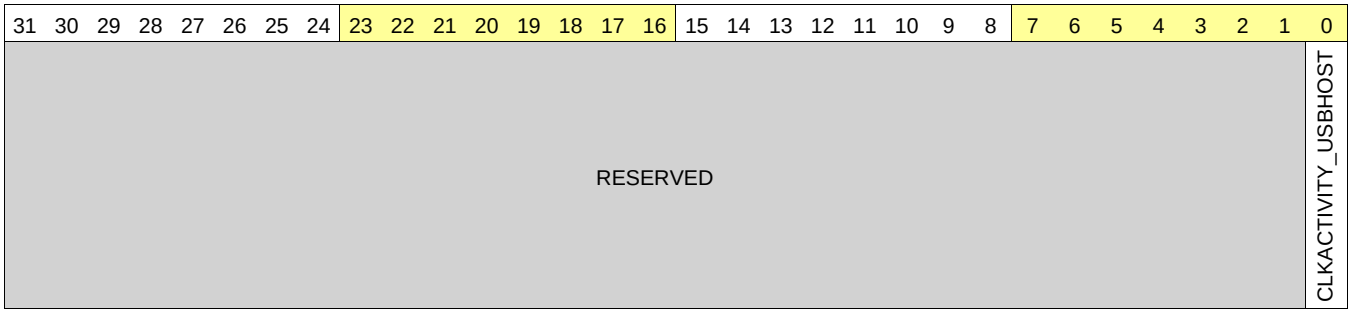
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														CLKTRCTRL_USBHOST	

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	CLKTRCTRL_USBHOST	Controls the clock state transition of the USB HOST clock domain. 0x0: Automatic transition is disabled 0x1: Start a software supervised sleep transition on the domain 0x2: Start a software supervised wake-up transition on the domain 0x3: Automatic transition is enabled. Transition is supervised by the hardware.	RW	0x0

4.12.1.13.7 CM_CLKSTST_USBHOST

Table 4-164. CM_CLKSTST_USBHOST

Address Offset	0x0000 004C		
Physical Address	0x4800 544C	**Instance**	USBHOST_CM
Description	This register provides a status on the interface clock activity in the domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved	R	0x00000000
0	CLKACTIVITY_USBHOST	Interface clock activity status 0x0: No domain interface clock activity 0x1: Domain interface clock is active	R	0x0

4.12.2 PRM Module Registers

This section describes the PRM module registers.Table 4-165 lists the physical address of the PRM modules. Table 4-166 through Table 4-170 provide register mapping summaries and describe the bits in the individual registers.

Table 4-165. PRM Instance Summary

Module Name	Base address (hex)	Size	Section
Reserved	0x4830 6000	8192 bytes	
OCP_System_Reg_PRM	0x4830 6800	8192 bytes	Section 4.12.2.2
MPU_PRM	0x4830 6900	8192 bytes	Section 4.12.2.3
CORE_PRM	0x4830 6A00	8192 bytes	Section 4.12.2.4
SGX_PRM	0x4830 6B00	8192 bytes	Section 4.12.2.5
WKUP_PRM	0x4830 6C00	8192 bytes	Section 4.12.2.6
Clock_Control_Reg_PRM	0x4830 6D00	8192 bytes	Section 4.12.2.7
DSS_PRM	0x4830 6E00	8192 bytes	Section 4.12.2.8
Reserved	0x4830 6F00	8192 bytes	
PER_PRM	0x4830 7000	8192 bytes	Section 4.12.2.9
EMU_PRM	0x4830 7100	8192 bytes	Section 4.12.2.10
Global_Reg_PRM	0x4830 7200	65536 bytes	Section 4.12.2.11
NEON_PRM	0x4830 7300	8192 bytes	Section 4.12.2.12
USBHOST_PRM	0x4830 7400	8192 bytes	Section 4.12.2.13

4.12.2.1 PRM Module Registers Mapping Summary

Table 4-166. OCP_System_Reg_PRM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
PRM_REVISION	R	32	0x0000 0004	0x4830 6804	C	Section 4.12.2.2.1
PRM_SYSCONFIG	RW	32	0x0000 0014	0x4830 6814	W	Section 4.12.2.2.2
PRM_IRQSTATUS_MPU	RW	32	0x0000 0018	0x4830 6818	W	Section 4.12.2.2.3
PRM_IRQENABLE_MPU	RW	32	0x0000 001C	0x4830 681C	W	Section 4.12.2.2.4

Table 4-167. MPU_PRM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_MPU	RW	32	0x0000 0058	0x4830 6958	C	Section 4.12.2.3.1
PM_WKDEP_MPU	RW	32	0x0000 00C8	0x4830 69C8	W	Section 4.12.2.3.2
PM_EVGENCTRL_MPU	RW	32	0x0000 00D4	0x4830 69D4	W	Section 4.12.2.3.3
PM_EVGENONTIM_MPU	RW	32	0x0000 00D8	0x4830 69D8	W	Section 4.12.2.3.4
PM_EVGENOFFTIM_MPU	RW	32	0x0000 00DC	0x4830 69DC	W	Section 9.2.6.2.1
PM_PWSTCTRL_MPU	RW	32	0x0000 00E0	0x4830 69E0	W	Section 4.12.2.3.6
PM_PWSTST_MPU	R	32	0x0000 00E4	0x4830 69E4	C	Section 4.12.2.3.7
PM_PREPWSTST_MPU	RW	32	0x0000 00E8	0x4830 69E8	C	Section 4.12.2.3.8

Table 4-168. CORE_PRM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_CORE	RW	32	0x0000 0058	0x4830 6A58	C	Section 4.12.2.4.1
PM_WKEN1_CORE	RW	32	0x0000 00A0	0x4830 6AA0	W	Section 4.12.2.4.2
PM_MPUGRPSEL1_CORE	RW	32	0x0000 00A4	0x4830 6AA4	W	Section 4.12.2.4.3
Reserved	RW	32	0x0000 00A8	0x4830 6AA8	W	
PM_WKST1_CORE	RW	32	0x0000 00B0	0x4830 6AB0	C	Section 4.12.2.4.4
PM_WKST3_CORE	RW	32	0x0000 00B8	0x4830 6AB8	C	Section 4.12.2.4.5
PM_PWSTCTRL_CORE	RW	32	0x0000 00E0	0x4830 6AE0	W	Section 4.12.2.4.6
PM_PWSTST_CORE	R	32	0x0000 00E4	0x4830 6AE4	C	Section 4.12.2.4.7
PM_PREPWSTST_CORE	RW	32	0x0000 00E8	0x4830 6AE8	C	Section 4.12.2.4.8
PM_WKEN3_CORE	RW	32	0x0000 00F0	0x4830 6AF0	W	Section 4.12.2.4.9
Reserved	RW	32	0x0000 00F4	0x4830 6AF4	W	
PM_MPUGRPSEL3_CORE	RW	32	0x0000 00F8	0x4830 6AF8	W	Section 4.12.2.4.10

Table 4-169. SGX_PRM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_SGX	RW	32	0x0000 0058	0x4830 6B58	C	Section 4.12.2.5.1
PM_WKDEP_SGX	RW	32	0x0000 00C8	0x4830 6BC8	W	Section 4.12.2.5.2
PM_PWSTCTRL_SGX	RW	32	0x0000 00E0	0x4830 6BE0	W	Section 4.12.2.5.3
PM_PWSTST_SGX	R	32	0x0000 00E4	0x4830 6BE4	C	Section 4.12.2.5.4
PM_PREPWSTCTRL_SGX	RW	32	0x0000 00E8	0x4830 6BE8	C	Section 4.12.2.5.5

Table 4-170. WKUP_PRM Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
PM_WKEN_WKUP	RW	32	0x0000 00A0	0x4830 6CA0	W	Section 4.12.2.6.1
PM_MPUGRPSEL_WKUP	RW	32	0x0000 00A4	0x4830 6CA4	W	Section 4.12.2.6.2
Reserved	RW	32	0x0000 00A8	0x4830 6CA8	W	
PM_WKST_WKUP	RW	32	0x0000 00B0	0x4830 6CB0	C	Section 4.12.2.6.3

Table 4-171. Clock_Control_Reg_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
PRM_CLKSEL	RW	32	0x0000 0040	0x4830 6D40	C	Section 4.12.2.7.1
PRM_CLKOUT_CTRL	RW	32	0x0000 0070	0x4830 6D70	C	Section 4.12.2.7.2

Table 4-172. DSS_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_DSS	RW	32	0x0000 0058	0x4830 6E58	C	Section 4.12.2.8.1
PM_WKEN_DSS	RW	32	0x0000 00A0	0x4830 6EA0	W	Section 4.12.2.8.2

Table 4-172. DSS_PRM Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
PM_WKDEP_DSS	RW	32	0x0000 00C8	0x4830 6EC8	W	Section 4.12.2.8.3
PM_PWSTCTRL_DSS	RW	32	0x0000 00E0	0x4830 6EE0	W	Section 4.12.2.8.4
PM_PWSTST_DSS	R	32	0x0000 00E4	0x4830 6EE4	C	Section 4.12.2.8.5
PM_PREPWSTST_DSS	RW	32	0x0000 00E8	0x4830 6EE8	C	Section 4.12.2.8.6

Table 4-173. PER_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_PER	RW	32	0x0000 0058	0x4830 7058	C	Section 4.12.2.9.1
PM_WKEN_PER	RW	32	0x0000 00A0	0x4830 70A0	W	Section 4.12.2.9.2
PM_MPUGRPSEL_PER	RW	32	0x0000 00A4	0x4830 70A4	W	Section 4.12.2.9.3
Reserved	RW	32	0x0000 00A8	0x4830 70A8	W	
PM_WKST_PER	RW	32	0x0000 00B0	0x4830 70B0	C	Section 4.12.2.9.4
PM_WKDEP_PER	RW	32	0x0000 00C8	0x4830 70C8	W	Section 4.12.2.9.5
PM_PWSTCTRL_PER	RW	32	0x0000 00E0	0x4830 70E0	W	Section 4.12.2.9.6
PM_PWSTST_PER	R	32	0x0000 00E4	0x4830 70E4	C	Section 4.12.2.9.7
PM_PREPWSTST_PER	RW	32	0x0000 00E8	0x4830 70E8	C	Section 4.12.2.9.8

Table 4-174. EMU_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_EMU	RW	32	0x0000 0058	0x4830 7158	C	Section 4.12.2.10.1
Reserved	RW	32	0x0000 00E4	0x4830 71E4	C	

Table 4-175. Global_Reg_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
Reserved	RW	32	0x0000 0020	0x4830 7220	W	
Reserved	RW	32	0x0000 0024	0x4830 7224	W	
Reserved	RW	32	0x0000 0028	0x4830 7228	W	
Reserved	RW	32	0x0000 002C	0x4830 722C	W	
Reserved	RW	32	0x0000 0030	0x4830 7230	W	
Reserved	RW	32	0x0000 0034	0x4830 7234	W	
Reserved	RW	32	0x0000 0038	0x4830 7238	W	
Reserved	RW	32	0x0000 003C	0x4830 723C	W	
PRM_RSTCTRL	RW	32	0x0000 0050	0x4830 7250	C	Section 4.12.2.11.1
PRM_RSTTIME	RW	32	0x0000 0054	0x4830 7254	C	Section 4.12.2.11.2
PRM_RSTST	RW	32	0x0000 0058	0x4830 7258	C	Section 4.12.2.11.3
Reserved	RW	32	0x0000 0060	0x4830 7260	W	
Reserved	RW	32	0x0000 0064	0x4830 7264	C	
PRM_CLKSRC_CTRL	RW	32	0x0000 0070	0x4830 7270	C	Section 4.12.2.11.4

Table 4-175. Global_Reg_PRM Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
PRM_OBS	R	32	0x0000 0080	0x4830 7280	C	Section 4.12.2.11.5
Reserved	RW	32	0x0000 0090	0x4830 7290	C	
Reserved	RW	32	0x0000 0094	0x4830 7294	C	
PRM_CLKSETUP	RW	32	0x0000 0098	0x4830 7298	C	Section 4.12.2.11.6
PRM_POLCTRL	RW	32	0x0000 009C	0x4830 729C	C	Section 4.12.2.11.7
Reserved	RW	32	0x0000 00A0	0x4830 72A0	C	

Table 4-176. NEON_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_NEON	RW	32	0x0000 0058	0x4830 7358	C	Section 4.12.2.12.1
PM_WKDEP_NEON	RW	32	0x0000 00C8	0x4830 73C8	W	Section 4.12.2.12.2
PM_PWSTCTRL_NEON	RW	32	0x0000 00E0	0x4830 73E0	W	Section 4.12.2.12.3
PM_PWSTST_NEON	R	32	0x0000 00E4	0x4830 73E4	C	Section 4.12.2.12.4
PM_PREPWSTST_NEON	RW	32	0x0000 00E8	0x4830 73E8	C	Section 4.12.2.12.5

Table 4-177. USBHOST_PRM Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Reset Type	Section
RM_RSTST_USBHOST	RW	32	0x0000 0058	0x4830 7458	C	Section 4.12.2.13.1
PM_WKEN_USBHOST	RW	32	0x0000 00A0	0x4830 74A0	W	Section 4.12.2.13.2
PM_MPUGRPSEL_USBHOST	RW	32	0x0000 00A4	0x4830 74A4	W	Section 4.12.2.13.3
Reserved	RW	32	0x0000 00A8	0x4830 74A8	W	
PM_WKST_USBHOST	RW	32	0x0000 00B0	0x4830 74B0	W	Section 4.12.2.13.4
PM_WKDEP_USBHOST	RW	32	0x0000 00C8	0x4830 74C8	W	Section 4.12.2.13.5
PM_PWSTCTRL_USBHOST	RW	32	0x0000 00E0	0x4830 74E0	W	Section 4.12.2.13.6
PM_PWSTST_USBHOST	R	32	0x0000 00E4	0x4830 74E4	C	Section 4.12.2.13.7
PM_PREPWSTST_USBHOST	RW	32	0x0000 00E8	0x4830 74E8	C	Section 4.12.2.13.8

4.12.2.2 OCP_System_Reg_PRM Register Descriptions

4.12.2.2.1 PRM_REVISION

Table 4-178. PRM_REVISION

Address Offset	0x0000 0004		
Physical Address	0x4830 6804	**Instance**	OCP_System_Reg_PRM
Description	This register contains the IP revision code for the PRM part of the PRCM		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	0x10

4.12.2.2.2 PRM_SYSCONFIG

Table 4-179. PRM_SYSCONFIG

Address Offset	0x0000 0014		
Physical Address	0x4830 6814	**Instance**	OCP_System_Reg_PRM
Description	This register controls the various parameters of the interface		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	AUTOIDLE	Internal clock gating strategy (for the CM part of the PRCM) 0x0: Interface clock is free-running 0x1: Automatic clock gating strategy is enabled, based on the interface activity.	RW	0x1

4.12.2.2.3 PRM_IRQSTATUS_MPU

Table 4-180. PRM_IRQSTATUS_MPU

Address Offset	0x0000 0018		
Physical Address	0x4830 6818	**Instance**	OCP_System_Reg_PRM
Description	This interrupt status register regroups all the status of the module internal events that can generate an interrupt. Write 1 to a given bit resets this bit. This registers applies on the interrupt line 0 mapped to the MPU interrupt controller.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						SND_PERIPH_DPLL_ST	RESERVED																	MPU_DPLL_ST	PERIPH_DPLL_ST	CORE_DPLL_ST	TRANSITION_ST	EVGENOFF_ST	EVGENON_ST	RESERVED	WKUP_ST		

Bits	Field Name	Description	Type	Reset
31:26	RESERVED	Reserved: keep at reset value.	R	0x00
25	SND_PERIPH_DPLL_ST	DPLL5 recalibration event status Read 0x0: DPLL5 recalibration event is false Write 0x0: Status bit unchanged Read 0x1: DPLL5 recalibration event is true (pending) Write 0x1: Status bit is cleared to 0	RW	0x0
24:8	RESERVED	Reserved: keep at reset value.	R	0x00000
7	MPU_DPLL_ST	DPLL1 recalibration event status Read 0x0: DPLL1 recalibration event is false Write 0x0: Status bit unchanged Read 0x1: DPLL1 recalibration event is true (pending) Write 0x1: Status bit is cleared to 0	RW	0x0
6	PERIPH_DPLL_ST	DPLL4 recalibration event status Read 0x0: DPLL4 recalibration event is false Write 0x0: Status bit unchanged Read 0x1: DPLL4 recalibration event is true (pending) Write 0x1: Status bit is cleared to 0	RW	0x0
5	CORE_DPLL_ST	DPLL3 recalibration event status Read 0x0: DPLL3 recalibration event is false Write 0x0: Status bit unchanged Read 0x1: DPLL3 recalibration event is true (pending) Write 0x1: Status bit is cleared to 0	RW	0x0
4	TRANSITION_ST	Software supervised transition completed event status Read 0x0: Software supervised transition completed event is false Write 0x0: Status bit unchanged Read 0x1: Software supervised transition completed event is true (pending) Write 0x1: Status bit is cleared to 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
3	EVGENOFF_ST	Event Generator endOFFtime status Read 0x0: End of OFF time event is false Write 0x0: Status bit unchanged Read 0x1: End of OFF time event is true (pending) Write 0x1: Status bit is cleared to 0.	RW	0x0
2	EVGENON_ST	Event Generator endONtime status Read 0x0: End of ON time event is false Write 0x0: Status bit unchanged Read 0x1: End of ON time event is true (pending) Write 0x1: Status bit is cleared to 0.	RW	0x0
1	RESERVED	Reserved: keep at reset value.	R	0x0
0	WKUP_ST	MPU peripherals group wake-up event status Read 0x0: Wake-up event is false Write 0x0: Status bit unchanged Read 0x1: Wake-up event is true (pending) Write 0x1: Status bit is cleared to 0.	RW	0x0

4.12.2.2.4 PRM_IRQENABLE_MPU

Table 4-181. PRM_IRQENABLE_MPU

Address Offset	0x0000 001C		
Physical Address	0x4830 681C	**Instance**	OCP_System_Reg_PRM
Description	The interrupt enable register allows masking/unmasking the module internal sources of interrupt, on a event-by-event basis. This registers applies on the interrupt line 0 mapped to the MPU interrupt controller.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							SND_PERIPH_DPLL_RECAL_EN	RESERVED																MPU_DPLL_RECAL_EN	PERIPH_DPLL_RECAL_EN	CORE_DPLL_RECAL_EN	TRANSITION_EN	EVGENOFF_EN	EVGENON_EN	RESERVED	WKUP_EN

Bits	Field Name	Description	Type	Reset
31:26	RESERVED	Reserved: keep at reset value.	R	0x00
25	SND_PERIPH_DPLL_RECAL_EN	DPLL5 recalibration mask 0x0: DPLL5 recalibration event is masked 0x1: DPLL5 recalibration event generates an interrupt	RW	0x0
24:8	RESERVED	Reserved: keep at reset value.	R	0x00000
7	MPU_DPLL_RECAL_EN	DPLL1 recalibration mask 0x0: DPLL1 recalibration event is masked 0x1: DPLL1 recalibration event generates an interrupt	RW	0x0
6	PERIPH_DPLL_RECAL_EN	DPLL4 recalibration mask 0x0: DPLL4 recalibration event is masked 0x1: DPLL4 recalibration event generates an interrupt	RW	0x0
5	CORE_DPLL_RECAL_EN	DPLL3 recalibration mask 0x0: DPLL3 recalibration event is masked 0x1: DPLL3 recalibration event generates an interrupt	RW	0x0
4	TRANSITION_EN	Software supervised transition completed event mask 0x0: Software supervised transition completed event is masked. 0x1: Software supervised transition completed event generates an interrupt.	RW	0x0
3	EVGENOFF_EN	Event Generator endOFFtime mask 0x0: End of OFF time event is masked 0x1: End of OFF time event generates an interrupt	RW	0x0
2	EVGENON_EN	Event Generator endONtime mask 0x0: End of ON time event is masked 0x1: End of ON time event generates an interrupt	RW	0x0
1	RESERVED	Reserved: keep at reset value.	R	0x0

Bits	Field Name	Description	Type	Reset
0	WKUP_EN	MPU peripherals group wake-up event mask 0x0: MPU peripherals group wake-up event is masked 0x1: MPU peripherals group wake-up event generates an interrupt	RW	0x0

4.12.2.3 MPU_PRM Register Descriptions

4.12.2.3.1 RM_RSTST_MPU

Table 4-182. RM_RSTST_MPU

Address Offset	0x0000 0058		
Physical Address	0x4830 6958	**Instance**	MPU_PRM
Description	This register logs the different reset sources of the MPU domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				EMULATION_MPU_RST	RESERVED										GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Reserved: keep at reset value.	R	0x00000
11	EMULATION_MPU_RST	Emulation reset Read 0x0: No emulation reset. Write 0x0: Status bit unchanged Read 0x1: MPU domain has been reset upon an emulation reset Write 0x1: Status bit is cleared to 0	RW	0x0
10:2	RESERVED	Reserved: keep at reset value.	R	0x000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset. Write 0x0: Status bit unchanged Read 0x1: MPU domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset. Write 0x0: Status bit unchanged Read 0x1: MPU domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.3.2 PM_WKDEP_MPU

Table 4-183. PM_WKDEP_MPU

Address Offset	0x0000 00C8		
Physical Address	0x4830 69C8	**Instance**	MPU_PRM
Description	This register allows enabling or disabling the wake-up of the MPU domain upon another domain wakeup events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								EN_PER	RESERVED	EN_DSS	RESERVED				EN_CORE

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7	EN_PER	PER domain dependency 0x0: MPU domain is independent of PER domain wake-up event. 0x1: MPU domain is woken-up upon PER domain wake-up event.	RW	0x1
6	RESERVED	Reserved: keep at reset value.	R	0x0
5	EN_DSS	DSS domain dependency 0x0: MPU domain is independent of DSS domain wake-up event. 0x1: MPU domain is woken-up upon DSS domain wake-up event.	RW	0x1
4:1	RESERVED	Reserved: keep at reset value.	R	0x2
0	EN_CORE	CORE domain dependency 0x0: MPU domain is independent of CORE domain wake-up event. 0x1: MPU domain is woken-up upon CORE domain wake-up event.	RW	0x1

4.12.2.3.3 PM_EVGENCTRL_MPU

Table 4-184. PM_EVGENCTRL_MPU

Address Offset	0x0000 00D4		
Physical Address	0x4830 69D4	**Instance**	MPU_PRM
Description	This register allows controlling the feature of the event generator.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											OFFLOADMODE			ONLOADMODE		ENABLE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4:3	OFFLOADMODE	OFF load mode setting 0x0: Load on update of PM_EVGENOFFTIM_MPU 0x1: Reserved 0x2: Load on MPU standby signal assertion 0x3: Auto load	RW	0x2
2:1	ONLOADMODE	ON load mode setting 0x0: Load on update of PM_EVGENONTIM_MPU 0x1: Load on MPU standby signal de-assertion 0x2: Reserved 0x3: Auto load	RW	0x1
0	ENABLE	Event generator control 0x0: Disable event generator 0x1: Enable event generator	RW	0x0

4.12.2.3.4 PM_EVGENONTIM_MPU

Table 4-185. PM_EVGENONTIM_MPU

Address Offset	0x0000 00D8		
Physical Address	0x4830 69D8	**Instance**	MPU_PRM
Description	This register sets the ON count duration of the event generator (number of system clock cycles).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ONTIMEVAL																															

Bits	Field Name	Description	Type	Reset
31:0	ONTIMEVAL	Number of system clock cycles for the ON period.	RW	0x00000000

4.12.2.3.5 PM_EVGENOFFTIM_MPU

Table 4-186. PM_EVGENOFFTIM_MPU

Address Offset	0x0000 00DC		
Physical Address	0x4830 69DC	**Instance**	MPU_PRM
Description	This register sets the OFF count duration of the event generator (number of system clock cycles).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFTIMEVAL																															

Bits	Field Name	Description	Type	Reset
31:0	OFFTIMEVAL	Number of system clock cycles for the OFF period.	RW	0x00000000

4.12.2.3.6 PM_PWSTCTRL_MPU

Table 4-187. PM_PWSTCTRL_MPU

Address Offset	0x0000 00E0		
Physical Address	0x4830 69E0	**Instance**	MPU_PRM
Description	This register controls the MPU domain power state transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														L2CACHEONSTATE		RESERVED							L2CACHERETSTATE	RESERVED				MEMORYCHANGE	LOGICL1CACHERETSTATE	POWERSTATE	

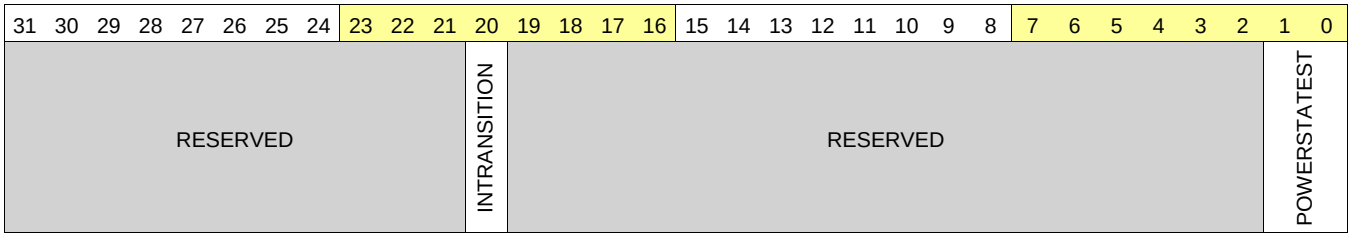
Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17:16	L2CACHEONSTATE	L2 Cache memory state when domain is ON; Other enums: Reserved 0x0: Reserved 0x1: Reserved 0x2: Reserved 0x3: L2 Cache memory is ON when domain is ON.	R/W	0x3
15:9	RESERVED	Reserved: keep at reset value.	R	0x00
8	L2CACHERETSTATE	L2 Cache memory state when domain is RETENTION 0x0:Reserved 0x1: L2 Cache memory is retained when domain is in RETENTION state.	R/W	0x1
7:4	RESERVED	Reserved: keep at reset value.	R	0x0
3	MEMORYCHANGE	Memory change control in ON state 0x0: Disable memory change 0x1: Enable memory change state in ON state. This bit is automatically cleared when memory state is effectively changed.	R/W	0x0
2	LOGICL1CACHERETSTATE	Logic and L1 Cache state when domain is RETENTION 0x0: Reserved 0x1: Logic and L1 Cache are retained when domain is in RETENTION state.	R/W	0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.3.7 PM_PWSTST_MPU

Table 4-188. PM_PWSTST_MPU

Address Offset	0x0000 00E4		
Physical Address	0x4830 69E4	**Instance**	MPU_PRM
Description	This register provides a status on the MPU domain power state.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: MPU domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x00031
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.3.8 PM_PREPWSTST_MPU

Table 4-189. PM_PREPWSTST_MPU

Address Offset	0x0000 00E8		
Physical Address	0x4830 69E8	**Instance**	MPU_PRM
Description	This register provides a status on the MPU domain previous power state. It indicates the state entered during the last sleep transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								LASTL2CACHESTATEENTERED		RESERVED			LASTLOGICL1CACHESTATEENTERED	LASTPOWERSTATEENTERED	

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7:6	LASTL2CACHESTATEENTERED	Last L2 Cache memory state entered 0x0: Reserved 0x1: L2 Cache memory was previously in RETENTION 0x2: Reserved 0x3: L2 Cache memory was previously ON	R/W	0x0
5:3	RESERVED	Reserved: keep at reset value.	R	0x0
2	LASTLOGICL1CACHESTATEENTERED	Last logic and L1 Cache state entered 0x0:Reserved 0x1: MPU domain logic and L1 Cache was previously ON	R/W	0x0
1:0	LASTPOWERSTATEENTERED	Last power state entered 0x0: Reserved 0x1: MPU domain was previously in RETENTION 0x2: MPU domain was previously INACTIVE 0x3: MPU domain was previously ON	R/W	0x0

4.12.2.4 CORE_PRM Register Descriptions

4.12.2.4.1 RM_RSTST_CORE

Table 4-190. RM_RSTST_CORE

Address Offset	0x0000 0058		
Physical Address	0x4830 6A58	**Instance**	CORE_PRM
Description	This register logs the different reset sources of the CORE domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset. Write 0x0: Status bit unchanged Read 0x1: CORE domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: CORE domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.4.2 PM_WKEN1_CORE

Table 4-191. PM_WKEN1_CORE

Address Offset	0x0000 00A0		
Physical Address	0x4830 6AA0	**Instance**	CORE_PRM
Description	This register allows enabling/disabling modules wake-up events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	EN_MMC3	RESERVED				EN_MMC2	EN_MMC1	RESERVED		EN_MCSPI4	EN_MCSPI3	EN_MCSPI2	EN_MCSPI1	EN_I2C3	EN_I2C2	EN_I2C1	EN_UART2	EN_UART1	EN_GPT11	EN_GPT10	EN_MCBSP5	EN_MCBSP1	RESERVED								

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved: keep at reset value.	RW	0x1
30	EN_MMC3	MMC SDIO 3 wake-up control 0x0: MMC 3 wake-up is disabled 0x1: MMC 3 wake-up event is enabled	RW	0x1
29:26	RESERVED	Reserved: keep at reset value.	R	0x0
25	EN_MMC2	MMC SDIO 2 wake-up control 0x0: MMC 2 wake-up is disabled 0x1: MMC 2 wake-up event is enabled	RW	0x1
24	EN_MMC1	MMC SDIO 1 wake-up control 0x0: MMC 1 wake-up is disabled 0x1: MMC 1 wake-up event is enabled	RW	0x1
23:22	RESERVED	Reserved: keep at reset value.	R	0x0
21	EN_MCSPI4	McSPI 4 wake-up control 0x0: McSPI 4 wake-up is disabled 0x1: McSPI 4 wake-up event is enabled	RW	0x1
20	EN_MCSPI3	McSPI 3 wake-up control 0x0: McSPI 3 wake-up is disabled 0x1: McSPI 3 wake-up event is enabled	RW	0x1
19	EN_MCSPI2	McSPI 2 wake-up control 0x0: McSPI 2 wake-up is disabled 0x1: McSPI 2 wake-up event is enabled	RW	0x1
18	EN_MCSPI1	McSPI 1 wake-up control 0x0: McSPI 1 wake-up is disabled 0x1: McSPI 1 wake-up event is enabled	RW	0x1
17	EN_I2C3	I2C 3 wake-up control 0x0: I2C 3 wake-up is disabled 0x1: I2C 3 wake-up event is enabled	RW	0x1
16	EN_I2C2	I2C 2 wake-up control 0x0: I2C 2 wake-up is disabled 0x1: I2C 2 wake-up event is enabled	RW	0x1
15	EN_I2C1	I2C 1 wake-up control 0x0: I2C 1 wake-up is disabled 0x1: I2C 1 wake-up event is enabled	RW	0x1

Bits	Field Name	Description	Type	Reset
14	EN_UART2	UART 2 wake-up control 0x0: UART 2 wake-up is disabled 0x1: UART 2 wake-up event is enabled	RW	0x1
13	EN_UART1	UART 1 wake-up control 0x0: UART 1 wake-up is disabled 0x1: UART 1 wake-up event is enabled	RW	0x1
12	EN_GPT11	GPTIMER 11 wake-up control 0x0: GPTIMER 11 wake-up is disabled 0x1: GPTIMER 11 wake-up event is enabled	RW	0x1
11	EN_GPT10	GPTIMER 10 wake-up control 0x0: GPTIMER 10 wake-up is disabled 0x1: GPTIMER 10 wake-up event is enabled	RW	0x1
10	EN_MCBSP5	McBSP 5 wake-up control 0x0: McBSP 5 wake-up is disabled 0x1: McBSP 5 wake-up event is enabled	RW	0x1
9	EN_MCBSP1	McBSP 1 wake-up control 0x0: McBSP 1 wake-up is disabled 0x1: McBSP 1 wake-up event is enabled	RW	0x1
8:0	RESERVED	Reserved: keep at reset value.	R	0x018

4.12.2.4.3 PM_MPUGRPSEL1_CORE

Table 4-192. PM_MPUGRPSEL1_CORE

Address Offset	0x0000 00A4		
Physical Address	0x4830 6AA4	**Instance**	CORE_PRM
Description	This register allows selecting the group of modules that wake-up the MPU.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	GRPSEL_MMC3	RESERVED				GRPSEL_MMC2	GRPSEL_MMC1	RESERVED		GRPSEL_MCSPI4	GRPSEL_MCSPI3	GRPSEL_MCSPI2	GRPSEL_MCSPI1	GRPSEL_I2C3	GRPSEL_I2C2	GRPSEL_I2C1	GRPSEL_UART2	GRPSEL_UART1	GRPSEL_GPT11	GRPSEL_GPT10	GRPSEL_MCBSP5	GRPSEL_MCBSP1	RESERVED							

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved: keep at reset value.	RW	0x1
30	GRPSEL_MMC3	Select the MMC 3 in the MPU wake-up events group 0x0: MMC 3 is not attached to the MPU wake-up events group. 0x1: MMC 3 is attached to the MPU wake-up events group.	RW	0x1
29:26	RESERVED	Reserved: keep at reset value.	R	0x0
25	GRPSEL_MMC2	Select the MMC 2 in the MPU wake-up events group 0x0: MMC 2 is not attached to the MPU wake-up events group. 0x1: MMC 2 is attached to the MPU wake-up events group.	RW	0x1
24	GRPSEL_MMC1	Select the MMC 1 in the MPU wake-up events group 0x0: MMC 1 is not attached to the MPU wake-up events group. 0x1: MMC 1 is attached to the MPU wake-up events group.	RW	0x1
23:22	RESERVED	Reserved: keep at reset value.	R	0x0
21	GRPSEL_MCSPI4	Select the McSPI 4 in the MPU wake-up events group 0x0: McSPI 4 is not attached to the MPU wake-up events group. 0x1: McSPI 4 is attached to the MPU wake-up events group.	RW	0x1
20	GRPSEL_MCSPI3	Select the McSPI 3 in the MPU wake-up events group 0x0: McSPI 3 is not attached to the MPU wake-up events group. 0x1: McSPI 3 is attached to the MPU wake-up events group.	RW	0x1
19	GRPSEL_MCSPI2	Select the McSPI 2 in the MPU wake-up events group 0x0: McSPI 2 is not attached to the MPU wake-up events group. 0x1: McSPI 2 is attached to the MPU wake-up events group.	RW	0x1
18	GRPSEL_MCSPI1	Select the McSPI 1 in the MPU wake-up events group 0x0: McSPI 1 is not attached to the MPU wake-up events group. 0x1: McSPI 1 is attached to the MPU wake-up events group.	RW	0x1
17	GRPSEL_I2C3	Select the I2C 3 in the MPU wake-up events group 0x0: I2C 3 is not attached to the MPU wake-up events group. 0x1: I2C 3 is attached to the MPU wake-up events group.	RW	0x1
16	GRPSEL_I2C2	Select the I2C 2 in the MPU wake-up events group 0x0: I2C 2 is not attached to the MPU wake-up events group. 0x1: I2C 2 is attached to the MPU wake-up events group.	RW	0x1
15	GRPSEL_I2C1	Select the I2C 1 in the MPU wake-up events group 0x0: I2C 1 is not attached to the MPU wake-up events group. 0x1: I2C 1 is attached to the MPU wake-up events group.	RW	0x1

Bits	Field Name	Description	Type	Reset
14	GRPSEL_UART2	Select the UART 2 in the MPU wake-up events group 0x0: UART 2 is not attached to the MPU wake-up events group. 0x1: UART 2 is attached to the MPU wake-up events group.	RW	0x1
13	GRPSEL_UART1	Select the UART 1 in the MPU wake-up events group 0x0: UART 1 is not attached to the MPU wake-up events group. 0x1: UART 1 is attached to the MPU wake-up events group.	RW	0x1
12	GRPSEL_GPT11	Select the GPTIMER 11 in the MPU wake-up events group 0x0: GPTIMER 11 is not attached to the MPU wake-up events group. 0x1: GPTIMER 11 is attached to the MPU wake-up events group.	RW	0x1
11	GRPSEL_GPT10	Select the GPTIMER 10 in the MPU wake-up events group 0x0: GPTIMER 10 is not attached to the MPU wake-up events group. 0x1: GPTIMER 10 is attached to the MPU wake-up events group.	RW	0x1
10	GRPSEL_MCBSP5	Select the McBSP 5 in the MPU wake-up events group 0x0: McBSP 5 is not attached to the MPU wake-up events group. 0x1: McBSP 5 is attached to the MPU wake-up events group.	RW	0x1
9	GRPSEL_MCBSP1	Select the McBSP 1 in the MPU wake-up events group 0x0: McBSP 1 is not attached to the MPU wake-up events group. 0x1: McBSP 1 is attached to the MPU wake-up events group.	RW	0x1
8:0	RESERVED	Reserved: keep at reset value.	R	0x018

4.12.2.4.4 PM_WKST1_CORE

Table 4-193. PM_WKST1_CORE

Address Offset	0x0000 00B0		
Physical Address	0x4830 6AB0	**Instance**	CORE_PRM
Description	This register logs the modules wake-up events. Must be cleared by software. It prevents further domain transition if it is not cleared.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	ST_MMC3	RESERVED				ST_MMC2	ST_MMC1	RESERVED		ST_MCSPI4	ST_MCSPI3	ST_MCSPI2	ST_MCSPI1	ST_I2C3	ST_I2C2	ST_I2C1	ST_UART2	ST_UART1	ST_GPT11	ST_GPT10	ST_MCBSP5	ST_MCBSP1	RESERVED								

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved: keep at reset value.	RW	0x0
30	ST_MMC3	MMC 3 Wake-up status Read 0x0: MMC 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: MMC 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
29:26	RESERVED	Reserved: keep at reset value.	R	0x0
25	ST_MMC2	MMC 2 Wake-up status Read 0x0: MMC 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: MMC 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
24	ST_MMC1	MMC 1 Wake-up status Read 0x0: MMC 1 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: MMC 1 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
23:22	RESERVED	Reserved: keep at reset value.	R	0x0
21	ST_MCSPI4	McSPI 4 Wake-up status Read 0x0: McSPI 4 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McSPI 4 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
20	ST_MCSPI3	McSPI 3 Wake-up status Read 0x0: McSPI 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McSPI 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
19	ST_MCSPI2	McSPI 2 Wake-up status Read 0x0: McSPI 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McSPI 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0

Bits	Field Name	Description	Type	Reset
18	ST_MCSPI1	McSPI 1 Wake-up status Read 0x0: McSPI 1 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McSPI 1 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
17	ST_I2C3	I2C 3 Wake-up status Read 0x0: I2C 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: I2C 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
16	ST_I2C2	I2C 2 Wake-up status Read 0x0: I2C 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: I2C 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
15	ST_I2C1	I2C 1 Wake-up status Read 0x0: I2C 1 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: I2C 1 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
14	ST_UART2	UART 2 Wake-up status Read 0x0: UART 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: UART 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
13	ST_UART1	UART 1 Wake-up status Read 0x0: UART 1 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: UART 1 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
12	ST_GPT11	GPTIMER 11 Wake-up status Read 0x0: GPTIMER 11 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: Status bit is cleared to 0 Write 0x1: GPTIMER 11 wake-up occurred	RW	0x0
11	ST_GPT10	GPTIMER 10 Wake-up status Read 0x0: GPTIMER 10 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 10 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
10	ST_MCBSP5	McBSP 5 Wake-up status Read 0x0: McBSP 5 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McBSP 5 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0

Bits	Field Name	Description	Type	Reset
9	ST_MCBSP1	McBSP 1 Wake-up status Read 0x0: McBSP 1 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McBSP 1 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
8:0	RESERVED	Reserved: keep at reset value.	R	0x000

4.12.2.4.5 PM_WKST3_CORE

Table 4-194. PM_WKST3_CORE

Address Offset	0x0000 00B8		
Physical Address	0x4830 6AB8	**Instance**	CORE_PRM
Description	This register logs the modules wake-up events. Must be cleared by software. It prevents further domain transition if it is not cleared.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													ST_USBTLL	RESERVED	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	ST_USBTLL	USB TLL Wake-up status Read 0x0: USB TLL wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: USB TLL wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
1:0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.4.6 PM_PWSTCTRL_CORE

Table 4-195. PM_PWSTCTRL_CORE

Address Offset	0x0000 00E0		
Physical Address	0x4830 6AE0	**Instance**	CORE_PRM
Description	This register controls the CORE domain power state transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												MEM2ONSTATE		MEM1ONSTATE		RESERVED						MEM2RETSTATE	MEM1RETSTATE	RESERVED				MEMORYCHANGE	LOGICRETSTATE	POWERSTATE	

Bits	Field Name	Description	Type	Reset
31:20	RESERVED	Reserved: keep at reset value.	R	0x000
19:18	MEM2ONSTATE	Memory block 2 state when domain is ON 0x0: Reserved 0x1: Memory block 2 is in RETENTION when domain is ON. 0x2: Reserved 0x3: Memory block 2 is ON when domain is ON.	RW	0x3
17:16	MEM1ONSTATE	Memory block 1 state when domain is ON 0x0: Reserved 0x1: Memory block 1 is in RETENTION when domain is ON. 0x2: Reserved 0x3: Memory block 1 is ON when domain is ON.	R/W	0x3
15:10	RESERVED	Reserved: keep at reset value.	R	0x00
9	MEM2RETSTATE	Memory block 2 state when domain is RETENTION 0x0: Reserved 0x1: Memory block 2 is retained when domain is in RETENTION state.	R/W	0x1
8	MEM1RETSTATE	Memory block 1 state when domain is RETENTION 0x0: Reserved 0x1: Memory block 1 is retained when domain is in RETENTION state.	R/W	0x1
7:4	RESERVED	Reserved: keep at reset value.	R	0x0
3	MEMORYCHANGE	Memory change control in ON state 0x0: Disable memory change 0x1: Enable memory change state in ON state. This bit is automatically cleared when memory state is effectively changed.	R/W	0x0
2	LOGICRETSTATE	Logic state when domain is RETENTION 0x0: Reserved 0x1: Logic is retained when domain is in RETENTION state.	R/W	0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.4.7 PM_PWSTST_CORE

Table 4-196. PM_PWSTST_CORE

Address Offset	0x0000 00E4		
Physical Address	0x4830 6AE4	**Instance**	CORE_PRM
Description	This register provides a status on the power state transition of the CORE domain.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											INTRANSITION	RESERVED																		POWERSTATEST	

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: CORE domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x0003D
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.4.8 PM_PREPWSTST_CORE

Table 4-197. PM_PREPWSTST_CORE

Address Offset	0x0000 00E8		
Physical Address	0x4830 6AE8	**Instance**	CORE_PRM
Description	This register provides a status on the power state transition of the CORE domain.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								LASTMEM2STATEENTERED		LASTMEM1STATEENTERED		RESERVED	LASTLOGICSTATEENTERED	LASTPOWERSTATEENTERED	

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7:6	LASTMEM2STATEENTERED	Last Memory block 2 state entered 0x0: Reserved 0x1: Memory was previously in RETENTION 0x2: Reserved 0x3: Memory was previously ON	R/W	0x0
5:4	LASTMEM1STATEENTERED	Last Memory block 1 state entered 0x0: Reserved 0x1:Memory was previously in RETENTION 0x2: Reserved 0x3: Memory was previously ON	R/W	0x0
3	RESERVED	Reserved: keep at reset value.	R	0x0
2	LASTLOGICSTATEENTERED	Last logic state entered 0x0: Reserved 0x1: CORE domain logic was previously ON	R/W	0x0
1:0	LASTPOWERSTATEENTERED	Last power state entered 0x0: Reserved 0x1: CORE domain was previously in RETENTION 0x2: CORE domain was previously INACTIVE 0x3: CORE domain was previously ON	R/W	0x0

4.12.2.4.9 PM_WKEN3_CORE

Table 4-198. PM_WKEN3_CORE

Address Offset	0x0000 00F0		
Physical Address	0x4830 6AF0	**Instance**	CORE_PRM
Description	This register allows enabling/disabling modules wake-up events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													EN_USBTLL	RESERVED	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	EN_USBTLL	USB TLL wake-up control 0x0: USB TLL wake-up is disabled 0x1: USB TLL wake-up event is enabled	RW	0x1
1:0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.4.10 PM_MPUGRPSEL3_CORE

Table 4-199. PM_MPUGRPSEL3_CORE

Address Offset	0x0000 00F8		
Physical Address	0x4830 6AF8	**Instance**	CORE_PRM
Description	This register allows selecting the group of modules that wake-up the MPU.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													GRPSEL_USBTLL	RESERVED	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	GRPSEL_USBTLL	Select the USB TLL in the MPU wake-up events group 0x0: USB TLL is not attached to the MPU wake-up events group. 0x1: USB TLL is attached to the MPU wake-up events group.	RW	0x1
1:0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.5 SGX_PRM Register Descriptions

4.12.2.5.1 RM_RSTST_SGX

Table 4-200. RM_RSTST_SGX

Address Offset	0x0000 0058		
Physical Address	0x4830 6B58	**Instance**	SGX_PRM
Description	This register logs the different reset sources of the SGX domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: SGX domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: SGX domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.5.2 PM_WKDEP_SGX

Table 4-201. PM_WKDEP_SGX

Address Offset	0x0000 00C8		
Physical Address	0x4830 6BC8	**Instance**	SGX_PRM
Description	This register allows enabling or disabling the wake-up of the SGX domain upon another domain wakeup.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											EN_WKUP	RESERVED		EN_MPU	RESERVED

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4	EN_WKUP	WAKEUP domain dependency	RW	0x1
		0x0: SGX domain is independent of WKUP domain wake-up event.		
		0x1: SGX domain is woken-up upon WKUP domain wake-up event.		
3:2	RESERVED	Reserved: keep at reset value.	R	0x1
1	EN_MPU	MPU domain dependency	RW	0x1
		0x0: SGX domain is independent of MPU domain wake-up.		
		0x1: SGX domain is woken-up upon MPU domain wake-up.		
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.5.3 PM_PWSTCTRL_SGX

Table 4-202. PM_PWSTCTRL_SGX

Address Offset	0x0000 00E0		
Physical Address	0x4830 6AE0	**Instance**	CORE_PRM
Description	This register controls the CORE domain power state transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MEMONSTATE		RESERVED							MEMRETSTATE	RESERVED					LOGICRETSTATE	POWERSTATE	

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17:16	MEMONSTATE	Memory state when ON 0x3: Memory is always ON when domain is ON.	R	0x3
15:9	RESERVED	Reserved: keep at reset value.	R	0x00
8	MEMRETSTATE	Memory state when RETENTION 0x1: Memory is always retained when domain is in RETENTION state.	R	0x1
7:3	RESERVED	Reserved: keep at reset value.	R	0x00
2	LOGICRETSTATE	Logic state when RETENTION 0x1: Logic is always retained when domain is in RETENTION state.	R	0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.5.4 PM_PWSTST_SGX

Table 4-203. PM_PWSTST_SGX

Address Offset	0x0000 00E4		
Physical Address	0x4830 6BE4	**Instance**	SGX_PRM
Description	This register provides a status on the power state transition of the SGX domain.		
Type	R		

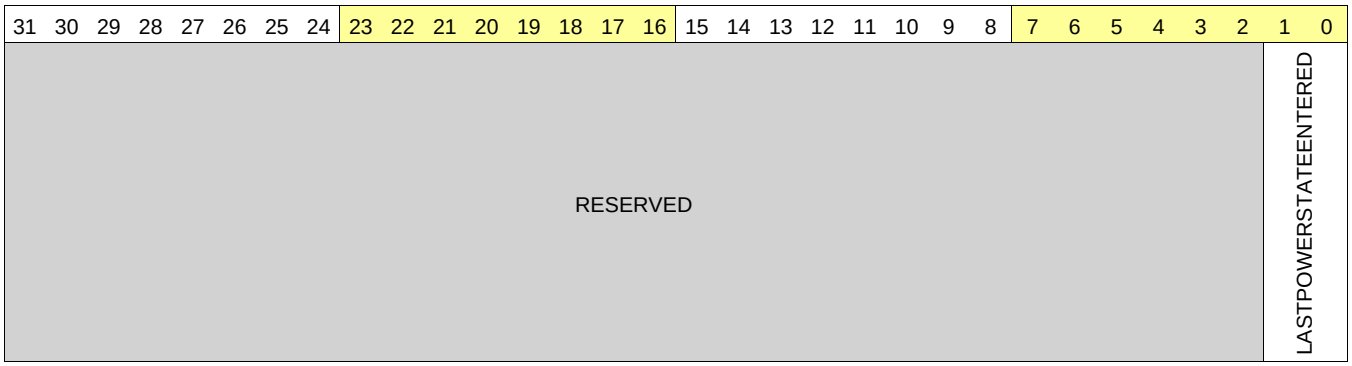
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											INTRANSITION	RESERVED																		POWERSTATEST	

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: SGX domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x00000
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.5.5 PM_PREPWSTCTRL_SGX

Table 4-204. RM_RSTST_SGX

Address Offset	0x0000 00E8		
Physical Address	0x4830 6BE8	**Instance**	SGX_PRM
Description	This register provides a status on the SGX domain previous power state. It indicates the state entered during the last sleep transition.		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	LASTPOWERSTATEENTERED	Last power state entered	RW	0x0
		0x0: Reserved		
		0x1: SGX domain was previously in RETENTION		
		0x2: SGX domain was previously INACTIVE		
		0x3: SGX domain was previously ON		

4.12.2.6 WKUP_PRM Register Descriptions

4.12.2.6.1 PM_WKEN_WKUP

Table 4-205. PM_WKEN_WKUP

Address Offset	0x0000 00A0		
Physical Address	0x4830 6CA0	**Instance**	WKUP_PRM
Description	This register allows enabling/disabling modules wake-up events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												EN_GPIO1	RESERVED	EN_GPT12	EN_GPT1

Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved: keep at reset value.	R	0x000003C
3	EN_GPIO1	GPIO 1 wake-up control 0x0: GPIO 1 wake-up is disabled 0x1: GPIO 1 wake-up event is enabled	RW	0x1
2	RESERVED	Reserved: keep at reset value.	R	0x0
1	EN_GPT12	GPTIMER 12 wake-up is always enabled	R	0x1
0	EN_GPT1	GPTIMER 1 wake-up control 0x0: GPTIMER 1 wake-up is disabled 0x1: GPTIMER 1 wake-up event is enabled	RW	0x1

4.12.2.6.2 PM_MPUGRPSEL_WKUP

Table 4-206. PM_MPUGRPSEL_WKUP

Address Offset	0x0000 00A4		
Physical Address	0x4830 6CA4	**Instance**	WKUP_PRM
Description	IO pad is always selected in the MPU wake-up events group		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												GRPSEL_GPIO1	RESERVED	GRPSEL_GPT12	GRPSEL_GPT1

Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved: keep at reset value.	R	0x000003C
3	GRPSEL_GPIO1	Select the GPIO 1 in the MPU wake-up events group 0x0: GPIO 1 is not attached to the MPU wake-up events group. 0x1: GPIO 1 is attached to the MPU wake-up events group.	RW	0x1
2	RESERVED	Reserved: keep at reset value.	R	0x0
1	GRPSEL_GPT12	GPTIMER 12 is always selected in the MPU wake-up events group	R	0x1
0	GRPSEL_GPT1	Select the GPTIMER 1 in the MPU wake-up events group 0x0: GPTIMER 1 is not attached to the MPU wake-up events group. 0x1: GPTIMER 1 is attached to the MPU wake-up events group.	RW	0x1

4.12.2.6.3 PM_WKST_WKUP

Table 4-207. PM_WKST_WKUP

Address Offset	0x0000 00B0		
Physical Address	0x4830 6CB0	**Instance**	WKUP_PRM
Description	This register logs the modules wake-up events. Must be cleared by software. It prevents further domain transition if it is not cleared.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												ST_GPIO1	RESERVED	ST_GPT12	ST_GPT1

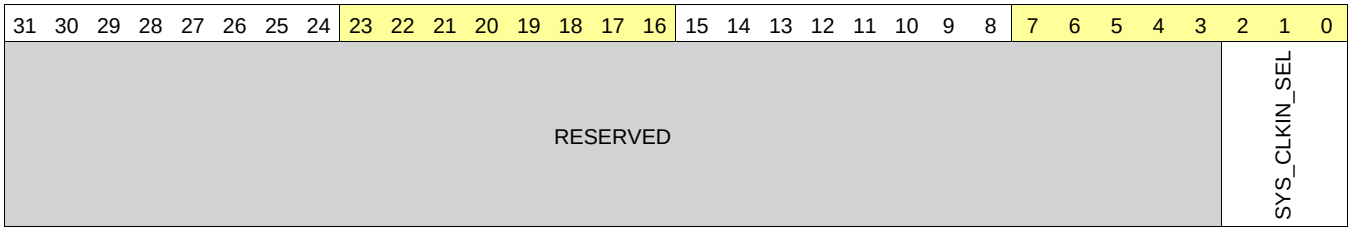
Bits	Field Name	Description	Type	Reset
31:4	RESERVED	Reserved: keep at reset value.	R	0x0000000
3	ST_GPIO1	GPIO 1 Wake-up status Read 0x0: GPIO 1 wakeup did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPIO 1 wakeup occurred Write 0x1: Status bit is cleared to 0	RW	0x0
2	RESERVED	Reserved: keep at reset value.	R	0x0
1	ST_GPT12	GPTIMER 12 Wake-up status Read 0x0: GP Timer 1 wake-up has not occurred or was masked Write 0x0: Status bit unchanged Read 0x1: GP Timer 1 wake-up has occurred Write 0x1: Status bit is cleared to 0	RW	0x0
0	ST_GPT1	GPTIMER 1 Wake-up status Read 0x0: GPTIMER 1 wakeup did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 1 wakeup occurred Write 0x1: Status bit is cleared to 0	RW	0x0

4.12.2.7 Clock_Control_Reg_PRM Register Descriptions

4.12.2.7.1 PRM_CLKSEL

Table 4-208. PRM_CLKSEL

Address Offset	0x0000 0040		
Physical Address	0x4830 6D40	**Instance**	Clock_Control_Reg_PRM
Description	This register controls the selection of the system clock frequency. This register is reset on power-up only.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2:0	SYS_CLKIN_SEL	System clock input selection; Other enums: Reserved	RW	0x4
		0x0: Reserved		
		0x1: Reserved		
		0x2: Reserved		
		0x3: OSC_SYS_CLK is 26 MHz		
		0x4: Reserved		
		0x5: Reserved		

4.12.2.7.2 PRM_CLKOUT_CTRL

Table 4-209. PRM_CLKOUT_CTRL

Address Offset	0x0000 0070		
Physical Address	0x4830 6D70	**Instance**	Clock_Control_Reg_PRM
Description	This register provides control over the SYS_CLKOUT1 output clock.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								CLKOUT_EN	RESERVED						

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7	CLKOUT_EN	This bit controls the external output clock (sys_clkout1) activity 0x0: sys_clkout1 is disabled 0x1: sys_clkout1 is enabled	RW	0x1
6:0	RESERVED	Reserved: keep at reset value.	R	0x00

4.12.2.8 DSS_PRM Register Descriptions

4.12.2.8.1 RM_RSTST_DSS

Table 4-210. RM_RSTST_DSS

Address Offset	0x0000 0058		
Physical Address	0x4830 6E58	**Instance**	DSS_PRM
Description	This register logs the different reset sources of the DSS domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: DISPLAY domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: DISPLAY domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.8.2 PM_WKEN_DSS

Table 4-211. PM_WKEN_DSS

Address Offset	0x0000 00A0		
Physical Address	0x4830 6EA0	**Instance**	DSS_PRM
Description	This register allows enabling/disabling modules wake-up events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															EN_DSS

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	EN_DSS	DSS Wake-up enable 0x0: DSS wake-up is disabled 0x1: DSS wake-up event is enabled	RW	0x1

4.12.2.8.3 PM_WKDEP_DSS

Table 4-212. PM_WKDEP_DSS

Address Offset	0x0000 00C8		
Physical Address	0x4830 6EC8	**Instance**	DSS_PRM
Description	This register allows enabling or disabling the wake-up of the DISPLAY domain upon another domain wakeup.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											EN_WKUP	RESERVED		EN_MPU	RESERVED

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4	EN_WKUP	WAKEUP domain dependency	RW	0x1
		0x0: DSS domain is independent of WKUP domain wake-up event.		
		0x1: DSS domain is woken-up upon WKUP domain wake-up event.		
3:2	RESERVED	Reserved: keep at reset value.	R	0x1
1	EN_MPU	MPU domain dependency	RW	0x1
		0x0: DSS domain is independent of MPU domain wake-up.		
		0x1: DSS domain is woken-up upon MPU domain wake-up.		
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.8.4 PM_PWSTCTRL_DSS

Table 4-213. PM_PWSTCTRL_DSS

Address Offset	0x0000 00E0		
Physical Address	0x4830 6EE0	**Instance**	DSS_PRM
Description	This register controls the DISPLAY domain power state transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MEMONSTATE		RESERVED							MEMRETSTATE	RESERVED					LOGICRETSTATE	POWERSTATE	

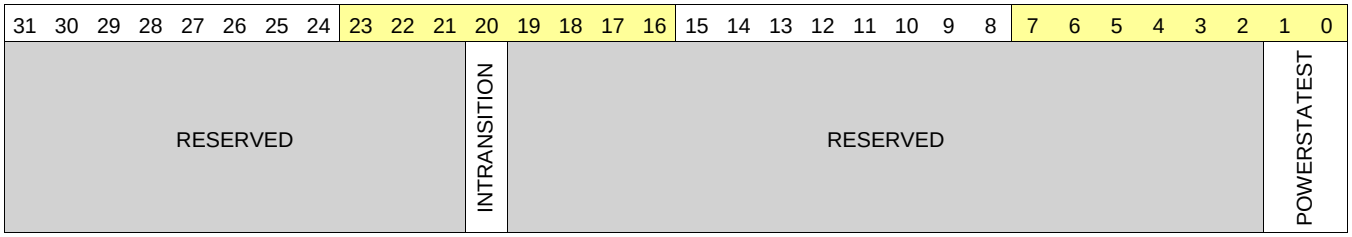
Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17:16	MEMONSTATE	Memory state when ON 0x3: Memory is always ON when domain is ON.	R	0x3
15:9	RESERVED	Reserved: keep at reset value.	R	0x00
8	MEMRETSTATE	Memory state when RETENTION 0x1: Memory is always retained when domain is in RETENTION state.	R	0x1
7:3	RESERVED	Reserved: keep at reset value.	R	0x00
2	LOGICRETSTATE	Logic state when RETENTION 0x1: Logic is always retained when domain is in RETENTION state.	R	0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.8.5 PM_PWSTST_DSS

Table 4-214. PM_PWSTST_DSS

Address Offset	0x0000 00E4		
Physical Address	0x4830 6EE4	**Instance**	DSS_PRM
Description	This register provides a status on the power state transition of the DSS domain.		
Type	R		

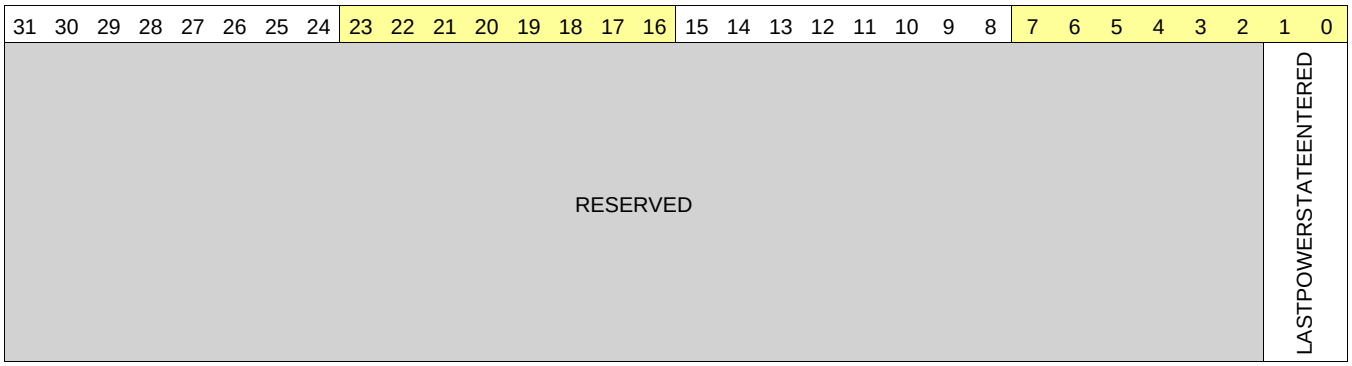


Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: DISPLAY domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x00000
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.8.6 PM_PREPWSTST_DSS

Table 4-215. PM_PREPWSTST_DSS

Address Offset	0x0000 00E8		
Physical Address	0x4830 6EE8	**Instance**	DSS_PRM
Description	This register provides a status on the DSS domain previous power state. It indicates the state entered during the last sleep transition.		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	LASTPOWERSTATEENTERED	Last power state entered	RW	0x0
		0x0: Reserved		
		0x1: DSS domain was previously in RETENTION		
		0x2:DSS domain was previously INACTIVE		
		0x3: DSS domain was previously ON		

4.12.2.9 PER_PRM Register Descriptions

4.12.2.9.1 RM_RSTST_PER

Table 4-216. RM_RSTST_PER

Address Offset	0x0000 0058		
Physical Address	0x4830 7058	**Instance**	PER_PRM
Description	This register logs the different reset sources of the PERIPHERAL domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: PER domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: PER domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.9.2 PM_WKEN_PER

Table 4-217. PM_WKEN_PER

Address Offset	0x0000 00A0		
Physical Address	0x4830 70A0	**Instance**	PER_PRM
Description	This register allows enabling/disabling modules wake-up events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														EN_GPIO6	EN_GPIO5	EN_GPIO4	EN_GPIO3	EN_GPIO2	RESERVED	EN_UART3	EN_GPT9	EN_GPT8	EN_GPT7	EN_GPT6	EN_GPT5	EN_GPT4	EN_GPT3	EN_GPT2	EN_MCBSP4	EN_MCBSP3	EN_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17	EN_GPIO6	GPIO 6 wake-up control 0x0: GPIO 6 wake-up is disabled 0x1: GPIO 6 wake-up event is enabled	RW	0x1
16	EN_GPIO5	GPIO 5 wake-up control 0x0: GPIO 5 wake-up is disabled 0x1: GPIO 5 wake-up event is enabled	RW	0x1
15	EN_GPIO4	GPIO 4 wake-up control 0x0: GPIO 4 wake-up is disabled 0x1: GPIO 4 wake-up event is enabled	RW	0x1
14	EN_GPIO3	GPIO 3 wake-up control 0x0: GPIO 3 wake-up is disabled 0x1: GPIO 3 wake-up event is enabled	RW	0x1
13	EN_GPIO2	GPIO 2 wake-up control 0x0: GPIO 2 wake-up is disabled 0x1: GPIO 2 wake-up event is enabled	RW	0x1
12	RESERVED	Reserved: keep at reset value.	R	0x0
11	EN_UART3	UART 3 wake-up control 0x0: UART 3 wake-up is disabled 0x1: UART 3 wake-up event is enabled	RW	0x1
10	EN_GPT9	GPTIMER 9 wake-up control 0x0: GPTIMER 9 wake-up is disabled 0x1: GPTIMER 9 wake-up event is enabled	RW	0x1
9	EN_GPT8	GPTIMER 8 wake-up control 0x0: GPTIMER 8 wake-up is disabled 0x1: GPTIMER 8 wake-up event is enabled	RW	0x1
8	EN_GPT7	GPTIMER 7 wake-up control 0x0: GPTIMER 7 wake-up is disabled 0x1: GPTIMER 7 wake-up event is enabled	RW	0x1
7	EN_GPT6	GPTIMER 6 wake-up control 0x0: GPTIMER 6 wake-up is disabled 0x1: GPTIMER 6 wake-up event is enabled	RW	0x1

Bits	Field Name	Description	Type	Reset
6	EN_GPT5	GPTIMER 5 wake-up control 0x0: GPTIMER 5 wake-up is disabled 0x1: GPTIMER 5 wake-up event is enabled	RW	0x1
5	EN_GPT4	GPTIMER 4 wake-up control 0x0: GPTIMER 4 wake-up is disabled 0x1: GPTIMER 4 wake-up event is enabled	RW	0x1
4	EN_GPT3	GPTIMER 3 wake-up control 0x0: GPTIMER 3 wake-up is disabled 0x1: GPTIMER 3 wake-up event is enabled	RW	0x1
3	EN_GPT2	GPTIMER 2 wake-up control 0x0: GPTIMER 2 wake-up is disabled 0x1: GPTIMER 2 wake-up event is enabled	RW	0x1
2	EN_MCBSP4	McBSP 4 wake-up control 0x0: McBSP 4 wake-up is disabled 0x1: McBSP 4 wake-up event is enabled	RW	0x1
1	EN_MCBSP3	McBSP3 wake-up control 0x0: McBSP 3 wake-up is disabled 0x1: McBSP 3 wake-up event is enabled	RW	0x1
0	EN_MCBSP2	McBSP 2 wake-up control 0x0: McBSP 2 wake-up is disabled 0x1: McBSP 2 wake-up event is enabled	RW	0x1

4.12.2.9.3 PM_MPUGRPSEL_PER

Table 4-218. PM_MPUGRPSEL_PER

Address Offset	0x0000 00A4		
Physical Address	0x4830 70A4	**Instance**	PER_PRM
Description	This register allows selecting the group of modules that wake-up the MPU.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														GRPSEL_GPIO6	GRPSEL_GPIO5	GRPSEL_GPIO4	GRPSEL_GPIO3	GRPSEL_GPIO2	RESERVED	GRPSEL_UART3	GRPSEL_GPT9	GRPSEL_GPT8	GRPSEL_GPT7	GRPSEL_GPT6	GRPSEL_GPT5	GRPSEL_GPT4	GRPSEL_GPT3	GRPSEL_GPT2	GRPSEL_MCBSP4	GRPSEL_MCBSP3	GRPSEL_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17	GRPSEL_GPIO6	Select the GPIO 6 in the MPU wake-up events group 0x0: GPIO 6 is not attached to the MPU wake-up events group. 0x1: GPIO 6 is attached to the MPU wake-up events group.	RW	0x1
16	GRPSEL_GPIO5	Select the GPIO 5 in the MPU wake-up events group 0x0: GPIO 5 is not attached to the MPU wake-up events group. 0x1: GPIO 5 is attached to the MPU wake-up events group.	RW	0x1
15	GRPSEL_GPIO4	Select the GPIO 4 in the MPU wake-up events group 0x0: GPIO 4 is not attached to the MPU wake-up events group. 0x1: GPIO 4 is attached to the MPU wake-up events group.	RW	0x1
14	GRPSEL_GPIO3	Select the GPIO 3 in the MPU wake-up events group 0x0: GPIO 3 is not attached to the MPU wake-up events group. 0x1: GPIO 3 is attached to the MPU wake-up events group.	RW	0x1
13	GRPSEL_GPIO2	Select the GPIO 2 in the MPU wake-up events group 0x0: GPIO 2 is not attached to the MPU wake-up events group. 0x1: GPIO 2 is attached to the MPU wake-up events group.	RW	0x1
12	RESERVED	Reserved: keep at reset value.	R	0x0
11	GRPSEL_UART3	Select the UART 3 in the MPU wake-up events group 0x0: UART 3 is not attached to the MPU wake-up events group. 0x1: UART 3 is attached to the MPU wake-up events group.	RW	0x1
10	GRPSEL_GPT9	Select the GPTIMER 9 in the MPU wake-up events group 0x0: GPTIMER 9 is not attached to the MPU wake-up events group. 0x1: GPTIMER 9 is attached to the MPU wake-up events group.	RW	0x1
9	GRPSEL_GPT8	Select the GPTIMER 8 in the MPU wake-up events group 0x0: GPTIMER 8 is not attached to the MPU wake-up events group. 0x1: GPTIMER 8 is attached to the MPU wake-up events group.	RW	0x1
8	GRPSEL_GPT7	Select the GPTIMER 7 in the MPU wake-up events group 0x0: GPTIMER 7 is not attached to the MPU wake-up events group. 0x1: GPTIMER 7 is attached to the MPU wake-up events group.	RW	0x1

Bits	Field Name	Description	Type	Reset
7	GRPSEL_GPT6	Select the GPTIMER 6 in the MPU wake-up events group 0x0: GPTIMER 6 is not attached to the MPU wake-up events group. 0x1: GPTIMER 6 is attached to the MPU wake-up events group.	RW	0x1
6	GRPSEL_GPT5	Select the GPTIMER 5 in the MPU wake-up events group 0x0: GPTIMER 5 is not attached to the MPU wake-up events group. 0x1: GPTIMER 5 is attached to the MPU wake-up events group.	RW	0x1
5	GRPSEL_GPT4	Select the GPTIMER 4 in the MPU wake-up events group 0x0: GPTIMER 4 is not attached to the MPU wake-up events group. 0x1: GPTIMER 4 is attached to the MPU wake-up events group.	RW	0x1
4	GRPSEL_GPT3	Select the GPTIMER 3 in the MPU wake-up events group 0x0: GPTIMER 3 is not attached to the MPU wake-up events group. 0x1: GPTIMER 3 is attached to the MPU wake-up events group.	RW	0x1
3	GRPSEL_GPT2	Select the GPTIMER 2 in the MPU wake-up events group 0x0: GPTIMER 2 is not attached to the MPU wake-up events group. 0x1: GPTIMER 2 is attached to the MPU wake-up events group.	RW	0x1
2	GRPSEL_MCBSP4	Select the McBSP 4 in the MPU wake-up events group 0x0: McBSP 4 is not attached to the MPU wake-up events group. 0x1: McBSP 4 is attached to the MPU wake-up events group.	RW	0x1
1	GRPSEL_MCBSP3	Select the McBSP 3 in the MPU wake-up events group 0x0: McBSP 3 is not attached to the MPU wake-up events group. 0x1: McBSP 3 is attached to the MPU wake-up events group.	RW	0x1
0	GRPSEL_MCBSP2	Select the McBSP 2 in the MPU wake-up events group 0x0: McBSP 2 is not attached to the MPU wake-up events group. 0x1: McBSP 2 is attached to the MPU wake-up events group.	RW	0x1

4.12.2.9.4 PM_WKST_PER

Table 4-219. PM_WKST_PER

Address Offset	0x0000 00B0		
Physical Address	0x4830 70B0	**Instance**	PER_PRM
Description	This register logs the modules wake-up events. Must be cleared by software. It prevents further domain transition if it is not cleared.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														ST_GPIO6	ST_GPIO5	ST_GPIO4	ST_GPIO3	ST_GPIO2	RESERVED	ST_UART3	ST_GPT9	ST_GPT8	ST_GPT7	ST_GPT6	ST_GPT5	ST_GPT4	ST_GPT3	ST_GPT2	EN_MCBSP4	EN_MCBSP3	EN_MCBSP2

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17	ST_GPIO6	GPIO 6 Wake-up status Read 0x0: GPIO 6 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPIO 6 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
16	ST_GPIO5	GPIO 5 Wake-up status Read 0x0: GPIO 5 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPIO 5 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
15	ST_GPIO4	GPIO 4 Wake-up status Read 0x0: GPIO 4 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPIO 4 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
14	ST_GPIO3	GPIO 3 Wake-up status Read 0x0: GPIO 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPIO 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
13	ST_GPIO2	GPIO 2 Wake-up status Read 0x0: GPIO 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPIO 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
12	RESERVED	Reserved: keep at reset value.	R	0x0
11	ST_UART3	UART 3 Wake-up status Read 0x0: UART 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: UART 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0

Bits	Field Name	Description	Type	Reset
10	ST_GPT9	GPTIMER 9 Wake-up status Read 0x0: GPTIMER 9 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 9 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
9	ST_GPT8	GPTIMER 8 Wake-up status Read 0x0: GPTIMER 8 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 8 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
8	ST_GPT7	GPTIMER 7 Wake-up status Read 0x0: GPTIMER 7 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 7 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
7	ST_GPT6	GPTIMER 6 Wake-up status Read 0x0: GPTIMER 6 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 6 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
6	ST_GPT5	GPTIMER 5 Wake-up status Read 0x0: GPTIMER 5 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 5 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
5	ST_GPT4	GPTIMER 4 Wake-up status Read 0x0: GPTIMER 4 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 4 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
4	ST_GPT3	GPTIMER 3 Wake-up status Read 0x0: GPTIMER 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
3	ST_GPT2	GPTIMER 2 Wake-up status Read 0x0: GPTIMER 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: GPTIMER 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
2	EN_MCBSP4	McBSP 4 Wake-up status Read 0x0: McBSP 4 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McBSP 4 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0

Bits	Field Name	Description	Type	Reset
1	EN_MCBSP3	McBSP3 Wake-up status Read 0x0: McBSP 3 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McBSP 3 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0
0	EN_MCBSP2	McBSP 2 Wake-up status Read 0x0: McBSP 2 wake-up did not occur or was masked Write 0x0: Status bit unchanged Read 0x1: McBSP 2 wake-up occurred Write 0x1: Status bit is cleared to 0	RW	0x0

4.12.2.9.5 PM_WKDEP_PER

Table 4-220. PM_WKDEP_PER

Address Offset	0x0000 00C8		
Physical Address	0x4830 70C8	**Instance**	PER_PRM
Description	This register allows enabling or disabling the wake-up of the PER domain upon another domain wakeup events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											EN_WKUP	RESERVED		EN_MPU	EN_CORE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4	EN_WKUP	WAKEUP domain dependency	RW	0x1
		0x0: PER domain is independent of WKUP domain wake-up event.		
		0x1: PER domain is woken-up upon WKUP domain wake-up event.		
3:2	RESERVED	Reserved: keep at reset value.	R	0x1
1	EN_MPU	MPU domain dependency	RW	0x1
		0x0: PER domain is independent of MPU domain wake-up event.		
		0x1: PER domain is woken-up upon MPU domain wake-up event.		
0	EN_CORE	CORE domain dependency (CORE-L3 clock domain only, not CORE-L4)	RW	0x1
		0x0: PER domain is independent of CORE domain wake-up event (CORE-L3 clock domain only, not CORE-L4).		
		0x1: PER domain is not woken-up upon CORE domain wake-up event.		

4.12.2.9.6 PM_PWSTCTRL_PER

Table 4-221. PM_PWSTCTRL_PER

Address Offset	0x0000 00E0		
Physical Address	0x4830 70E0	**Instance**	PER_PRM
Description	This register controls the PERIPHERAL domain power state transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MEMONSTATE		RESERVED							MEMRETSTATE	RESERVED					LOGICRETSTATE	POWERSTATE	

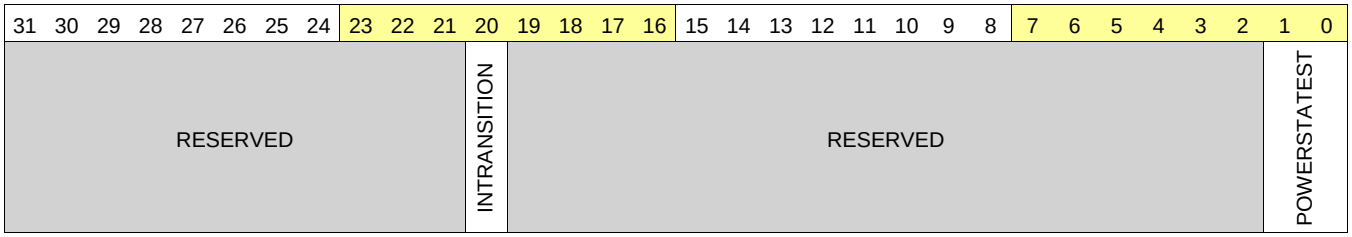
Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17:16	MEMONSTATE	Memory state when ON 0x3: Memory is always ON when domain is ON.	R	0x3
15:9	RESERVED	Reserved: keep at reset value.	R	0x00
8	MEMRETSTATE	Memory state when RETENTION 0x1: Memory is always retained when domain is in RETENTION state.	R	0x1
7:3	RESERVED	Reserved: keep at reset value.	R	0x00
2	LOGICRETSTATE	Logic state when domain is RETENTION 0x0: Reserved 0x1: Logic is always retained when domain is in RETENTION state.	R/W	0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.9.7 PM_PWSTST_PER

Table 4-222. PM_PWSTST_PER

Address Offset	0x0000 00E4		
Physical Address	0x4830 70E4	**Instance**	PER_PRM
Description	This register provides a status on the power state transition of the PERIPHERAL domain.		
Type	R		



Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: PERIPHERAL domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x00001
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.9.8 PM_PREPWSTST_PER

Table 4-223. PM_PREPWSTST_PER

Address Offset	0x0000 00E8		
Physical Address	0x4830 70E8	**Instance**	PER_PRM
Description	This register provides a status on the PERIPHERAL domain previous power state. It indicates the state entered during the last sleep transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													LASTLOGICSTATEENTERED	LASTPOWERSTATEENTERED	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved	R	0x00000000
2	LASTLOGICSTATEENTERED	Last logic and L1 Cache state entered 0x0:Reserved 0x1: PER domain logic and L1 Cache was previously ON	R/W	0x0
1:0	LASTPOWERSTATEENTERED	Last power state entered 0x0: Reserved 0x1: PERIPHERAL domain was previously in RETENTION 0x2: PERIPHERAL domain was previously INACTIVE 0x3: PERIPHERAL domain was previously ON	R/W	0x0

4.12.2.10 EMU_PRM Register Descriptions

4.12.2.10.1 RM_RSTST_EMU

Table 4-224. RM_RSTST_EMU

Address Offset	0x0000 0058		
Physical Address	0x4830 7158	**Instance**	EMU_PRM
Description	This register logs the different reset sources of the EMU domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: MPU domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: MPU domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.11 Global_Reg_PRM Register Descriptions

4.12.2.11.1 PRM_RSTCTRL

Table 4-225. PRM_RSTCTRL

Address Offset	0x0000 0050		
Physical Address	0x4830 7250	**Instance**	Global_Reg_PRM
Description	Global software and DPLL3 reset control. This register is auto-cleared. Only write 1 is possible. A read returns 0 only.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													RST_DPLL3	RST_GS	RESERVED

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	RST_DPLL3	DPLL3 software reset control. This bit is reset only upon a global cold source of reset. 0x0: DPLL3 software reset is cleared. 0x1: Asserts the DPLL3 software reset and induces a global cold reset on the whole chip. The software must ensure the SDRAM is properly put in sef-refresh mode before applying this reset.	RW	0x0
1	RST_GS	Global software reset control. This bit is reset upon any global source of reset (warm and cold). 0x0: Global software reset is cleared. 0x1: Asserts a global software reset.	RW	0x0
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.11.2 PRM_RSTTIME

Table 4-226. PRM_RSTTIME

Address Offset	0x0000 0054		
Physical Address	0x4830 7254	**Instance**	Global_Reg_PRM
Description	Reset duration control.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RSTTIME1							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000010
7:0	RSTTIME1	(Global) reset duration 1 (number of Func_32k.clk clock cycles)	RW	0x06

4.12.2.11.3 PRM_RSTST

Table 4-227. PRM_RSTST

Address Offset	0x0000 0058		
Physical Address	0x4830 7258	**Instance**	Global_Reg_PRM
Description	This register logs the global reset sources. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					ICECRUSHER_RST	ICEPICK_RST	RESERVED	EXTERNAL_WARM_RST	SECURE_WD_RST	MPU_WD_RST	SECURITY_VIOL_RST	RESERVED	GLOBAL_SW_RST	GLOBAL_COLD_RST

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Reserved: keep at reset value.	R	0x000000
10	ICECRUSHER_RST	IceCrusher reset event. This is a source of warm reset initiated by the emulation. Read 0x0: No IceCrusher reset Write 0x0: Status bit unchanged Read 0x1: IceCrusher reset occurred Write 0x1: Status bit is cleared to 0	RW	0x0
9	ICEPICK_RST	IcePick reset event. This is a source of warm reset initiated by the emulation. Read 0x0: No IcePick reset Write 0x0: Status bit unchanged Read 0x1: IcePick reset occurred Write 0x1: Status bit is cleared to 0	RW	0x0
8:7	RESERVED	Reserved: keep at reset value.	RW	0x0
6	EXTERNAL_WARM_RST	External warm reset event Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: Global external warm reset occurred Write 0x1: Status bit is cleared to 0	RW	0x0
5	SECURE_WD_RST	Secure watchdog reset event Read 0x0: No security watchdog reset Write 0x0: Status bit unchanged Read 0x1: Security watchdog reset has occurred Write 0x1: Status bit is cleared to 0	RW	0x0
4	MPU_WD_RST	MPU watchdog reset event Read 0x0: No MPU watchdog reset Write 0x0: Status bit unchanged Read 0x1: MPU wachtdog reset occurred Write 0x1: Status bit is cleared to 0	RW	0x0

Bits	Field Name	Description	Type	Reset
3	SECURITY_VIOL_RST	Security violation reset event Read 0x0: No security violation reset Write 0x0: Status bit unchanged Read 0x1: Security violation reset has occurred Write 0x1: Status bit is cleared to 0	RW	0x0
2	RESERVED	Reserved: keep at reset value.	R	0x0
1	GLOBAL_SW_RST	Global software reset event Read 0x0: No global software reset Write 0x0: Status bit unchanged Read 0x1: Global software reset occurred Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBAL_COLD_RST	Power-up (cold) reset event Read 0x0: No power-on reset Write 0x0: Status bit unchanged Read 0x1: Power-on reset occurred Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.11.4 PRM_CLKSRC_CTRL

Table 4-228. PRM_CLKSRC_CTRL

Address Offset	0x0000 0070		
Physical Address	0x4830 7270	**Instance**	Global_Reg_PRM
Description	This register provides control over the device source clock.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								SYSCLKDIV		RESERVED	AUTOEXTCLKMODE		RESERVED	SYSCLKSEL	

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved: keep at reset value.	R	0x000000
7:6	SYSCLKDIV	This field controls the system clock input divider 0x0: reserved 0x1: Syst_clk is external clock / 1 0x2: Syst_clk is external clock / 2 0x3: reserved	RW	0x1
5	RESERVED	Reserved: keep at reset value.	R	0x0
4:3	AUTOEXTCLKMODE	This field allows to control the external clock request (CLKREQ) and the oscillator 0x0: CLKREQ is kept asserted or the oscillator is always active (in master mode) 0x1: CLKREQ is de-asserted or the oscillator is put in power-down mode (in master mode) when all the voltages domains are SLEEP, RETENTION or OFF states. 0x2: CLKREQ is de-asserted or the oscillator is put in power-down mode (in master mode) when all the voltages domains are RETENTION or OFF states. 0x3: CLKREQ is de-asserted or the oscillator is put in power-down mode (in master mode) only when all the voltage domains are in OFF states.	RW	0x0
2	RESERVED	Reserved: keep at reset value.	R	0x0
1:0	SYSCLKSEL	This field reflects the mode of the oscillator. It is automatically set accordingly to the external tied-off configuration and its value is insignificant before the release of the power-on reset. 0x0: Bypass mode: the system clock is issued from an external square clock source 0x1: Oscillator mode: the system clock is issued from an external quartz 0x2: Reserved 0x3: Unknown state (not know before release of the power-on reset)	R	0x3

4.12.2.11.5 PRM_OBS

Table 4-229. PRM_OBS

Address Offset	0x0000 0080		
Physical Address	0x4830 7280	**Instance**	Global_Reg_PRM
Description	This register logs the observable signals (18 bits). This register is intended to be read through the debugger interface when the device is in standby mode.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																OBS_BUS															

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Read is undefined.	R	0x0000
17:0	OBS_BUS	Indicates the current value on the observable bus.	R	0x00000

4.12.2.11.6 PRM_CLKSETUP

Table 4-230. PRM_CLKSETUP

Address Offset	0x0000 0098		
Physical Address	0x4830 7298	**Instance**	Global_Registers_PRM
Description	This register allows setting the setup time of the oscillator system clock (sys_clk), based on number of 32 kHz clock cycles.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																SETUP_TIME															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved: keep at reset value.	R	0x0000
15:0	SETUP_TIME	Number of 32kHz clock cycles for the SETUP duration	R/W	0x0000

4.12.2.11.7 PRM_POLCTRL

Table 4-231. PRM_POLCTRL

Address Offset	0x0000 009C		
Physical Address	0x4830 729C	**Instance**	Global_Registers_PRM
Description	This register allows setting the polarity of device outputs control signals.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													CLKOUT_POL	CLKREQ_POL	RESERVED

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000001
2	CLKOUT_POL	Controls the external output clock polarity when disabled 0x0: sys_clkout is gated low when inactive 0x1: sys_clkout is gated high when inactive	R/W	0x0
1	CLKREQ_POL	Controls the polarity of the SYS_CLKREQ signal 0x0: SYS_CLKREQ is active low 0x1: SYS_CLKREQ is active high	R/W	0x1
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.12 NEON_PRM Register Descriptions

4.12.2.12.1 RM_RSTST_NEON

Table 4-232. RM_RSTST_NEON

Address Offset	0x0000 0058		
Physical Address	0x4830 7358	**Instance**	NEON_PRM
Description	This register logs the different reset sources of the NEON domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														GLOBALWARM_RST	GLOBALCOLD_RST

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: NEON domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: NEON domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.12.2 PM_WKDEP_NEON

Table 4-233. PM_WKDEP_NEON

Address Offset	0x0000 00C8		
Physical Address	0x4830 73C8	**Instance**	NEON_PRM
Description	This register allows enabling or disabling the wake-up of the NEON domain upon another domain wakeup.		
Type	RW		

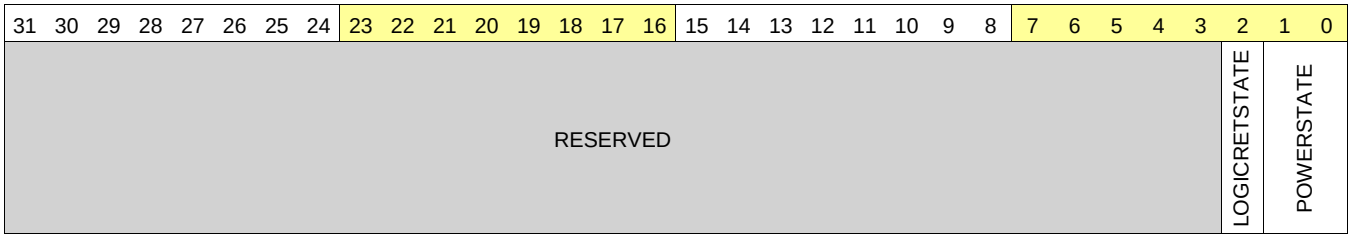
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														EN_MPU	RESERVED

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	EN_MPU	MPU domain dependency 0x0: NEON domain is independent of MPU domain wake-up. 0x1: NEON domain is woken-up upon MPU domain wake-up.	RW	0x1
0	RESERVED	Reserved: keep at reset value.	R	0x0

4.12.2.12.3 PM_PWSTCTRL_NEON

Table 4-234. PM_PWSTCTRL_NEON

Address Offset	0x0000 00E0		
Physical Address	0x4830 73E0	**Instance**	NEON_PRM
Description	This register controls the NEON domain power state transition.		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved: keep at reset value.	R	0x00000000
2	LOGICRETSTATE	Logic state when RETENTION 0x1: Logic is always retained when domain is in RETENTION state.		0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.12.4 PM_PWSTST_NEON

Table 4-235. PM_PWSTST_NEON

Address Offset	0x0000 00E4		
Physical Address	0x4830 73E4	**Instance**	NEON_PRM
Description	This register provides a status on the power state transition of the NEON domain.		
Type	R		

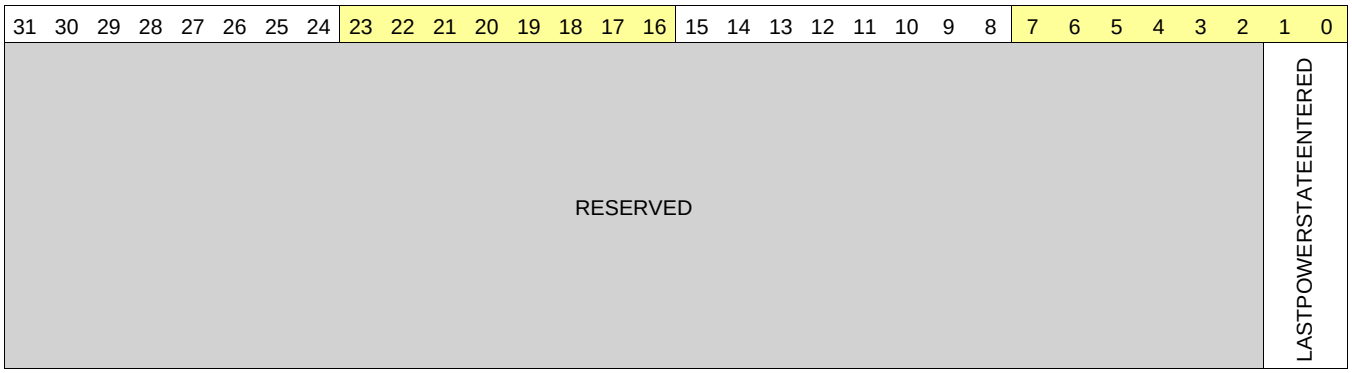
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											INTRANSITION	RESERVED																		POWERSTATEST	

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: NEON domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x00000
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.12.5 PM_PREPWSTST_NEON

Table 4-236. PM_PREPWSTST_NEON

Address Offset	0x0000 00E8		
Physical Address	0x4830 73E8	**Instance**	NEON_PRM
Description	This register provides a status on the NEON domain previous power state. It indicates the state entered during the last sleep transition.		
Type	R/W		



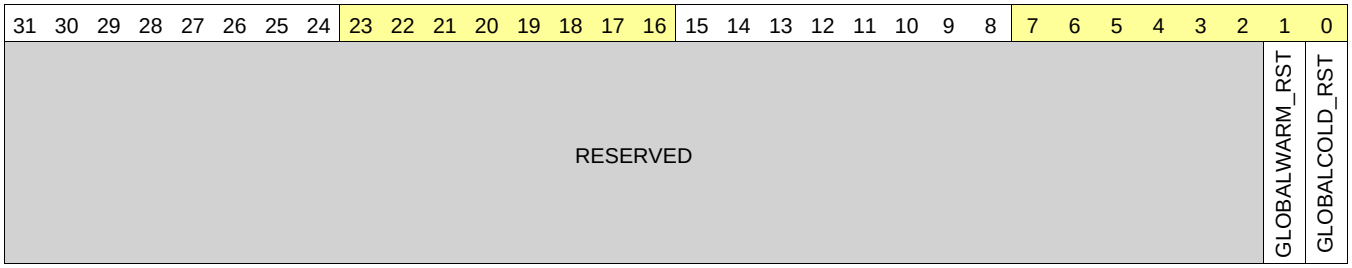
Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	LASTPOWERSTATEENTERED	Last power state entered	R/W	0x0
		0x0: Reserved		
		0x1: NEON domain was previously in RETENTION		
		0x2: NEON domain was previously INACTIVE		
		0x3: NEON domain was previously ON		

4.12.2.13 USBHOST_PRM Register Descriptions

4.12.2.13.1 RM_RSTST_USBHOST

Table 4-237. RM_RSTST_USBHOST

Address Offset	0x0000 0058		
Physical Address	0x4830 7458	**Instance**	USBHOST_PRM
Description	This register logs the different reset sources of the USB HOST domain. Each bit is set upon release of the domain reset signal. Must be cleared by software.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1	GLOBALWARM_RST	Global warm reset Read 0x0: No global warm reset Write 0x0: Status bit unchanged Read 0x1: USB HOST domain has been reset upon a global warm reset Write 0x1: Status bit is cleared to 0	RW	0x0
0	GLOBALCOLD_RST	Global cold reset Read 0x0: No global cold reset Write 0x0: Status bit unchanged Read 0x1: USB HOST domain has been reset upon a global cold reset Write 0x1: Status bit is cleared to 0	RW	0x1

4.12.2.13.2 PM_WKEN_USBHOST

Table 4-238. PM_WKEN_USBHOST

Address Offset	0x0000 00A0		
Physical Address	0x4830 74A0	**Instance**	USBHOST_PRM
Description	This register allows enabling/disabling modules wake-up events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															EN_USBHOST

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	EN_USBHOST	USB HOST Wake-up enable 0x0: USB HOST wake-up is disabled 0x1: USB HOST wake-up event is enabled	RW	0x1

4.12.2.13.3 PM_MPUGRPSEL_USBHOST

Table 4-239. PM_MPUGRPSEL_USBHOST

Address Offset	0x0000 00A4		
Physical Address	0x4830 74A4	**Instance**	USBHOST_PRM
Description	This register allows selecting the group of modules that wake-up the MPU.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															GRPSEL_USBHOST

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	GRPSEL_USBHOST	Select the USBHOST in the MPU wake-up events group 0x0: USBHOST is not attached to the MPU wake-up events group. 0x1: USBHOST is attached to the MPU wake-up events group.	RW	0x1

4.12.2.13.4 PM_WKST_USBHOST

Table 4-240. PM_WKST_USBHOST

Address Offset	0x0000 00B0		
Physical Address	0x4830 74B0	**Instance**	USBHOST_PRM
Description	This register logs the modules wake-up events. Must be cleared by software. It prevents further domain transition if it is not cleared.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															ST_USBHOST

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved: keep at reset value.	R	0x00000000
0	ST_USBHOST	USB HOST Wake-up status	RW	0x0
		0x0: USB HOST wake-up has not occurred or was masked.		
		0x0: Status bit unchanged		
		0x1: USB HOST wake-up has occurred.		
		0x1: Status bit is cleared to 0		

4.12.2.13.5 PM_WKDEP_USBHOST

Table 4-241. PM_WKDEP_USBHOST

Address Offset	0x0000 00C8		
Physical Address	0x4830 74C8	**Instance**	USBHOST_PRM
Description	This register allows enabling or disabling the wake-up of the USB HOST domain upon another domain wakeup.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											EN_WKUP	RESERVED		EN_MPU	EN_CORE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved: keep at reset value.	R	0x0000000
4	EN_WKUP	WAKEUP domain dependency 0x0: USB HOST domain is independent of WKUP domain wake-up event. 0x1: USB HOST domain is woken-up upon WKUP domain wake-up event.	RW	0x1
3:2	RESERVED	Reserved: keep at reset value.	R	0x1
1	EN_MPU	MPU domain dependency 0x0: USB HOST domain is independent of MPU domain wake-up. 0x1: USB HOST domain is woken-up upon MPU domain wake-up.	RW	0x1
0	EN_CORE	CORE domain dependency 0x0: USB HOST domain is independent of CORE domain wake-up. 0x1: USB HOST domain is woken-up upon CORE domain wake-up.	RW	0x1

4.12.2.13.6 PM_PWSTCTRL_USBHOST

Table 4-242. PM_PWSTCTRL_USBHOST

Address Offset	0x0000 00E0		
Physical Address	0x4830 74E0	**Instance**	USBHOST_PRM
Description	This register controls the USB HOST domain power state transition.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MEMONSTATE		RESERVED							MEMRETSTATE	RESERVED					LOGICRETSTATE	POWERSTATE	

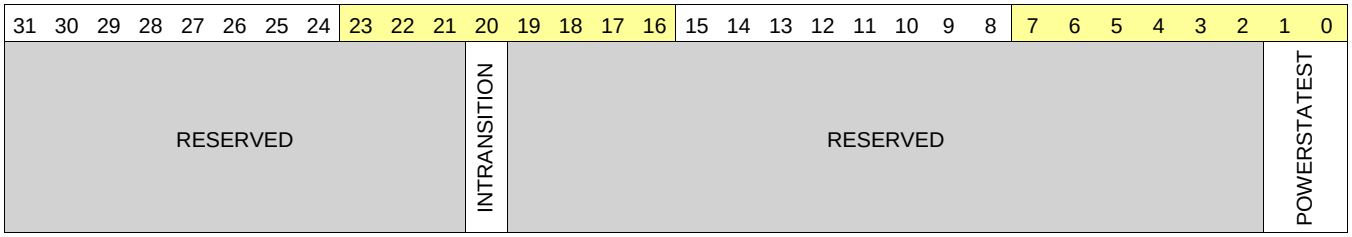
Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved: keep at reset value.	R	0x0000
17:16	MEMONSTATE	Memory state when ON 0x3: Memory is always ON when domain is ON.	R	0x3
15:9	RESERVED	Reserved: keep at reset value.	R	0x00
8	MEMRETSTATE	Memory state when RETENTION 0x1: Memory is always retained when domain is in RETENTION state.	R	0x1
7:3	RESERVED	Reserved: keep at reset value.	R	0x00
2	LOGICRETSTATE	Logic state when RETENTION 0x1: Logic is always retained when domain is in RETENTION state.	R	0x1
1:0	POWERSTATE	Power state control 0x0: Reserved 0x1: RETENTION state 0x2: Reserved 0x3: ON State	R/W	0x3

NOTE: This register should not be programmed with reserved values of bit fields for proper functioning of power state control.

4.12.2.13.7 PM_PWSTST_USBHOST

Table 4-243. PM_PWSTST_USBHOST

Address Offset	0x0000 00E4		
Physical Address	0x4830 74E4	**Instance**	USBHOST_PRM
Description	This register provides a status on the power state transition of the USB HOST domain.		
Type	R		

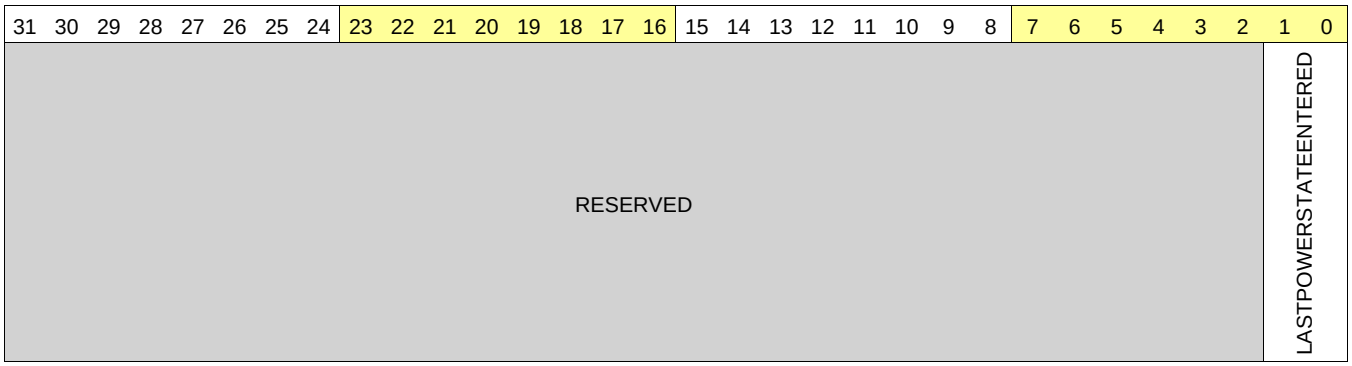


Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	INTRANSITION	Domain transition status 0x0: No transition 0x1: USB HOST domain transition is in progress.	R	0x0
19:2	RESERVED	Reserved	R	0x00000
1:0	POWERSTATEST	Current power state status 0x0: Reserved 0x1: Reserved 0x2: Power domain is INACTIVE 0x3: Power domain is ON	R	0x3

4.12.2.13.8 PM_PREPWSTST_USBHOST

Table 4-244. PM_PREPWSTST_USBHOST

Address Offset	0x0000 00E8		
Physical Address	0x4830 74E8	**Instance**	USBHOST_PRM
Description	This register provides a status on the USBHOST domain previous power state. It indicates the state entered during the last sleep transition.		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reserved: keep at reset value.	R	0x00000000
1:0	LASTPOWERSTATEENTERED	Last power state entered	R/W	0x0
		0x0: Reserved		
		0x1: USB HOST domain was previously in RETENTION		
		0x2: USB HOST domain was previously INACTIVE		
		0x3: USB HOST domain was previously ON		

4.13 Revision History

Table 4-245 lists the changes made since the previous version of this document.

Table 4-245. Document Revision History

Reference	Additions/Modifications/Deletions
Section 4.5.5.10	Updated 2nd paragraph.
Section 4.5.9.1	Changed section.
Figure 4-24	Corrected figure.
Figure 4-25	Added figure.
Section 4.7.4.1.3	Deleted 2nd paragraph.
Section 4.7.4.1.4	Added section.
Table 4-25	Updated table.
Section 4.7.7.1	Updated 1st sentence, 3rd paragraph.
Table 4-45	Updated table.
Figure 4-65	Corrected figure.
Section 4.12.1.4.3	Changed bits 28:26 to reserved.
Section 4.12.1.4.4	Changed all bits to reserved.
Section 4.12.1.4.6	Changed bits 28:26 to reserved.
Section 4.12.1.4.7	Changed all bits to reserved.
Section 4.12.1.4.9	Changed bits 28:26 to reserved.
Section 4.12.1.4.10	Changed all bits to reserved.
	Changed Smart Reflex bit 6 to reserved.
	Changed Smart Reflex bit 7 to reserved.
Section 4.12.2.3.6	Added Register.
Section 4.12.2.3.8	Added Register.
Section 4.12.2.4.6	Added Register.
Section 4.12.2.4.8	Added Register.
Section 4.12.2.5.3	Added Register.
Section 4.12.2.5.5	Added Register.
	Changed Smart Reflex bit 7 to reserved.
	Changed Smart Reflex bit 7 to reserved.
	Changed Smart Reflex bit 7 to reserved.
Section 4.12.2.8.4	Added Register.
Section 4.12.2.8.6	Added Register.
Section 4.12.2.9.6	Added Register.
Section 4.12.2.9.8	Added Register.
Section 4.12.2.11.5	Updated description.
Section 4.12.2.11.6	Added Register.
Section 4.12.2.11.7	Added Register.
Section 4.12.2.12.3	Added Register.
Section 4.12.2.12.5	Added Register.
Section 4.12.2.13.6	Added Register.
Section 4.12.2.13.8	Added Register.
Section 4.12	Updated descriptions for reserved bits in multiple register description tables.

Interconnect

This chapter describes the Interconnect.

NOTE:

- The L3 interconnect is an instantiation of the **SonicsMX®** interconnect from **Sonics, Inc**.
- The L4 interconnects are instantiations of the **Sonics3220™** interconnect from **Sonics, Inc**.

SMX is an abbreviated naming for SonicsMX.

Topic **Page**

5.1 Interconnect Overview

This chapter gives information about all modules and features in the high-tier device. To flag interconnect response being blocked, the time-out of target agents attached to unavailable modules can be enabled with the lowest setting.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

5.1.1 Terminology

The following terminology is critical to understanding the interconnect:

- Initiator: Module able to initiate read and write requests to the chip interconnect (typically: processors, DMA, etc.).
- Target: Unlike an initiator, a target module cannot generate read/write requests to the chip interconnect, but it can respond to these requests. However, it may generate interrupts or a DMA request to the system (typically: peripherals, memory controllers).

NOTE: A module can have several separate ports; therefore, a module can be both an initiator and a target.

- Agent: Each connection of one module to one interconnect is done using an agent, which is an adaptation (sometimes configurable) between the module and the interconnect. A target module is connected by a target agent (TA), and an initiator module is connected by an initiator agent (IA).
- OCP: Open-core protocol (www.ocpip.org) is point-to-point standard protocol between one master port and one slave port.
- OCP master port: Port that can generate OCP commands. An initiator includes at least one master port.
- OCP slave port: Port that responds to OCP commands. A target includes one slave port.
- Interconnect: The decoding, routing, and arbitration logic that enables the connection between multiple initiator modules and multiple target modules connected on it.
- Register target (RT): Special TA used to access the interconnect internal configuration registers.
- Dataflow signal: Any OCP signal that is part of a clearly identified OCP transfer or dataflow (typically: command, address, byte enables, etc.). The signal behavior is defined by the OCP protocol semantics.
- Sideband signal: Any OCP signal whose behavior is not associated to a precise OCP transaction or dataflow. The OCP standard does not define specific semantics for these signals.icnt
- Out-of-band error: Any OCP signal whose behavior is associated to an error-reporting scheme of the device, as opposed to in-band errors.

NOTE: Interrupt requests and DMA requests are not routed by the interconnect in the device.

- Firewall: A programmable security feature integrated in a target agent for each module requiring a secure data path. A firewall can be configured using three criteria:
  - Initiator requesting access
  - Address space access
  - Type of access
- Thread: Logical entities that allow to have separate independent data flows on a single port.
- Multithreaded ports: A physical port able to simultaneously handle several outstanding transactions. On a multithreaded port, one physical channel (port) is used concurrently for several logical channels (threads). The transfer in each thread must remain in order with respect to each other, but the order between threads can change between requests and responses. Thread management is used for performance optimization purposes and is automatically handled by the system.
- ConnID: Any transaction in the system interconnect is tagged by an in-band qualifier ConnID, which

uniquely identifies the initiator at a given interconnect point. A ConnID is transmitted inband with the request and is used for security and error-logging mechanism.

- Firewall comparison mechanism: A comparison made in the firewall between access in-band qualifiers and access permissions that are programmed in the firewall configuration registers. If the comparison is successful, access is allowed; otherwise, access is denied.
- MCmd qualifier: Command bus that indicates the type of transfer requested. Table 5-1 lists the commands encoded.

Table 5-1. MCmd Qualifier Description

MCmd[2:0]	Transaction Type
0 0 0	Idle
0 0 1	Write
0 1 0	Read
0 1 1	ReadEx
1 0 0	Not used
1 0 1	Write nonposted
1 1 0	Not used
1 1 1	Not used

- MReqInfo qualifier: Four MReqInfo qualifiers describe the access during the use of the firewall comparison mechanism, as described in Table 5-2.

Table 5-2. MReqInfo Qualifier Description

Qualifiers	Description
MReqType	0: Data access
	1: Opcode fetch
MReqSupervisor	0: User mode
	1: Supervisor mode
MReqDebug	0: Functional access
	1: Debug access
MReqSecure	0: Public transaction
	1: Secure transaction(1)

(1) In GP device MReqSecure = 1 (Secure value), is not supported.

- L3_PM_REQ_INFO_PERMISSION_i: Register that configures the combination of the MReqInfo, allowing access permission to the TM based on the MReqInfo in-band qualifier values.
- SError: Target that indicates an error condition to the initiator.
- SResp qualifier: Response from the target to the initiator concerning the transaction.

Table 5-3. SResp Qualifier Description

SResp[1:0]	Description
0 0	No response
0 1	Data valid/accept
1 0	Not used
1 1	Error

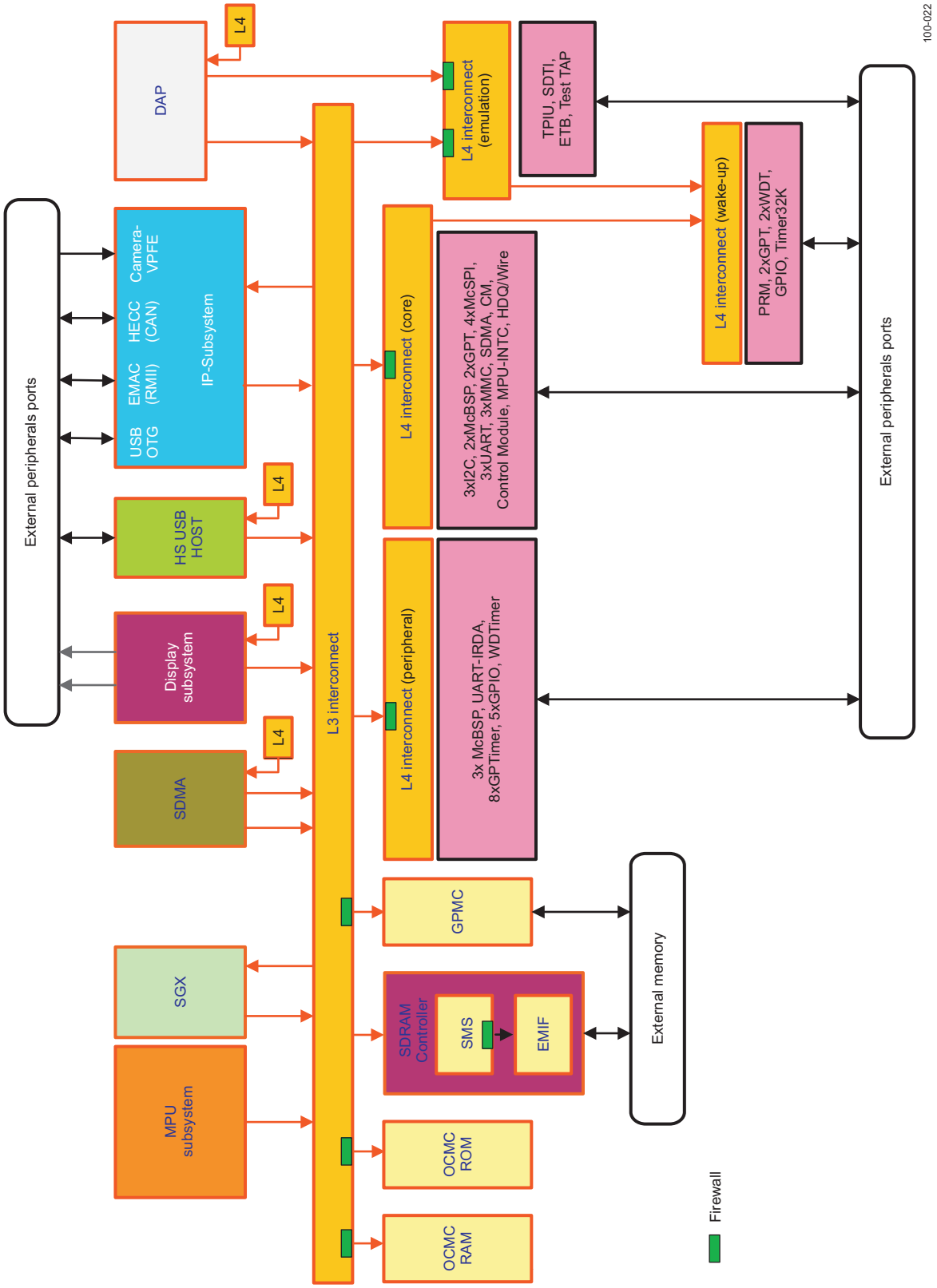
5.1.2 Architecture Overview

The device memory hierarchy includes four levels:

- L1 is internal to the CPUs. It concerns data exchange with the internal Level1 cache memory subsystem, and it is the closest memory to the microprocessor unit (MPU) core.
- L2 is included in the MPU subsystem.

- The chip-level interconnect consists of one L3 interconnect and four L4 interconnects. It enables communication among the modules and subsystems in the device. Figure 5-1 shows an overview of the L3 and L4 interconnect architecture.
  - L3 handles many types of data transfers, especially exchanges with system-on-chip/external memories. L3 transfers data with a maximum width of 64 bits from the initiator to the target. The L3 interconnect is a little-endian platform
  - L4 is composed of the L4-Core, L4-Per, L4-Wakeup, and L4-Emu interconnects and handles data transfers to peripherals. It supports 32-bit data width transfer and is optimized to support the interconnection of many peripheral targets. These backplanes assume little-endian transactions for narrower targets (8-bit, 16-bit) when doing data packing and unpacking.

Figure 5-1. Interconnect Architecture Overview



Modules are connected to the interconnect through an IA for the initiator module and a TA for target modules. Each module/subsystem connection is statically configured to tune the access depending on the characteristics of the module.

To secure a data path, some TAs include configurable firewalls (FWs) for security purposes. A firewall restricts or filters the accesses allowed to an initiator according to different access criteria. The firewalls can usually be configured by software.

The L3 and L4 interconnect default setting is fully functional; it enables all possible functional data paths and a minimal default security setting. However, it is possible to modify the interconnect parameters to fit user expectations.

5.1.3 Module Distribution

IAs and TAs provide the interface to connect the different modules and the interconnect.

Table 5-4 through Table 5-13 list the device modules, subsystems, and associated agents. The agents are listed for each interconnect domain:

- L3 initiator and target agents
- L4-Core initiator and target agents
- L4-Per initiator and target agents
- L4-Emu initiator and target agents
- L4-Wakeup initiator and target agents

5.1.3.1 L3 Interconnect Agents

Table 5-4 and Table 5-5 list the IAs and TAs, respectively, of the L3 interconnect.

Table 5-4. L3 Initiator Agents

Module Name	Description
MPU SS	MPU subsystem port
Display SS	Display subsystem port
IPSS	IP-Subsystem initiator port (Camera-VPFE, USBOTG, and EMAC)
SGX SS	Graphics subsystem port
sDMA read	System DMA read port
sDMA write	System DMA write port
High-Speed (HS) USB Host	Universal serial bus High-Speed port Host Controller
DAP	Debug access port (JTAG/Emulation access to system resources)

Table 5-5. L3 Target Agents

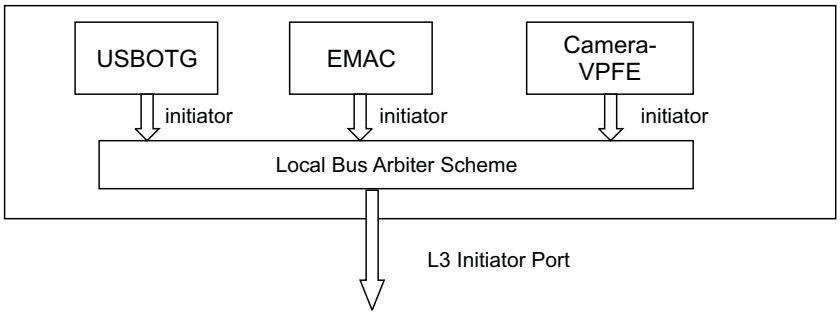
Module Name	Description
SMS	SDRAM memory scheduler port
GPMC	General-purpose memory controller (for flash memory, SRAM, SROM, etc.) port
OCM-ROM	On-chip memory ROM port
OCM-RAM	On-chip memory RAM port
SGX	Graphics subsystem port
IPSS	Register target port for IPSS
RT	Register target port to configure L3
L4-Core	Port for L4-Core interconnect
L4-Peripherals	Port for L4-Per interconnect
L4-Emu	Port for L4-Emu interconnect

NOTE: The IP-Subsystem (abbreviated as IPSS) integrates the following modules:

- Camera-VPFE
- EMAC
- USBOTG
- CAN-HECC

The camera-VPFE, EMAC and USBOTG are initiators on L3 and share one L3 initiator port as shown in Figure 5-2.

Figure 5-2. L3 Port Initiators



All IPSS modules (Camera-VPFE, EMAC, USBOTG, and CAN-HECC) have a register target port on L3.

For more details on the register target module, see Section 5.4.2, *Register Target*.

5.1.3.2 L4-Core Agents

Table 5-6. L4-Core Initiator Agent

Module Name	Description
L3 interconnect	L3 interconnect port

NOTE: A unique L3 port is used for communication with the L4-Core. For the list of initiators allowed to access the L4 Core peripherals, see Table 5-14.

Table 5-7. L4-Core Target Agents

Module Name	Description
Display subsystem	Display subsystem configuration port
High-Speed (FS) USB Host	Universal serial bus High-Speed port Host controller
UART1	Universal asynchronous receiver transmitter port 1
UART2	Universal asynchronous receiver transmitter port 2
UART4	Universal asynchronous receiver transmitter port 4
I2C1	Multimaster interintegrated circuit 1
I2C2	Multimaster interintegrated circuit 2
I2C3	Multimaster interintegrated circuit 3
McBSP1	Multichannel buffered serial port 1
McBSP5	Multichannel buffered serial port 5
GPTIMER10	General-purpose timer 10
GPTIMER11	General-purpose timer 11
SPI1	Serial peripheral interface 1

Table 5-7. L4-Core Target Agents (continued)

Module Name	Description
SPI2	Serial peripheral interface 2
MMCHS1	Multimedia memory controller SDIO 1
MMCHS2	Multimedia memory controller SDIO 2
MMCHS3	Multimedia memory controller SDIO 3
HDQ/1-Wire	Single wire serial link low rate
MG	MagicGate®
SPI1	Serial peripheral interface 1
SPI2	Serial peripheral interface 2
SPI3	Serial peripheral interface 3
SPI4	Serial peripheral interface 4
sDMA	System DMA controller
L4-Wakeup	L4-Wakeup interconnect
CM	Clock manager
SCM	System control module

5.1.3.3 L4-Per Agents

Table 5-8. L4-Per Initiator Agent

Module Name	Description
L3 interconnect	L3 interconnect port

NOTE: A unique L3 port is used for communication with L4-Per. For the list of initiators allowed to access the L4-Per peripherals, see Table 5-14.

Table 5-9. L4-Per Target Agents

Module Name	Description
UARTIrDA	Universal asynchronous receiver/transmitter and infrared data association port
McBSP2	Multichannel buffered serial port 2
McBSP3	Multichannel buffered serial port 3
GPTIMER2	General-purpose timer 2
GPTIMER3	General-purpose timer 3
GPTIMER4	General-purpose timer 4
GPTIMER5	General-purpose timer 5
GPTIMER6	General-purpose timer 6
GPTIMER7	General-purpose timer 7
GPTIMER8	General-purpose timer 8
GPTIMER9	General-purpose timer 9
GPIO2	General-purpose I/O 2
GPIO3	General-purpose I/O 3
GPIO4	General-purpose I/O 4
GPIO5	General-purpose I/O 5
GPIO6	General-purpose I/O 6

5.1.3.4 L4-Emu Agents

Table 5-10. L4-Emu Initiator Agents

Module Name	Description
L3 interconnect	L3 interconnect port
DAP	DAP port

NOTE: The L3 and DAP ports are used for communication with L4-Emu. For the list of initiators allowed to access the L4-Emu peripherals, see Table 5-14.

Table 5-11. L4-Emu Target Agents

Module Name	Description
L4-Wakeup	L4 wake-up interconnect
SDTI	System debug trace interface
ETB	Embedded trace buffer
TPIU	Trace port interface unit
DAPCTL	Debug access port

5.1.3.5 L4-Wakeup Agents

Table 5-12. L4-Wakeup Initiator Agent

Module Name	Description
L4-Core interconnect	L4-Core interconnect port
L4-Emu interconnect	L4-Emulation interconnect port

NOTE: The L4-Emu and L4-Core ports are used to communicate with the L4-Wakeup. For the list of initiators allowed to access the L4-Wakeup peripherals, see Table 5-14.

Table 5-13. L4-Wakeup Target Agents

Module Name	Description
PRM	Power reset management
GPIO1	General-purpose I/O 1
GPTIMER1	General-purpose timer 1
GPTIMER12[(1)]	General-purpose timer 12
WDTIMER1[(1)]	Secure watchdog timer
WDTIMER2	MPU subsystem watchdog timer
32KTIMER	32-kHz timer

(1) Not available in GP device.

5.1.4 Connectivity Matrix

Table 5-14 lists the functional paths between the L3 interconnect initiator modules and the L3 and L4 TAs. The functional paths are indicated by the use of the following:

- Cell contains a + sign when a functional path exists.
- Cell is blank when no functional path exists.

Table 5-14. Connectivity Matrix

Initiator Ports	L4-Core Target (1)	L4-Per Target (1)	L4-Emu Target (1)	L4-Wakeup Target (1)	SMS Target	GPMC Target	OCM RAM Target	OCM ROM Target	RT Target	IPSS Target	SGX Target
MPU IA	+	+	+	+	+	+	+	+	+	+	+
SGX IA					+	+	+				
IPSS IA					+	+	+				
DSS IA					+		+				
USB Host IA					+	+	+				
sDMA RD IA	+	+	+	+	+	+	+				+
sDMA WR IA	+	+	+	+	+	+	+				+
DAP IA	+	+	+	+	+	+	+	+	+	+	+

(1) A functional data path always exists from L4 IAs (Core, Per, Emu, and Wakeup) to any L4 target module (Core, Per, Emu, and Wakeup). As a consequence, all L3 initiator modules for which a data path exists to an L4 TA can access L4 peripherals. Restrictions on peripheral access depend on the security settings and L4 protection mechanism programming. For more details, see Section 5.9.3, *L4 Security and Firewalls*.

5.2 L3 Interconnect

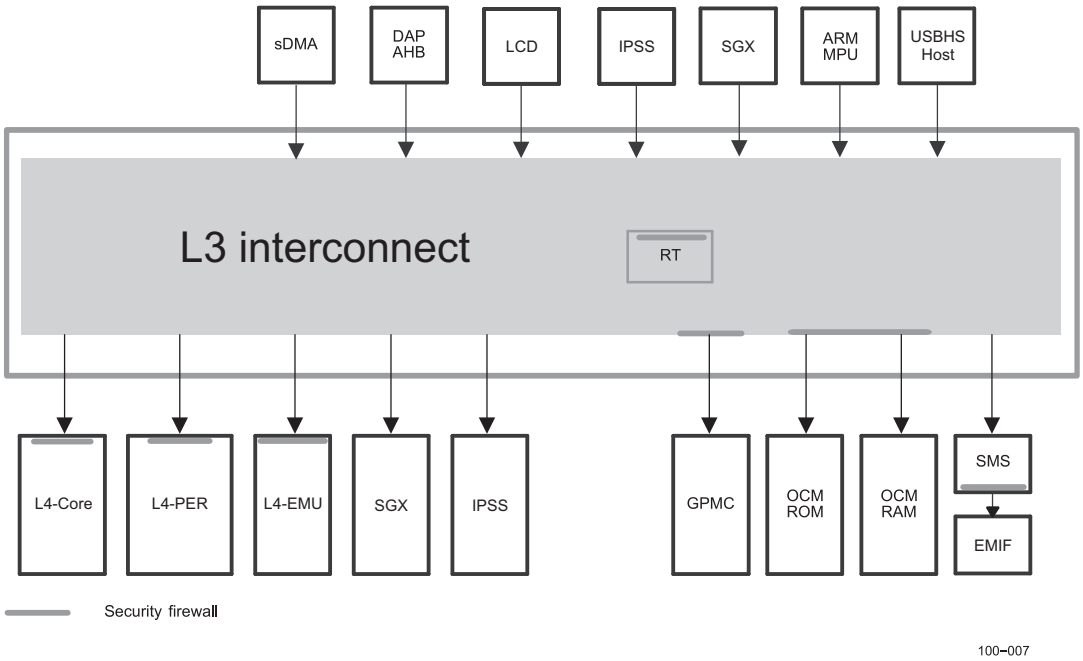
This section describes the L3 interconnect and its components. With the exception of register points, each component includes functionality for both the request and response network.

5.2.1 Overview

The L3 interconnect links cores in a flexible topology that couples low power with high performance. Innovative physical structures and advanced protocols ensure bandwidth and latency to individual IP cores, providing dedicated connections between IP cores and logical connections over a shared interconnect.

Figure 5-3 shows the L3 interconnect.

Figure 5-3. L3 Interconnect Overview



The following are the main features of the L3 interconnect:

- 64-bit multipath interconnect to eliminate on-chip bottlenecks
- Special internal target for access to L3 registers (RT)
- Guaranteed quality of service for real-time hardware operators, while maintaining optimal memory latency for MPU accesses to memory resources
- True little-endian platform
- Transaction error tracking and logging
- Built-in security features:
  - Allow access only to authorized initiator
  - Distributed region-based firewalls for system resource sharing and protection management
- Signaling support for chip-level power management infrastructure
- Two interrupt line signaling transaction error

5.3 L3 Interconnect Integration

5.3.1 Clocking, Reset, and Power-Management Scheme

5.3.1.1 Clocks

The power, reset, and clock management (PRCM) module provides the L3_ICLK as the main clock to the L3 interconnect.

In addition to L3_ICLK, four additional sample signals are provided to module agents to allow internal synchronization. The modules are as follows:

- L4-Core
- L4-Per

For more details on the L3 clock and its setting, see the *Power, Reset, and Clock Management* chapter.

Table 5-15. L3 Interconnect Clocks

Type	Name	Source	Description
Interface/functional	L3_ICLK	PRCM	Main clock for L3 interconnect

5.3.1.2 Resets

The L3 interconnect receives a single reset signal, CORE_RST, from the PRCM module. CORE_RST is the reset signal to the core power domain. (For more details, see the *Power, Reset, and Clock Management* chapter.) When asserted, CORE_RST resets the L3 internal registers. There is no software reset for the L3 interconnect.

Table 5-16. L3 Interconnect Reset

Type	Reset Domain	Source	Description
Hardware	CORE_RST	PRCM	Asynchronous reset for the entire interconnect

5.3.1.3 Power Management

As part of the system-wide power-management scheme, the L3 interconnect enters an idle state at the request of the PRCM module. (For more details, see the *Power, Reset, and Clock Management* chapter.) The L3 interconnect is always in smart-idle mode; that is, it goes into idle state after receiving the request from the PRCM module once all transfer requests are serviced. This functionality is handled by hardware. The L3 interconnect sends an acknowledge signal back to the PRCM module when it enters the idle state.

5.3.2 Hardware Requests

5.3.2.1 Interrupt Requests

Three interrupt lines are present at the boundary of the L3 interconnect (see Table 5-17). They are used for hardware error management.

Table 5-17. L3 Interconnect Hardware Requests

Type	Name	Destination	Description
Interrupt	M_IRQ_9	MPU interrupt controller for debug errors	L3 interconnect provides a mechanism to group core-detected and internal interconnect errors. See Section 5.9.4, *Error Handling*.
Interrupt	M_IRQ_10	MPU interrupt controller for application errors	

5.4 L3 Interconnect Functional Description

5.4.1 Initiator Identification

An InitiatorID is assigned to every thread on every initiator socket. The ID uniquely identifies the initiator and thread for an interconnect transfer see Table 5-18. The interconnect uses InitiatorIDs for a number of purposes, including the following:

- Initiator source identification for the protection mechanism (see Section 5.4.3, *L3 Security and Firewalls*)
- Response route generation (performed internally to the TAs)
- Firewall error logging
- L3 interconnect error logging

Table 5-18. InitiatorID Definition

Initiator	InitiatorID
HS USB Host	0
IP-Subsytem (Camera-VPFE, USBOTG, and EMAC)	2
sDMA rd	3 , 4 , 5, 6
sDMA wr	7, 8
DAP	9
SGX	13
MPU SS	23, 24, 25, 26, 27
LCD	29

5.4.2 Register Target

An RT is a specialized TA used to access L3 interconnect internal configuration registers.

RT configuration options are a subset of those available for TAs. For more details, see Section 5.6.5.

5.4.3 L3 Security and Firewalls

Security in the device relies heavily on L3 firewalls and their configuration. Nine targets are protected through the use of firewalls. The number of protected regions varies on the target, with a maximum of eight regions. Table 5-19 lists the security configuration and the number of protected regions for each target.

Table 5-19. Target Firewall and Region Configuration

Target	Firewall	Number of Regions
SMS	Included in the SMS module	See the *Memory Subsystem* chapter.
GPMC	Yes	8
OCM-RAM	Yes	8
OCM-ROM	Yes	2
IP-Subsytem (Camera-VPFE, USBOTG, EMAC and CAN-HECC)	Yes	
L4-Core	Included in the L4 module	See Section 5.7, L4 Interconnect.
L4-Wakeup	Included in the L4 module	See Section 5.7, L4 Interconnect.
L4-Emu	Included in the L4 module	See Section 5.7, L4 Interconnect.
RT	Yes	2

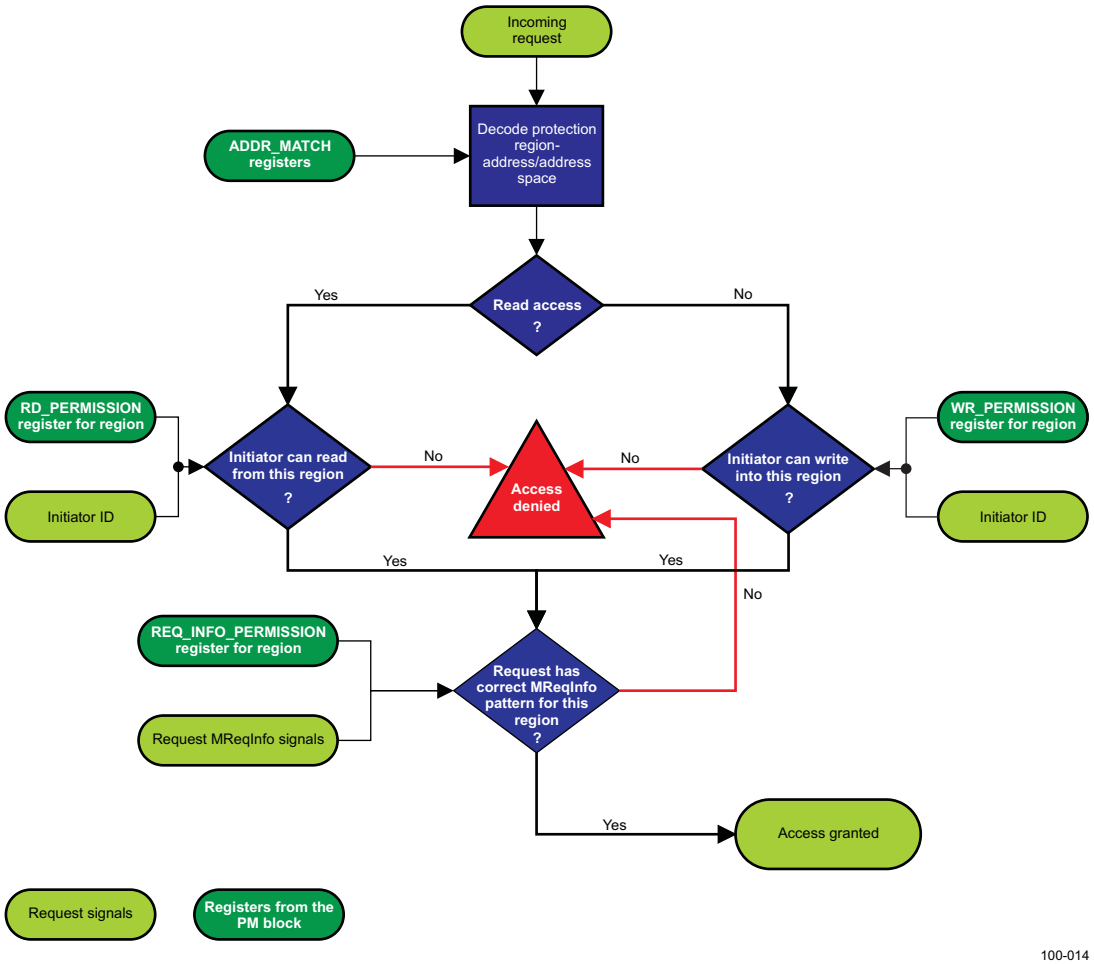
The protection mechanism designates protection regions within the address space of certain targets. Access to these regions is granted only for certain initiators, based on transaction attributes (transmitted through MReqInfo). Each protection region is characterized with several configurable attributes (base address, size, specific access rights, and priority setting).

The protection mechanism uses the following attributes of a request:

- The address field and the address space field are used to determine which region has been hit. The region ID selects one table entry of the firewall look-up table.
- The region ID points to a unique set of permission registers: Read_Permission, Write_Permission, ReqInfo_Permission.
- The Initiator ID is used to determine the permission of the initiator (read/write permission) with respect to the concerned region.
- The access types (read or write) and the transaction attributes (MReqInfo in-band qualifiers) are used to grant or reject the access.

The first check determines whether the type of incoming request (read or write) is allowed, according to the Initiator MConnID and its read and write permissions. The second check determines whether the MReqInfo bits of the incoming request are within the allowed pattern established by the MReqInfo permission bits. If both check results are positive, the request is allowed to access the target. Otherwise, the access is denied, the request is not forwarded to the target, an out-of-band error indication is reported, and an in-band error response is returned to the initiator (except for writes that were posted at the IA). Figure 5-4 shows the flow used to identify and accept a new request.

Figure 5-4. Flowchart of the Protection Mechanism



To summarize, firewalls accept or reject a request depending on the following:

- Initiator originating the request
- Command (read or write) requested
- MReqInfo bus state
- Region access in the target memory space

Software must configure the L3 firewalls properly to allow the right initiators, with the right MReqInfo access, on the well-defined size region. All the registers relative to the L3 firewalls are grouped in the protection mechanism (PM) register block.

NOTE: The PM qualifies the protection mechanism register associated with a firewall target. The targets protected by a firewall are listed in Table 5-19. These PM registers do not exist if no firewall is associated with the target.

5.4.3.1 Protection Region

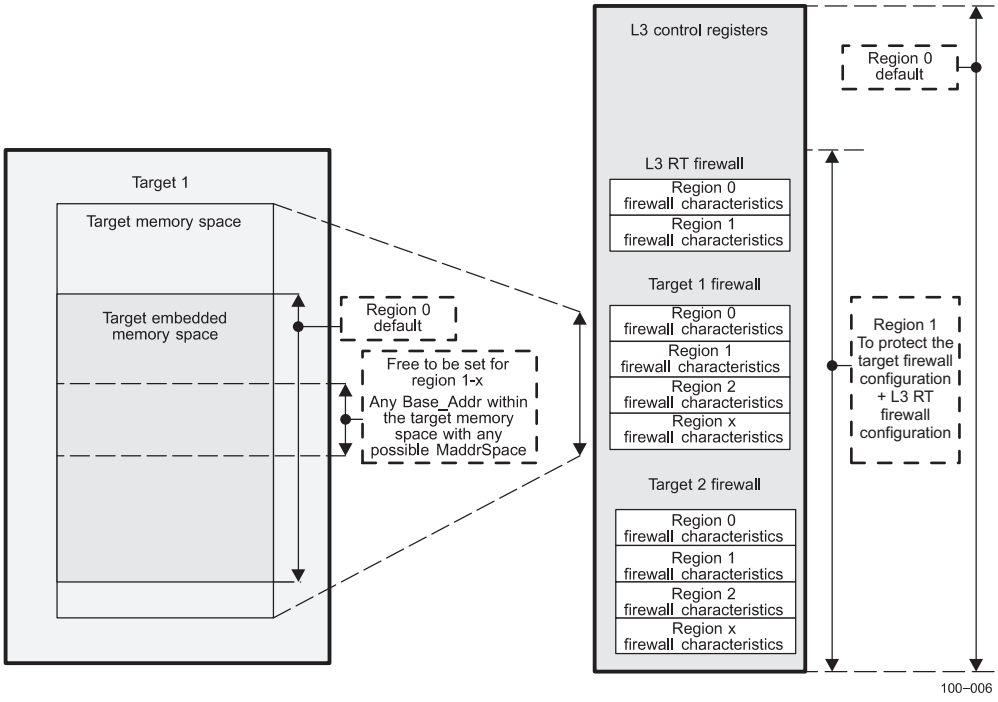
Two types of regions are distinguished in a target firewall (see Figure 5-5):

- Default region: Available in all targets; spans the entire target address range
- Normal region: Number varies in a target; they have identical capabilities

Each region has the following characteristics:

- A base address, relative to the target address itself, and an address space
- A size
- Specific access rights, as defined through the MReqInfo qualifiers
- A priority level, from 0 (lowest) to 3 (highest)

Figure 5-5. L3 Firewall Implementation



5.4.3.1.1 Default Region/Region0

Region 0 is the default region of a whole target; it spans the entire target address space.

The default region is always the lowest priority. This region is systematically overlapped when other regions are set.

The L3_PM_ADDR_MATCH_k register is not accessible. Its configuration corresponds to all the possible addresses for the target, including all of ADDR_SPACE. It is not possible to program multiple ADDR_SPACE addresses in a normal region.

5.4.3.1.2 Normal Regions

Normal regions have identical features.

A given request either maps into a specific protection region or is considered to have hit the default protection region if no normal region is hit.

The protection regions can only be configured for a memory space that is power-of-two in size and size-aligned using the SIZE bit field L3_PM_ADDR_MATCH_k [7:3] (as listed in Table 5-20). When the SIZE bit field L3_PM_ADDR_MATCH_k [7:3] is set to 0, the region is disabled.

NOTE: k denotes the region number. Depending on the target, n varies from 0 to 7.

Table 5-20 lists the size encoding for each value set in the SIZE bit field.

Table 5-20. L3 Firewall Size Parameter Definition

Possible Configuration		
Size[1]	**Region Size**	**Base_Addr**[2]
0x0	Region disabled	Any (nonsignificant)
0x1	1K-byte secure	0x0000000
		0x0000400
		0x0000800
		.
0x2	2K-byte secure	0x0000000
		0x0000800
		0x0001000
		...
...	...	...
0x17	2^(17-1)K-byte secure	
Others	Not allowed	-

(1) When the size parameter is set to 0, the region is disabled.
(2) The base address depends on user settings.

5.4.3.2 Priority Level Overview

Each L3 firewall region is prioritized. Depending on its priority level, a region can override the settings of another region.

- Region 0 is the only allowed priority 0 region (lowest priority).
- Region 1 is the only allowed priority 3 region (highest priority).
- Others regions are defined as priority 1 or 2.

CAUTION

LEVEL bitfield value of ADDR_MATCH_1 register (Region 1 firewall configuration) must be kept to his default reset value and must not be changed; otherwise, the result will be unpredictable and can create unexpected security holes or denial of service.

Protection level is defined by the LEVEL bit L3_PM_ADDR_MATCH_k , where n is greater than 2. Figure 5-6 represents the priority level with associated regions.

When an address hits two or more regions with different priority levels, the highest priority region protections are applied. The overlay region can overlap all or a part of a nonoverlay region.

CAUTION

Configuring two overlapping protection regions with the same priority level leads to undefined behavior.

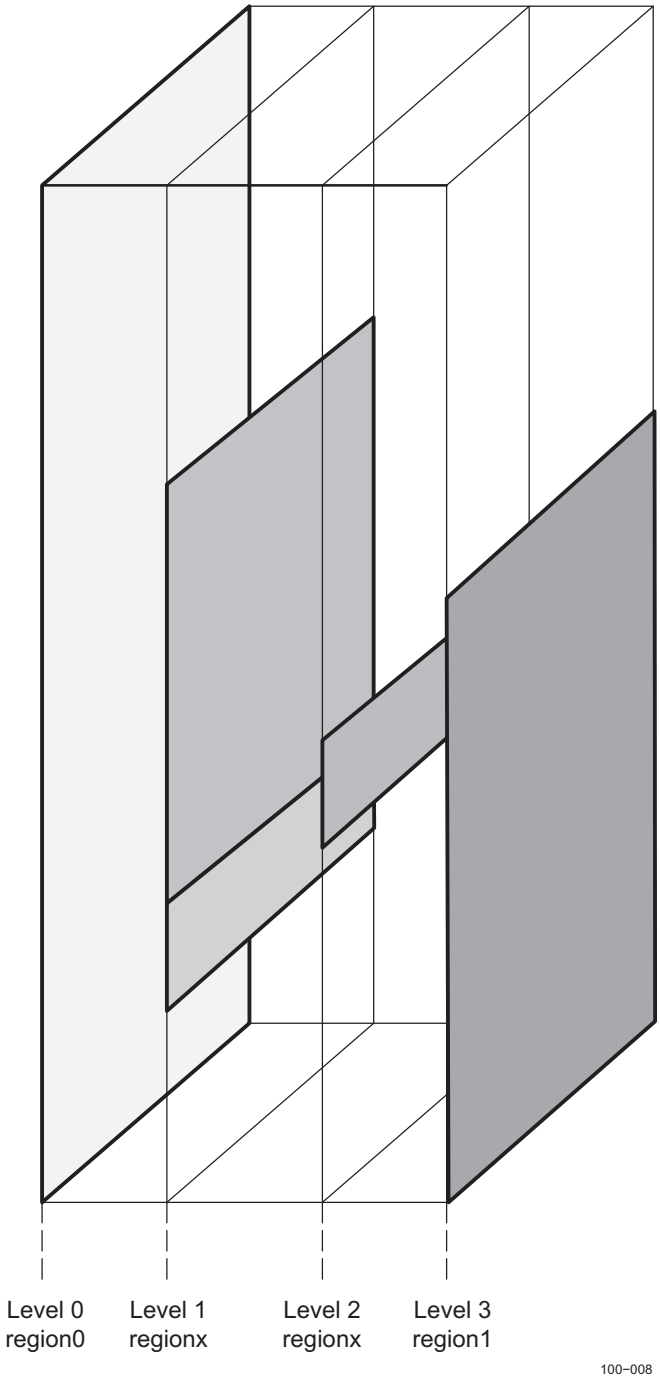
Hardware behavior in the case of overlapping protection regions is undefined. A region with higher priority must be used to mask a region that is being reprogrammed, and any security holes must be avoided during this reconfiguration.

To change the protection settings of a region, follow this procedure:

1. Ensure that a free region is available to be used as a high-priority region.
2. Program this region as a high-priority region so its parameters match the region to be changed. The final programming must be the L3_PM_ADDR_MATCH_k register, which includes the priority attribute and the size parameter to enable the region.
3. Disable the region to be configured or reconfigure it by setting the SIZE bit field to 0.
4. Set up all region control registers with the new configuration. The final programming must be the L3_PM_ADDR_MATCH_k register, which includes the size parameter to enable the region.
5. Disable the high-priority region by setting its size to 0.

This procedure must be used each time there is an overlap between the originally defined region and the newly defined region. The use of a high-priority region is required to keep security active during programming.

Figure 5-6. L3 Region Overlay and Priority Level Overview



5.4.3.3 Read and Write Permission

Read permission and write permission are configured using two registers:

- L3_PM_READ_PERMISSION_i
- L3_PM_WRITE_PERMISSION_i

The L3_PM_READ_PERMISSION_i and L3_PM_WRITE_PERMISSION_i registers allow the setting of read and write permission to one or more initiators. To grant read or write access, set the bit associated with the initiator to 1.

5.4.3.4 REQ_INFO_PERMISSION Configuration

The firewall comparison mechanism enables access to a protected target only when a correct combination of four MReqInfo in-band parameters is transmitted.

MReqInfo is a combination of a fixed 4-bit pattern that corresponds to a combination of the parameters MReqSecure, MRreqDebug, MrReqType, and MReqSupervisor.

NOTE: In GP device MReqSecure = 1 (Secure value), is not supported.

Different valid MReqInfo combinations can be defined for each L3 firewall region based on the L3_PM_REQ_INFO_PERMISSION_i value, which is programmed by software.

For each region, L3_PM_REQ_INFO_PERMISSION_i lists the possible MReqInfo combinations. Setting a Reqbit in this register determines the type of access allowed to the initiator.

Table 5-21 lists the MReqInfo combinations available and the Reqbit associated with it.

Table 5-21. MReqInfo Parameter Combinations

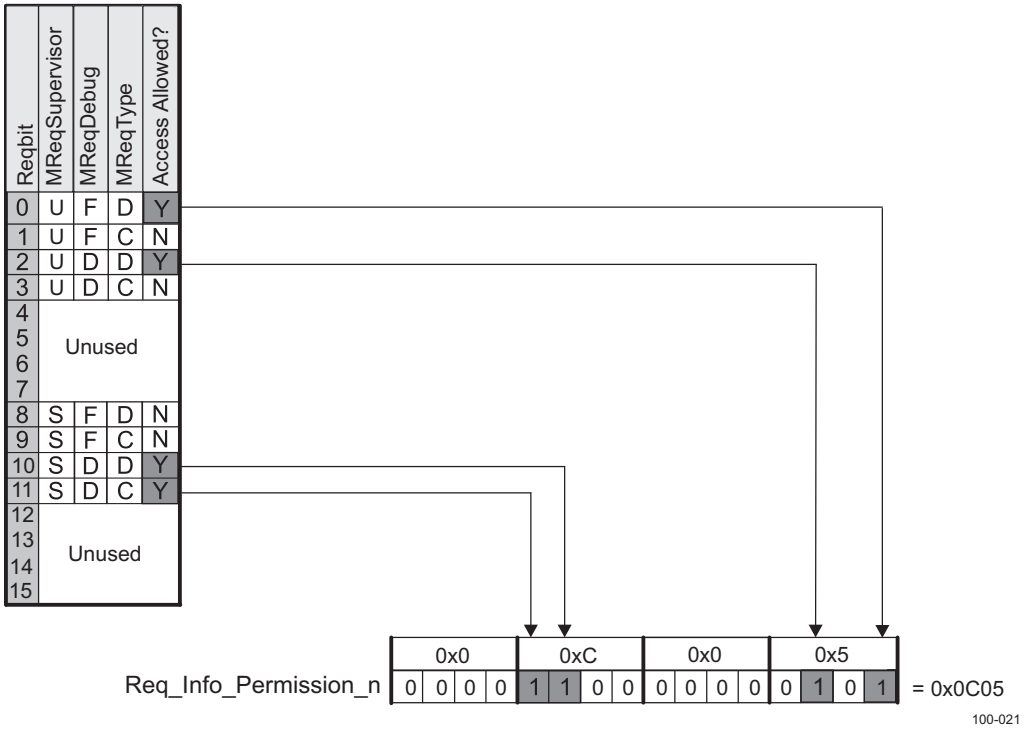
Reqbit	MReqInfo			
	MReqSupervisor	MReqSecure[1]	MReqDebug	MReqType
0	User	Public	Functional	Data
1	User	Public	Functional	Code
2	User	Public	Debug	Data
3	User	Public	Debug	Code
4	User	Secure[1]	Functional	Data
5	User	Secure[1]	Functional	Code
6	User	Secure[1]	Debug	Data
7	User	Secure[1]	Debug	Code
8	Supervisor	Public	Functional	Data
9	Supervisor	Public	Functional	Code
10	Supervisor	Public	Debug	Data
11	Supervisor	Public	Debug	Code
12	Supervisor	Secure[1]	Functional	Data
13	Supervisor	Secure[1]	Functional	Code
14	Supervisor	Secure[1]	Debug	Data
15	Supervisor	Secure[1]	Debug	Code

(1) In GP device MReqSecure = 1 (Secure value), is not supported.

Figure 5-7 shows an example of the L3_PM_REQ_INFO_PERMISSION_i setting. In this example, L3_PM_REQ_INFO_PERMISSION_i is set to 0x0C14 (16'b0000_1100_0001_0100) for a specific region, which will grant access only if the request:

- User + Public + Debug + Data
- User + Secure + Functional + Data
- Supervisor + Public + Debug + Data
- Supervisor + Public + Debug + Code

Figure 5-7. Example of REQ_INFO_PERMISSION Register



As another example, to configure a target accessible only for data and in a public mode, Reqbit 0, 2, 8, and 10 must be set. Therefore, the L3_PM_REQ_INFO_PERMISSION_i register must be set to 0x5050.

5.4.3.5 L3 Firewall Registers Overview

Table 5-22 lists the L3 firewall permission-setting registers. L3_PM_ADDR_MATCH_k, which is shown in this table, is not an accessible register.

Table 5-22. L3 Firewall Permission-Setting Registers

Register Name	Register Field Name	Field Modifiability	Parameter Comments	Region Comments
Region 0				This region is the default region. The default setting can be changed only with correct access according to L3 RT register.
L3_PM_ADDR_MATCH_k (k=0)	ADDR_SPACE[2:0]	Hard coded	Corresponds to all the target memory space	
	SIZE[7:3]	Hard coded	Corresponds to all the target memory space	
	Reserved		Default region: Level 0	
	BASE_ADDR[63:10]	Hard coded	Target-dependent	
L3_PM_REQ_INFO_PERMISSION_i (i=0)	REQ_INFO[15:0]	Yes	Type of access permitted. See Table 5-21.	
L3_PM_READ_PERMISSION_i (i=0)	READ_PERMISSION[15:0]	Yes	Initiator read permission, depending on connections. See Table 5-14.	
L3_PM_WRITE_PERMISSION_i (i=0)	WRITE_PERMISSION[15:0]	Yes	Initiator write permission, depending on connections. See Table 5-14.	

Table 5-22. L3 Firewall Permission-Setting Registers (continued)

Register Name	Register Field Name	Field Modifiability	Parameter Comments	Region Comments
Region 1-7				The default settings can be changed only with correct access based on the L3 RT register.
L3_PM_ADDR_MATCH_k	ADDR_SPACE[2:0]	Yes		
	SIZE [7:3]	Yes	The regions are power-of-two in size and size-aligned with Base_Addr reference. When Size = 0x0, the firewall is deactivated.	
	LEVEL[9]	Yes	Protection region level 0x0: Level 1 0x1: Level 2 Region 1 is always Level 3.	
	BASE_ADDR[63:10]	Yes	Target-dependent	
L3_PM_REQ_INFO_PERMISSION_i	REQ_INFO[15:0]	Yes	Type of access permitted. See Table 5-21.	
L3_PM_READ_PERMISSION_i	READ_PERMISSION[15:0]	Yes	Initiators read permission ,depending on connections. See Table 5-14.	
L3_PM_WRITE_PERMISSION_i	WRITE_PERMISSION[15:0]	Yes	Initiators write permission, depending on connections. See Table 5-14.	

5.4.3.6 L3 Firewall Error-Logging Registers

Table 5-23 lists the L3 firewall error-logging registers.

Table 5-23. L3 Firewall Error Logging Registers

Register Name	Register Field Name	Field Modifiability	Parameter Comments
L3_PM_ERROR_LOG	CMD[2:0]	Read only	Log the OCP command of the request that caused a protection violation. See Table 5-1.
	REGION[6:4]	Read only	Log the region number targeted by the request that caused the protection violation.
	INITIATOR_ID[15:8]	Read only	Log the InitiatorID request that caused the protection violation. See Table 5-18
	REQ_INFO[20:16]	Read only	Log the MReqInfo bits of the request that caused the protection violation. See Table 5-21.
	CODE[27:24]	Read/write	Log the error that occurred. See Table 5-25.
	MULT[31]	Read/write	If a second error is detected before the first is cleared, the MULT bit is set. Once set by hardware, the CODE and MULT bits can be cleared only by software or a full hardware reset. Software clears the CODE and MULT bits by writing a non-zero value to the CODE field and writing 1 to the MULT bit.

NOTE: This error log can be cleared in public mode by a read access. For more details, see Section 5.5.3.2, *Acknowledging Errors*.

5.4.3.7 L3 Firewall and System Control Module

When a security violation occurs, an interrupt is sent to the MPU interrupt controller (if enabled). An in-band error is sent back, and an out-band error is logged in the CONTROL.CONTROL_SEC_ERR_STATUS register. Two logging registers are used, depending on the functional mode:

- In application mode:
  - CONTROL.CONTROL_SEC_ERR_STATUS [00]: OCM-ROM security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [01]: OCM-RAM security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [02]: GPMC security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [04]: SMS security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [07]: L4-Core security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [12]: L3 RT security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [16]: L4-Per security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS [17]: L4-Emu security violation
- In debug mode:
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG [00]: OCM-ROM security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG [01]: OCM-RAM security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG [02]: GPMC security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG [03]: SMS security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG [12]: L3 RT security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG[16]: L4-Per security violation
  - CONTROL.CONTROL_SEC_ERR_STATUS_DEBUG[17]: L4-Emu security violation

When a violation occurs, these bits are cleared when the L3 or L4 firewall embedded error log registers are cleared.

The L3 interconnect allows the inputting of the reset value of some of its internal registers from the system control module. This feature is used to configure the L3 firewalls in a secure state at start-up when required for the different targets. For more information on these registers, see the *System Control Module* chapter.

CAUTION

The exported reset value registers in the system control module can be accessed only when the MPU is in secure mode. They cannot be accessed on general-purpose devices. Outside secure mode, a read to this register returns 0.

5.4.4 Error Handling

5.4.4.1 Error Detection and Logging

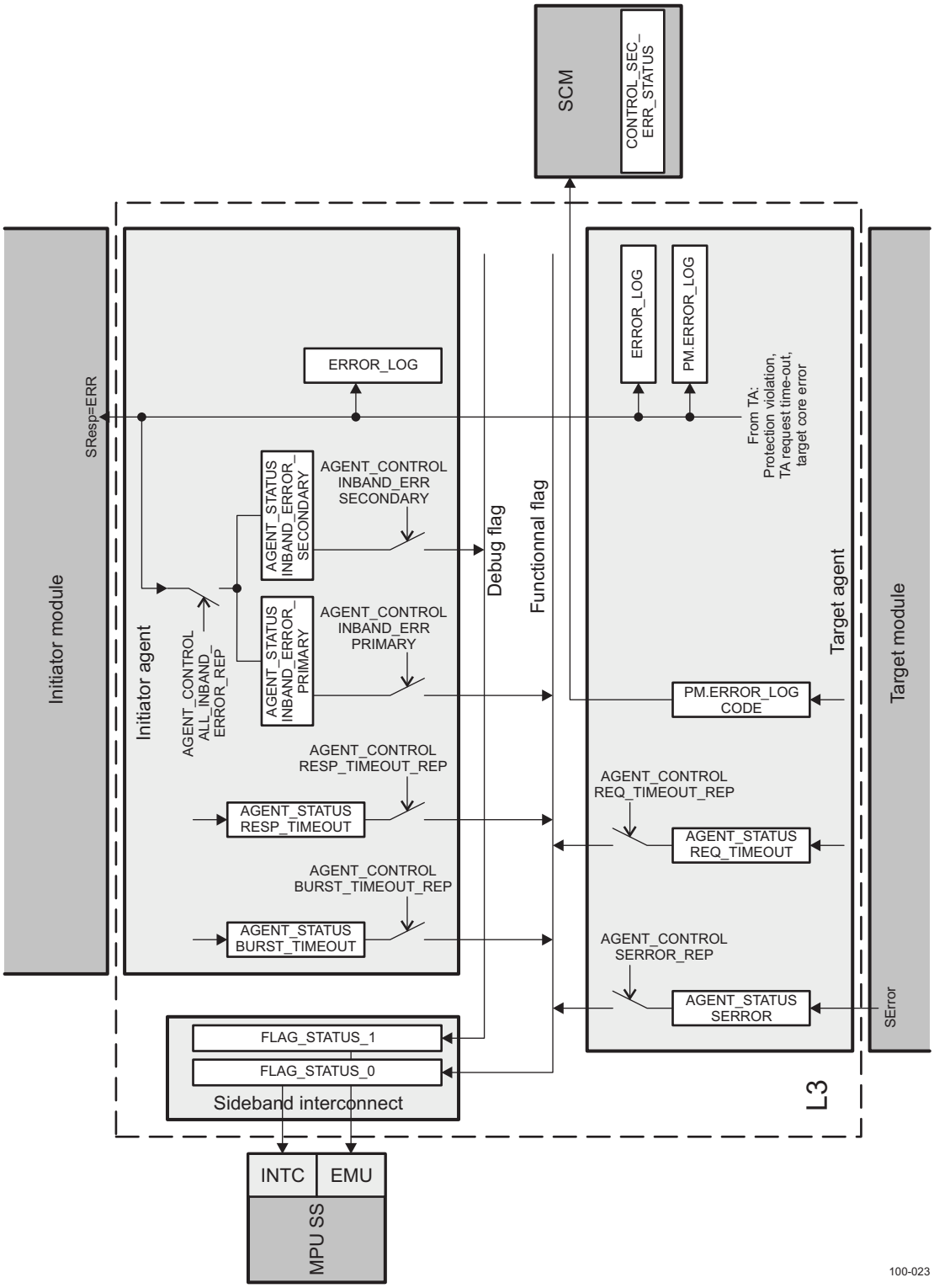
The L3 interconnect provides mechanisms for the detection, logging, and distribution of module-detected and internal interconnect errors. Hardware support is provided to assist in logging errors and cleaning up the state to allow error recovery software to run.

Two types of errors are identified by the L3 interconnect:

- Errors detected by modules and passed along by the interconnect:
  - SResp error: Using the SResp in-band qualifier from the target, the initiator is informed that its request was unsuccessful (see Table 5-3). This error is nonspecific and further analysis is required to find its cause.
  - SError: This out-of-band qualifier reports that the target is denied service due to an internal cause. No further analysis can be done in the L3 itself.
- Errors detected by the L3 interconnect:
  - Unsupported command: This error reports that the initiator sent a command that cannot be processed, because the target cannot accept it and no conversion to another command is possible. This error is detected only once per burst.
  - Address hole: This error reports an unknown address for a request. The address map is local to each IA; therefore, an address hole error is reported each time an initiator requests an access to a target it is not logically connected to, even if this address exists in the global L3 address map. This error is detected only once per burst.
  - Protection violation: This error indicates a request was rejected by a firewall.
  - Requests time-out: This error reports that the module did not end or respond to the request presented by the TA in the correct time interval. If the time interval for any request exceeds the time-out period, a time-out occurs. The target stops servicing requests.
  - Response time-out: This error reports that the module did not end the response presented by the IA in the correct time interval. If the time interval for any response exceeds the time-out period, a time-out occurs. The initiator stops servicing responses.
  - Burst open time-out: This error reports that a burst or ReadEx/Write pair is open and no command is presented to the module to end it. If the time interval for any command exceeds the time-out period, a time-out occurs. The initiator does not complete a burst or ReadEx/Write pair.

Figure 5-8 shows a global view of the register link to the error reporting structure in an initiator and the target agents.

Figure 5-8. L3 Error Reporting Structure



Most errors are reported to the IA that originated the request, except for the following:

- Initiator time-outs are reported only out-of-band. Any time-out in the IA results in flagging it as unavailable. A software reset of the agent is required to accept any new requests from the attached core.

- Posted write request, because the IA immediately generates a valid response to the initiator core. This error is only in out-of-band.

Table 5-24 lists where the errors are detected and logged.

Table 5-24. Error Types

Error	Detection	Logging	In-band Report	Out-of-band Report
SResp error	None needed	At IA	Yes[1]	Yes
SError assertion from target	None needed	At TA	No	Yes
Unsupported command	At IA	At IA	Yes	Yes
Address hole	At IA	At IA	Yes	Yes
Protection violation	At TA	At protection mechanism agent	Yes[1]	Yes
Request time-out	At TA	At TA	Yes [1]	Yes
Response time out	At IA	At IA	No	Yes
Burst time-out	At IA	At IA	No	Yes

(1) In case of a posted write, errors cannot be reported in-band.

The time-out errors (request/response and burst) are persistent and require the software to reset both the module and the agent. No request can be processed in the agent until the reset is performed.

Other errors affect only the current request. Subsequent requests are treated normally. Errors are logged, however, and the information about the error used for debugging is kept until the software acknowledges the error.

Table 5-25 lists the information logged for each error code. Information logged in the two error-logging registers (L3_IA_ERROR_LOG, L3_IA_ERROR_LOG_ADDR, L3_TA_ERROR_LOG and L3_TA_ERROR_LOG_ADDR for each IA and TA) depends on the error itself. The CODE field ERROR_LOG[27:24] identifies the type of error occurring for the IA and TA.

Table 5-25. CODE Field Definition

		Type of Agent			Information Logged				
CODE[3:0]	Error Type	IA	TA	PM	REQ_INFO	Secondary	InitiatorID	CMD	Address
0	No error	x	x						
1	Unsupported command	x			x	x	x	x	x
2	Address hole	x			x	x	x	x	x
3	Protection violation			x	x		x	x	
4	In-band error	x				x	x		
5	Not used								
6	Not used								
7	Request time-out not accepted		x		x		x	x	x
8	Request time-out, no response		x				x		
9-15	Not used								

5.4.4.2 Time-Out

This section gives information about all modules and features in the high-tier device. See the *Device Family* chapter to check availability of modules and features. To flag interconnect response being blocked, the time-out of target agents attached to unavailable modules can be enabled with the lowest setting.

A time-out mechanism can be enabled in the target agent and initiator agent register. When the mechanism is enabled for a target agent or initiator agent and commands are not accepted or responses are not returned within the expected delay, the L3 interconnect generates an error event.

The error is logged in the target agent REQ_TIMEOUT bit L3_TA_AGENT_STATUS[8] for a target agent time-out, the BURST_TIMEOUT bit L3_IA_AGENT_STATUS[16] bit for an initiator burst time-out, and the RESP_TIMEOUT bit L3_IA_AGENT_STATUS[8] for an initiator response time-out. The affected agent enters an error state that causes it to send error responses to any new request. To recover from this state, the target agent must be reset by system software.

The time-out is counted starting from the moment a command is presented to the target, whatever the target response to this command is. The L3 interconnect implements a centralized time-base circuit that broadcasts a set of four periodic pulse signals to all connected target agents. These four signals are referred to as 1x time-base, 4x time-base, 16x time-base, and 64x time-base.

The time-base circuit offers four possible sets of four time-base signals selected by programming the TIMEOUT_BASE field L3_RT_NETWORK_CONTROL[10:8]. Table 5-26 lists all of the values in the number of L3 clock cycles.

Each target agent can be programmed to refer to one of the four time-base signals. A time-out condition is detected when either the command acceptance or the response is not received after a delay of between one and three time-base periods. After the time-out is detected and logged, the behavior of the attached module is ignored. A new request to the module arriving at the timed-out target agent receives an error response. If the request is addressed to the agent internal registers, it is processed normally.

To recover from a time-out error, the software is assumed to reset first the faulty module using its internal soft-reset bit, and then the agent using software reset.

Table 5-26. L3 Timeout Register Target and Agent Programming

REQ_TIMEOUT[2:0], BURST_TIMEOUT[2:0], and RESP_TIMEOUT[2:0]					
TIMEOUT_BASE[2:0]	**0**	**1**	**2**	**3**	**4**
0	All L3 time-out features are disabled.				
1	Locally disabled	64	256	1024	4096
2		256	1024	4096	16384
3		1024	4096	16384	65536
4		4096	16384	65536	262144

5.4.4.3 Error Steering

The error reporting structure consist of errors logged individually in the initiator or TAs. Some errors can be enabled and reported out-of-band by setting the following bits to 1:

- L3_IA_AGENT_CONTROL.BURST_TIMEOUT_REP bit for burst time-out
- L3_IA_AGENT_CONTROL.RESP_TIMEOUT_REP bit for response time-out
- L3_TA_AGENT_CONTROL.REQ_TIMEOUT_REP bit for request time-out
- L3_TA_AGENT_CONTROL.SERROR_REP bit for request time-out

  Setting the L3_IA_AGENT_CONTROL.ALL_INBAND_ERROR_REP bit causes all in-band errors returned to the IA to be reported out-of-band.

  IAs connected to processors capable of generating the debug-flagged requests (the MPU subsystems) use error steering. Any error linked to an application (nondebug) request is qualified as primary; any error linked to a debug request is qualified as secondary. Setting the IA.INBAND_ERROR_PRIMARY_REP or INBAND_ERROR_SECONDARY_REP bit to 1 allows the reporting of in-band to out-of-band primary and secondary errors.

  The level of the current error is indicated in SECONDARY bit L3_IA_ERROR_LOG[30]. If an error occurs while another error is pending, the following occurs:

  - If the pending error is primary, the new error is discarded at the IA level. MULTI bit L3_IA_ERROR_LOG[31] is set in the initiator error log register to indicate that another error has been detected; no further information can be stored.

- If both the pending error and the new error are secondary, the latest error is discarded and MULTI bit L3_IA_ERROR_LOG[31] is set.
- If the pending error is secondary and the incoming error is primary, MULTI bit L3_IA_ERROR_LOG[31] is set and all useful information about the new error (MCmd, MAddr, MReqInfo, etc.) is stored. All information relative to the secondary error is discarded.

For protection violation, errors are detected at the TA and steered to the control module (see Section 5.4.3.7, *L3 Firewall and System Control Module*).

5.4.4.4 Global Error Reporting

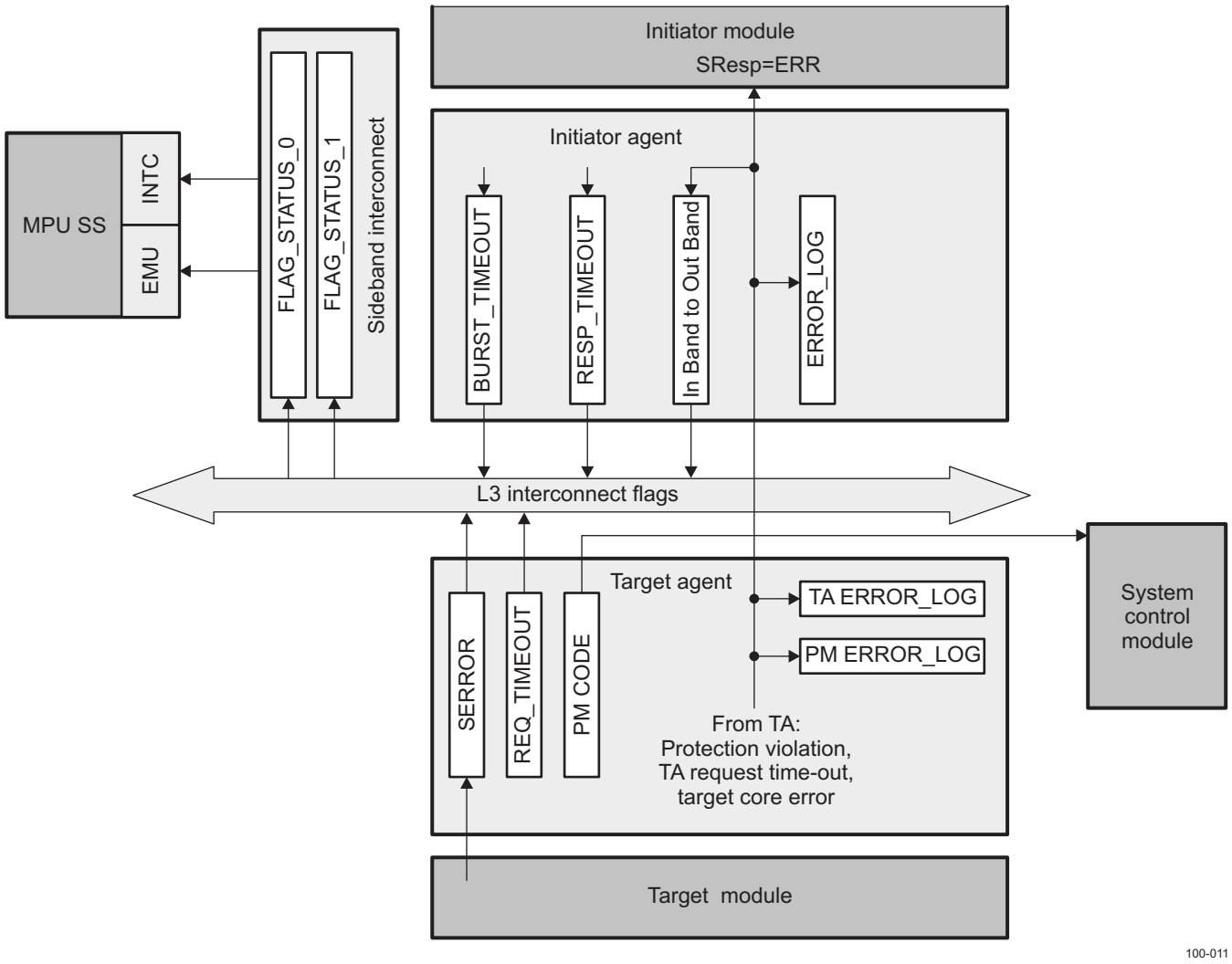
An out-of-band error refers to any flag output from the L3. It is not a formal error in the sense that it is not processed directly by the interconnect module; however, it is interpreted as such by one or several processors, typically by mapping it to an interrupt controller. All L3 out-of-band errors are ORed together and the result is transmitted to one or several interrupt controllers. All out-of-band errors are active high and must be acknowledged through a register access to be de-asserted.

Out-of-band errors are reported through error or flag signals, asynchronously to any other data flow, but synchronously to the clock.

All errors are reported out-of-band to the MPU subsystem simultaneously. The processors must check whether an error is relevant to the subsystems. These errors are routed not only to the MPU subsystem interrupt controllers, but also to their emulation logic (see Figure 5-9). Two composite flags help in asserting the error origin and criticality:

- The L3 application error flag reports application or nonattributable (not related to a request such as a time-out, SError) errors.
- The L3 debug error flag reports debug errors.

Figure 5-9. Global Error Routing



In addition to the SResp qualifier error reporting, some external targets use an SError signal to indicate that an internal error has occurred. Some resetting action may be required before any new request can be accepted, depending on the module and on the error itself. These signals are routed as flags internally to the L3. Table 5-27 lists the possible errors reported through an SError, propagated externally to the L3 interconnect, and aggregated to the L3 application error flag.

Table 5-27. L3 External Input Flags

Target	Flag	Description	Software Visible?
GPMC	SError	General-purpose error occurs in GPMC module	Yes, in FLAG_STATUS register and in L3 TA SERROR bit AGENT_STATUS[24]
L4-Core	SError	General-purpose error occurs on L4-Core	Yes, in FLAG_STATUS register and in L3 TA SERROR bit AGENT_STATUS[24]
L4-Emu	SError	General-purpose error occurs on L4-Emu	Yes, in FLAG_STATUS register and in L3 TA SERROR bit AGENT_STATUS[24]
L4-Per	SError	General-purpose error occurs on L4-Per	Yes, in FLAG_STATUS register and in L3 TA SERROR bit AGENT_STATUS[24]

NOTE: The user must ensure that the corresponding IRQ lines are set correctly in the interrupt controllers of the MPU subsystem.

Figure 5-10 shows the routing of errors to the application and debug error composite flags.

Figure 5-10. L3 Error Routing

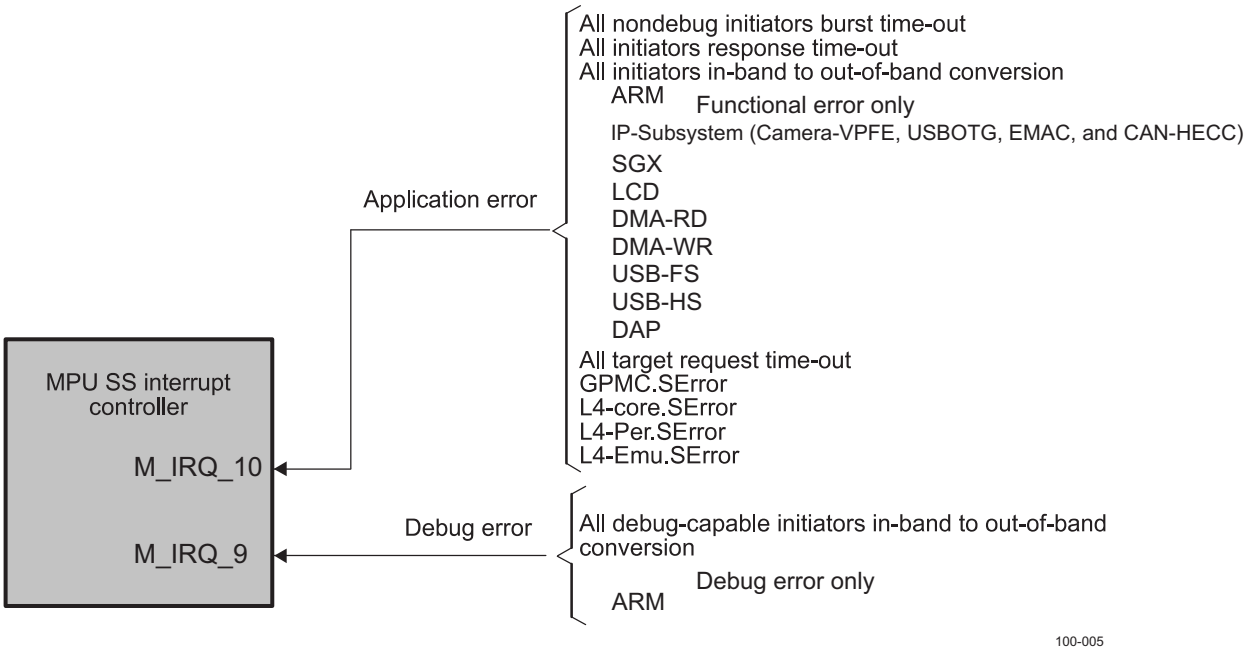


Table 5-28 lists the bit and source for the STATUS bit field L3_SI_FLAG_STATUS_0[63:0] for an application error, and Table 5-29 lists the bit and source for the STATUS bit field L3_SI_FLAG_STATUS_1[63:0] for an application error for a debug error.

Protection errors are reported in-band to the IA when possible. This may not be possible for posted writes, however. These errors are also reported asynchronously to the module.

Additionally, all protection errors (even posted writes) are seen by the in-band-to-out-of-band conversion logic in the IAs. Therefore, all protection errors feed into the L3 application error and L3 debug error composite flags, and hence back to the MPU subsystem.

The SMS, L4-Core, L-4 Per, and L4-Emu interconnects have internal firewalls that use the same reporting scheme. These errors are not routed directly to the L3 interconnect but are reported in-band to the initiator, which routes them to the composite flags. The SMS, L4-Core, L4-Per, and L4-Emu interconnects always provide an error on the in-band SResp qualifier for protection violations.

Table 5-28. L3_SI_FLAG_STATUS_0 for Application Error

Flag Bit Number	Source		Flag Bit Number	Source	
	Agent	Error		Agent	Error
0	MPU IA	Burst time-out	32	Reserved	
1	MPU IA	Response time-out	33		Burst time-out
2	MPU IA	Functional Inband error	34		Response time-out
3	Reserved		35		Functional Inband error
4			36	Reserved	
5			37		
6		Burst time-out	38		

Table 5-28. L3_SI_FLAG_STATUS_0 for Application Error (continued)

Flag Bit Number	Source		Flag Bit Number	Source	
	Agent	Error		Agent	Error
7		Response time-out	39	Reserved	
8		Functional Inband error	40		
9	SGX TA	Burst time-out	41		
10	SGX TA	Functional Inband error	42		
11	Reserved		43		
12		Burst time-out	44		
13		Response time-out	45		
14		Functional Inband error	46		
15	Display SS IA	Burst time-out	47		
16	Display SS IA	Functional Inband error	48	SMS TA	Request time-out
17	Reserved		49	GPMC TA	Request time-out
18	sDMA Rd IA	Burst time-out	50	OCM RAM TA	Request time-out
19	sDMA Rd IA	Functional Inband error	51	OCM ROM TA	Request time-out
20	Reserved		52	L4-Core TA	Request time-out
21	sDMA Wr IA	Burst time-out	53	L4-Per TA	Request time-out
22	sDMA Wr IA	Functional Inband error	54	IP-Subsystem (Camera-VPFE, USBOTG, EMAC and CAN-HECC)	Request time-out
23	Reserved		55	SGX TA	Request time-out
24	IP-Subsystem (Camera-VPFE, USBOTG, and EMAC)	Burst time-out	56	L4-Emu TA	Request time-out
25	IP-Subsystem (Camera-VPFE, USBOTG, and EMAC)	Response time-out	57	GPMC TA	SError assertion
26	IP-Subsystem (Camera-VPFE, USBOTG, and EMAC)	Functional Inband error	58	L4-Core TA	SError assertion
27	HS USB Host IA	Burst time-out	59	L4-Per TA	SError assertion
28	HS USB Host IA	Functional Inband error	60	L4-Emu	SError assertion
29	Reserved		61		SError assertion
30			62	Reserved	
31			63		

Table 5-29. L3_SI_FLAG_STATUS_1 for Debug Error

Flag Bit Number	Source		Flag Bit Number	Source	
	Agent	Error		Agent	Error
0	MPU DATA IA	Debug error	32		
1	Reserved		33		
2			34		
3			35		

Table 5-29. L3_SI_FLAG_STATUS_1 for Debug Error (continued)

Flag Bit Number	Source		Flag Bit Number	Source	
	Agent	Error		Agent	Error
4	Reserved		36	Reserved	
5			37		
6			38		
7			39		
8			40		
9			41		
10			42		
11			43		
12			44		
13			45		
14			46		
15			47		
16			48		
17			49		
18			50		
19			51		
20			52		
21			53		
22			54		
23			55		
24			56		
25			57		
26			58		
27			59		
28			60		
29			61		
30			62		
31			63		

5.5 L3 Interconnect Basic Programming Model

5.5.1 General Recommendation

The L3 interconnect registers must be read or written with little-endian attributes; otherwise, the result is undefined.

CAUTION

Overlapping between protection regions with the same priority level leads to unpredictable behavior and must be avoided.

5.5.2 Initialization

At the release of power on reset, the L3 firewall default configuration enables all accesses to target modules, except for a section of the OCM ROM for which default Region 0 is configured to enable access for the MPU with the secure attribute only (this section corresponds to the secure boot ROM in the device memory mapping).

Therefore, the settings of Region 1 in the OCM ROM firewall allow access to 32K bytes of the OCM (this accessible 32K bytes corresponds to the public boot ROM in memory space mapping). For more details, see the *Memory Mapping* chapter.

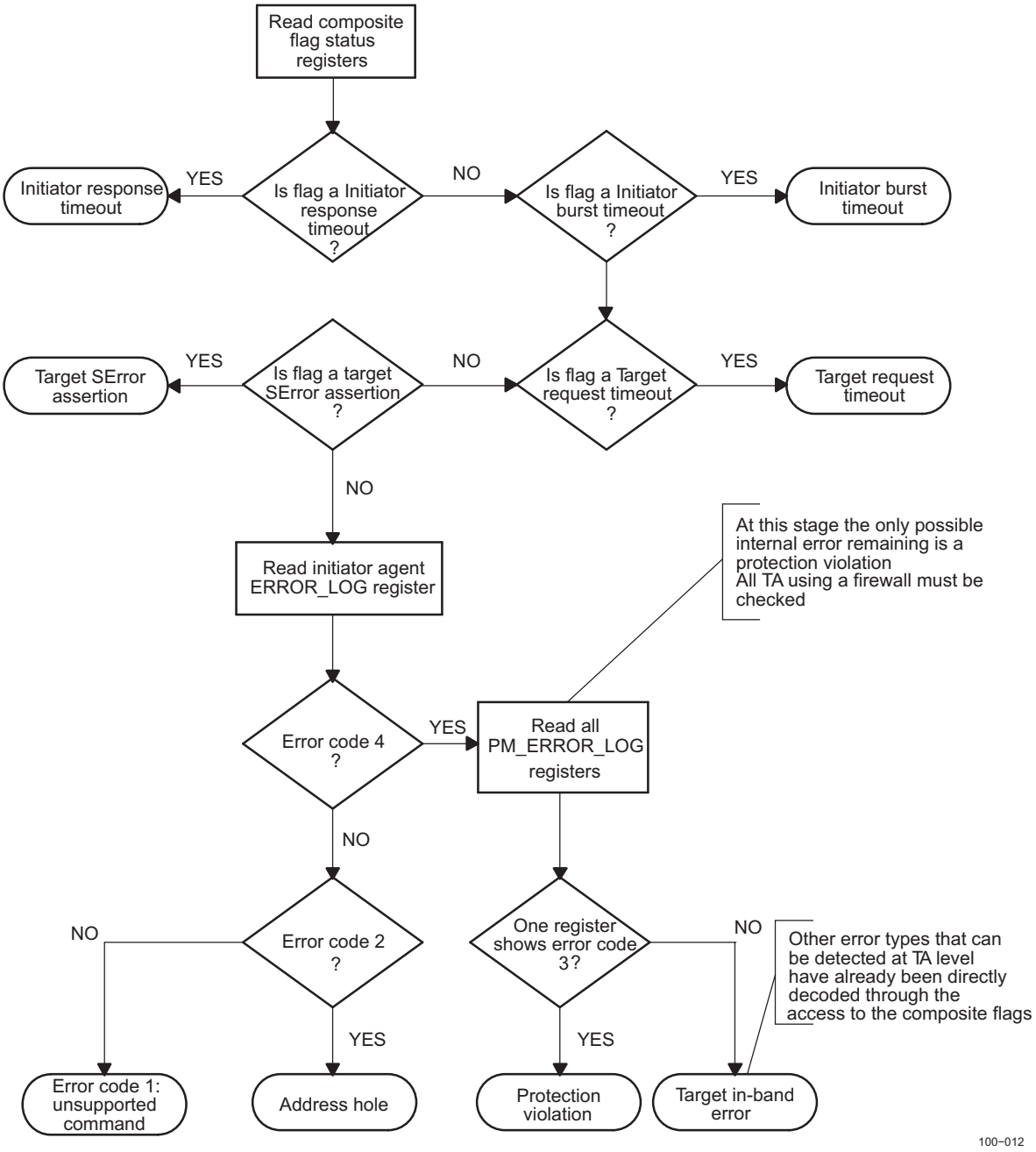
Generally, software must configure the firewall properly to avoid poor use of the hardware resources.

L3 time-out capabilities are also disabled at reset.

5.5.3 Error Analysis

The information required to analyze an error source is logged in several registers (see Table 5-25). The number of registers to access depends on the error source. When investigating the origin of an error, software reads a set of error log registers. At each stage, the register either states the current error or points to the next agent in which the error is logged. Figure 5-11 shows the software sequence required in most cases.

Figure 5-11. Typical Error Analysis Sequence



Errors that do not result from an in-band to out-of-band conversion can be extracted immediately from the application or debug error flag by reading the STATUS field L3_SI_FLAG_STATUS_0[63:0] and L3_SI_FLAG_STATUS_1[63:0] register; therefore, they do not require the whole analysis sequence shown in Table 5-25.

When analysis leads to a TA error, software must read the initiator agent ADDR field L3_IA_ERROR_LOG_ADDR[39:0] to extract the TA address causing the error.

5.5.3.1 Time-out Handling

This section gives information about all modules and features in the high-tier device. See the *Device Family* chapter to check availability of modules and features. To flag interconnect response being blocked, the time-out of target agents attached to unavailable modules can be enabled with the lowest setting.

Example 1

In this example, the MPU interrupt handler detects an error from the L3 interconnect. A read access from the L3_SI_FLAG_STATUS_0 register reports a value of 0x40000. As described in Table 5-28, the error detected is a burst time-out from the sDMA read port.

As with any time-out error, the affected module is now considered to be out-of-service. If necessary, the error can be cleared by sending a soft reset command to the agent (set CORE_RESET bit L3_IA_AGENT_CONTROL[0] to 1, and then to 0). During this time the initiator is off-line. Although the rest of the system still behaves normally, a time-out error is usually severe enough to require a complete reset of the chip.

Before resetting the agent or the system, the error log registers can learn the status of the interconnect and determine the type of failure. Reading the IA_SDMA_RD.L3_IA_AGENT_STATUS register shows whether the time-out was detected during a burst or during a read/write sequence.

5.5.3.2 Acknowledging Errors

Time-out errors can never be acknowledged. To return to a normal operation after an error, the faulty agent must be reset . An agent can be reset by asserting CORE_RESET bit L3_IA_AGENT_CONTROL[0] or L3_TA_AGENT_CONTROL[0], or by resetting the L3 interconnect through the PRCM.

Functional errors, including an in-band signal reporting a protection error, must be inactived through software. Setting the INBAND_ERROR_PRIMARY and INBAND_ERROR_SECONDARY bits L3_IA_AGENT_STATUS[28-29] in an IA, or setting the SERROR bit L3_IA_AGENT_STATUS[24] in a TA clears the reported error. Table 5-30 lists the bit to clear and the associated type of error.

The L3_IA_ERROR_LOG or L3_TA_ERROR_LOG register must also be cleared by writing a nonzero value simultaneously to the CODE bit field and the values currently stored in the MULTI and SECONDARY fields.

Table 5-30. Error Clearing

Agent Type	Error	Register Field
Initiator	In-band primary (application) error	INBAND_ERROR_PRIMARY
	In-band secondary (debug) error	INBAND_ERROR_SECONDARY
Target	Target asserts SError	SERROR

The procedure to clear protection errors depends on the system security configuration:

- Write a nonzero value simultaneously into CODE bit field L3_PM_ERROR_LOG[27:24] and the value currently stored in the MULTI bit L3_PM_ERROR_LOG[31] of the corresponding PM register block.
- Alternately, read either L3_PM_ERROR_CLEAR_SINGLE or L3_PM_ERROR_CLEAR_MULTI, depending on the current value of the MULTI field in the L3_PM_ERROR_LOG register. This solution, which allows the clearing of protection errors without having a write access on other protection registers, preserves security.

L3_SI_FLAG_STATUS_0 or L3_SI_FLAG_STATUS_1 must be checked at the end of any error acknowledging sequence to confirm that the acknowledgement was successful and that no other error is pending.

5.6 L3 Interconnect Registers

Table 5-31 lists the base address and address space for all L3 register blocks.

Table 5-31. Instance Summary

Module Name	Base Address	Size
RT	0x6800 0000	1K byte
SI	0x6800 0400	1K byte
IA_MPUSS	0x6800 1400	1K byte
Reserved	0x6800 1800	1K byte
IA_SGX	0x6800 1C00	1K byte
TA_SMS	0x6800 2000	1K byte
TA_GPMC	0x6800 2400	1K byte
TA_OCM_RAM	0x6800 2800	1K byte
TA_OCM_ROM	0x6800 2C00	1K byte
Reserved	0x6800 3000	1K byte
Reserved	0x6800 3400	1K byte
IA_USB_HS_Host	0x6800 4000	1K byte
IA_IPSS (Camera-VPFE, USBOTG, and EMAC)	0x6800 4400	1K byte
IA_sDMA_RD	0x6800 4C00	1K byte
IA_sDMA_WR	0x6800 5000	1K byte
IA_DSS	0x6800 5400	1K byte
IA_DAP	0x6800 5C00	1K byte
TA_IPSS (Camera-VPFE, USBOTG, EMAC, and CAN-HECC	0x6800 6000	
TA_SGX	0x6800 6400	1K byte
TA_L4_CORE	0x6800 6800	1K byte
TA_L4_PER	0x6800 6C00	1K byte
TA_L4_EMU	0x6800 7000	1K byte
PM_RT	0x6801 0000	1K byte
PM_GPMC	0x6801 2400	1K byte
PM_OCM_RAM	0x6801 2800	1K byte
PM_OCM_ROM	0x6801 2C00	1K byte
PM_IPSS (Camera-VPFE, USBOTG, EMAC and CAN-HECC)	0x6801 4000	1K byte

5.6.1 L3 Initiator Agent (L3 IA) Register Mapping Summary

This section describes the IA register block. Each IA in L3 interconnect has its own IA register block.

The following are the IA registers:

- MPU subsystem port
- SGX subsystem port
- High-Speed USB Host
- IP-Subsystem (Camera-VPFE, USBOTG, and EMAC)
- System DMA read port
- System DMA write port
- Display subsystem port
- DAP port

Table 5-32 through Table 5-35 list the IA registers and their physical addresses, depending on the module instance.

Table 5-32. Initiator Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	IA_MPUSS Physical Address	SGX Physical Address	Section
L3_IA_AGENT_CONTROL	RW	64	0x020	0x6800 1420	0x6800 1C20	Section 5.6.2.1
L3_IA_AGENT_STATUS	RW	64	0x028	0x6800 1428	0x6800 1C28	Section 5.6.2.2
L3_IA_ERROR_LOG	RW	64	0x058	0x6800 1458	0x6800 1C58	Section 5.6.2.3
L3_IA_ERROR_LOG_ADDR	R	64	0x060	0x6800 1460	0x6800 1C60	Section 5.6.2.4

Table 5-33. Initiator Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	IA_USB_ HS_Host Physical Address	IA_IPSS Physical Address	Section
L3_IA_AGENT_CONTROL	RW	64	0x020	0x6800 4020	0x6800 4420	Section 5.6.2.1
L3_IA_AGENT_STATUS	RW	64	0x028	0x6800 4028	0x6800 4428	Section 5.6.2.2
L3_IA_ERROR_LOG	RW	64	0x058	0x6800 4058	0x6800 4458	Section 5.6.2.3
L3_IA_ERROR_LOG_ADDR	R	64	0x060	0x6800 4060	0x6800 4460	Section 5.6.2.4

Table 5-34. Initiator Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	IA_DMA_RD Physical Address	IA_DMA_WR Physical Address	Section
L3_IA_AGENT_CONTROL	RW	64	0x020	0x6800 4C20	0x6800 5020	Section 5.6.2.1
L3_IA_AGENT_STATUS	RW	64	0x028	0x6800 4C28	0x6800 5028	Section 5.6.2.2
L3_IA_ERROR_LOG	RW	64	0x058	0x6800 4C58	0x6800 5058	Section 5.6.2.3
L3_IA_ERROR_LOG_ADDR	R	64	0x060	0x6800 4C60	0x6800 5060	Section 5.6.2.4

Table 5-35. Initiator Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	IA_DSS Physical Address	IA_DAP Physical Address	Section
L3_IA_AGENT_CONTROL	RW	64	0x020	0x6800 5420	0x6800 5C20	Section 5.6.2.1
L3_IA_AGENT_STATUS	RW	64	0x028	0x6800 5428	0x6800 5C28	Section 5.6.2.2
L3_IA_ERROR_LOG	RW	64	0x058	0x6800 5458	0x6800 5C58	Section 5.6.2.3

Table 5-35. Initiator Agent Common Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	IA_DSS Physical Address	IA_DAP Physical Address	Section
L3_IA_ERROR_LOG_ADDR	R	64	0x060	0x6800 5460	0x6800 5C60	Section 5.6.2.4

5.6.2 L3 Initiator Agent (L3 IA) Register Descriptions

5.6.2.1 L3_IA_AGENT_CONTROL

Table 5-36. L3_IA_AGENT_CONTROL

Address Offset	0x020
Physical Address	Please refer from Table 5-32 to Table 5-35
Description	Agent control register of IA block
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		INBAND_ERROR_SECONDARY_REP	INBAND_ERROR_PRIMARY_REP	ALL_INBAND_ERROR_REP	BURST_TIMEOUT_REP	RESP_TIMEOUT_REP	Reserved							BURST_TIMEOUT		Reserved						RESP_TIMEOUT		Reserved		REJECT	Reserved			CORE_RESET	

Bits	Field Name	Description	Type	Reset
63:30	Reserved	Reserved	R	0x000000000
29	INBAND_ERROR_ SECONDARY_REP	Reporting of in-band errors indicating debug error. 0x0:No special reporting 0x1:Report error	RW	1
	Reserved for instances 3 to 12	Reserved	R	0x0
28	INBAND_ERROR_PRIMARY_REP	Reporting of in-band errors indicating application error. 0x0:No special reporting 0x1:Report error	RW	1
27	ALL_INBAND_ERROR_REP	Reporting of all in-band errors 0x0:Only report errors that cannot be reported in-band 0x1:Report all in-band errors	RW	1
26	BURST_TIMEOUT_REP	Open burst and ReadEx/Write timeout reporting 0x0:No special reporting 0x1:Report out of band	RW	1

Bits	Field Name	Description	Type	Reset
25	RESP_TIMEOUT_REP	Response timeout reporting 0x0:No special reporting 0x1:Report out-of-band	RW	1
	Reserved for instances 3, 6 to 10 and 12	Reserved	R	0x00
24:19	Reserved	Reserved	R	0x00
18:16	BURST_TIMEOUT	Response Timeout Bound: 0x0: No timeout 0x1: 1x base cycles 0x2: 4x base cycles 0x3: 16x base cycles 0x4: 64x base cycles	RW	0x00
15:11	Reserved	Reserved	R	0x00
10:8	RESP_TIMEOUT	Response Timeout Bound: 0x0: No timeout 0x1: 1x base cycles 0x2: 4x base cycles 0x3: 16x base cycles 0x4: 64x base cycles	RW	0x0
	Reserved for instances 3, 6 to 10 and 12	Reserved	R	0x0
7:5	Reserved	Reserved	R	0x0
4	REJECT	Request rejection control 0x0:Normal operation 0x1:Block requests from the initiator.	RW	0
3:1	Reserved	Reserved	R	0x0
0	CORE_RESET	Reset control for agent and reset control on core 0x0:Core reset control inactive 0x1:Core reset control active	RW	0

5.6.2.2 L3_IA_AGENT_STATUS

Table 5-37. L3_IA_AGENT_STATUS

Address Offset	0x028
Physical Address	Please refer from Table 5-32 to Table 5-35
Description	Agent Status Register
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		INBAND_ERROR_SECONDARY	INBAND_ERROR_PRIMARY	Reserved											BURST_TIMEOUT	TIMEBASE				Reserved			RESP_TIMEOUT	READEX	BURST	RESP_WAITING	REQ_ACTIVE	Reserved			CORE_RESET

Bits	Field Name	Description	Type	Reset
63:30	Reserved	Reserved	R	0x000000000
29	INBAND_ERROR_ SECONDARY	Error Status for in-band errors indicating a debug error. Read 0x0:No in-band error received Write 0x0:Ignored Read 0x1:In-band error received Write 0x1:Clear in-band error	RW	0
	Reserved for instances 3 to 12	Reserved	R	0x0
28	INBAND_ERROR_ PRIMARY	Error Status for in-band errors indicating application Error Read 0x0:No in-band error received Write 0x0:Ignored Read 0x1:In-band error received Write 0x1:Clear in-band error	RW	0
27:17	Reserved	Reserved	R	0x0
16	BURST_TIMEOUT	Status of open burst and	R	0
15:12	TIMEBASE	Observation of timebase signals for internal verification	R	0x0
11:9	Reserved	Reserved	R	0x0
8	RESP_TIMEOUT	Response timeout status	R	0
	Reserved for instances 3, 6 to 10 and 12	Reserved	R	0
7	READEX	Status of ReadEx/Write	R	0
6	BURST	Status of open burst	R	0
5	RESP_WAITING	Response Waiting	R	0
	Reserved for instance 3,5,7 and 8	Reserved	R	0
4	REQ_ACTIVE	Requests outstanding	R	0

Bits	Field Name	Description	Type	Reset
3:1	Reserved	Reserved	R	0x0
0	CORE_RESET	Reset input from core interface	R	0

5.6.2.3 L3_IA_ERROR_LOG

Table 5-38. L3_IA_ERROR_LOG

Address Offset	0x058
Physical Address	Please refer from Table 5-32 to Table 5-35
Description	Error log register of IA block
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																REQ_INFO															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MULTI	SECONDARY	Reserved		CODE				Reserved								INITID								Reserved					CMD		

Bits	Field Name	Description	Type	Reset
63:48	Reserved	Reserved	R	0x0000
47:32	REQ_INFO	MReqInfo bits of command that caused the error	R	0x0000
31	MULTI	Multiple Errors Write 0x0:Ignored Read 0x0:Multiple error not seen Write 0x1:Clear MULTI flag Read 0x1:Multiple error seen	RW	0
30	SECONDARY	Indicates whether error was primary or secondary Write 0x0:Ignored Read 0x0:Primary Error Write 0x1:Reset SECONDARY field Read 0x1:Secondary Error	RW	0
29:28	Reserved	Reserved	R	0x0
27:24	CODE	Error code	RW	0x0
23:16	Reserved	Reserved	R	0x00
15:8	INITID	Initiator ID from which the command was launched	R	0x00
7:3	Reserved	Reserved	R	0x00
2:0	CMD	Command that caused the error	R	0x0

5.6.2.4 L3_IA_ERROR_LOG_ADDR

Table 5-39. L3_IA_ERROR_LOG_ADDR

Address Offset	0x060
Physical Address	Please refer from Table 5-32 to Table 5-35
Description	Error log address register of IA block
Type	R

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

Bits	Field Name	Description	Type	Reset
63:32	Reserved	Reserved	R	0x000000
31:0	ADDR	Address of the command that caused the error	R	0x0000000000

5.6.3 L3 Target Agent (L3 TA) Register Mapping Summary

This section describes the TA register block. Each TA in L3 interconnect has its own register block.

The following are the TA registers:

- SDRAM memory scheduler (TA_SMS module)
- General-purpose memory controller (TA_GPMC module)
- On-chip memory RAM (TA_OCM_RAM module)
- On-chip memory ROM (TA_OCM_ROM module)
- IP-Subsystem (Camera-VPFE, USBOTG, EMAC and CAN-HECC)
- SGX subsystem (TA_SGX module)
- L4-Core interconnect (TA_L4_CORE module)
- L4-Per interconnect (TA_L4_PER module)
- L4-Emu interconnect (TA_L4_EMU module)

Table 5-40 through Table 5-43 lists all initiator target registers and their physical addresses depending on the module instance.

Table 5-44 through Table 5-47 describe the individual common registers in the module instance.

Table 5-40. Target Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	TA_SMS Physical Address	TA_GPMC Physical Address	TA_OCM_RAM Physical Address	Section
L3_TA_AGENT_CONTROL	RW	64	0x6800 2020	0x6800 2420	0x6800 2820	Section 5.6.4.1
L3_TA_AGENT_STATUS	R	64	0x6800 2028	0x6800 2428	0x6800 2828	Section 5.6.4.2
L3_TA_ERROR_LOG	RW	64	0x6800 2058	0x6800 2458	0x6800 2858	Section 5.6.4.3
L3_TA_ERROR_LOG_ADDR	R	64	0x6800 2060	0x6800 2460	0x6800 2860	Section 5.6.4.4

Table 5-41. Target Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	TA_OCM_ROM Physical Address	Section
L3_TA_AGENT_CONTROL	RW	64	0x6800 2C20	Section 5.6.4.1
L3_TA_AGENT_STATUS	R	64	0x6800 2C28	Section 5.6.4.2
L3_TA_ERROR_LOG	RW	64	0x6800 2C58	Section 5.6.4.3
L3_TA_ERROR_LOG_ADDR	R	64	0x6800 2C60	Section 5.6.4.4

Table 5-42. Target Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	TA_IPSS Physical Address	TA_SGX Physical Address	Section
L3_TA_AGENT_CONTROL	RW	64	0x6800 6020	0x6800 6420	Section 5.6.4.1
L3_TA_AGENT_STATUS	R	64	0x6800 6028	0x6800 6428	Section 5.6.4.2
L3_TA_ERROR_LOG	RW	64	0x6800 6058	0x6800 6458	Section 5.6.4.3
L3_TA_ERROR_LOG_ADDR	R	64	0x6800 6060	0x6800 6460	Section 5.6.4.4

Table 5-43. Target Agent Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	TA_L4_CORE Physical Address	TA_L4_PER Physical Address	TA_L4_EMU Physical Address	Section
L3_TA_AGENT_CONTROL	RW	64	0x6800 6820	0x6800 6C20	0x6800 7020	Section 5.6.4.1
L3_TA_AGENT_STATUS	RW	64	0x6800 6828	0x6800 6C28	0x6800 7028	Section 5.6.4.2

Table 5-43. Target Agent Common Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	TA_L4_CORE Physical Address	TA_L4_PER Physical Address	TA_L4_EMU Physical Address	Section
L3_TA_ERROR_LOG	RW	64	0x6800 6858	0x6800 6C58	0x6800 7058	Section 5.6.4.3
L3_TA_ERROR_LOG_ADDR	R	64	0x6800 6860	0x6800 6C60	0x6800 7060	Section 5.6.4.4

5.6.4 L3 Target Agent (L3 TA) Register Descriptions

5.6.4.1 L3_TA_AGENT_CONTROL

Table 5-44. L3_TA_AGENT_CONTROL

Address Offset	0x020
Physical Address	Please refer from Table 5-40 to Table 5-43
Description	Agent control register of TA block.
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						REQ_TIMEOUT_REP	SERROR_REP	Reserved													REQ_TIMEOUT			Reserved			REJECT	Reserved			CORE_RESET

Bits	Field Name	Description	Type	Reset
63:26	Reserved	Reserved	R	0x0000000000
25	REQ_TIMEOUT_REP	Request Timeout Reporting 0x0: No special reporting 0x1: Report out of band	RW	1
24	SERROR_REP	SError reporting 0x0: Suppress Serror reporting 0x1: Report Serror	RW	1
	Reserved for instances 1, 3 to 6	Reserved	R	0x0000
23:11	Reserved	Reserved	R	0x0000
10:8	REQ_TIMEOUT	Request Timeout Bound: 0x0: No timeout 0x1: 1x base cycles 0x2: 4x base cycles 0x3: 16x base cycles 0x4: 64x base cycles	RW	0x0
7:5	Reserved	Reserved	R	0x0
4	REJECT	Request rejection control 0x0: Request rejection control 0x1: Block requests to this target	RW	0
3:1	Reserved	Reserved	R	0x0
0	CORE_RESET	Reset output on core 0x0: Inactive 0x1: Reset control active	RW	0

5.6.4.2 L3_TA_AGENT_STATUS

Table 5-45. L3_TA_AGENT_STATUS

Address Offset	0x028
Physical Address	Please refer from Table 5-40 to Table 5-43
Description	Agent Status Register.
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SERROR	Reserved							BURST_CLOSE	TIMEBASE				Reserved			REQ_TIMEOUT	READEX	BURST	RESP_ACTIVE	REQ_WAITING	Reserved			CORE_RESET

Bits	Field Name	Description	Type	Reset
63:25	Reserved	Reserved	R	0x0000000000
24	SERROR	Serror assertion detected	RW	0
	Reserved for instances 1, 3 to 6	Reserved	R	0
23:17	Reserved	Reserved	R	0x00
16	BURST_CLOSE	Forced burst close status Read 0x0: Normal operation Read 0x1: Burst close command	R	0
15:12	TIMEBASE	Observation of timebase signals.	R	0x0
11:9	Reserved	Reserved	R	0x0
8	REQ_TIMEOUT	Request timeout status Read 0x0: Normal operation Read 0x1: Request timed out, responding ERR to all the requests	R	0
7	READEX	Status of readEx/Write Read 0x0: No pending ReadEx Read 0x1: ReadEx pending on at lease one thread	R	0
6	BURST	Status of open burst Read 0x0: No open burst Read 0x1: Open burst on at least one thread	R	0
5	RESP_ACTIVE	Responses outstanding Read 0x0: No responses outstanding Read 0x1: Response outstanding in the target	R	0
4	REQ_WAITING	Requests waiting Read 0x0: No request waiting Read 0x1: Request waiting for acceptance by target	R	0
3:1	Reserved	Reserved	R	0x0
0	CORE_RESET	Reset input from core interface Read 0x0: Reset inactive Read 0x1: Reset active	R	0

5.6.4.3 L3_TA_ERROR_LOG

Table 5-46. L3_TA_ERROR_LOG

Address Offset	0x058
Physical Address	Please refer from Table 5-40 to Table 5-43
Description	Error log register of TA block - logs error detected by a target agent.
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																								REQ_INFO							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MULTI	Reserved			CODE				Reserved								INITID								Reserved					CMD		

Bits	Field Name	Description	Type	Reset
63:42	Reserved	Reserved	R	0x0000
41:32	REQ_INFO	MReqInfo bits of command that caused the error	R	0x0000
31	MULTI	Multiple Errors Write 0x0: Ignored Read 0x0: Multiple error not seen Write 0x1: Clear MULTI flag Read 0x1: Multiple error seen	RW	0
30:28	Reserved	Reserved	R	0x0
27:24	CODE	Error code	RW	0x0
23:16	Reserved	Reserved	R	0x00
15:8	INITID	Initiator ID from which command was launched	R	0x00
7:3	Reserved	Reserved	R	0x00
2:0	CMD	Command that caused the error	R	0x0

5.6.4.4 L3_TA_ERROR_LOG_ADDR

Table 5-47. L3_TA_ERROR_LOG_ADDR

Address Offset	0x060
Physical Address	Please refer from Table 5-40 to Table 5-43
Description	Error log address register of TA block
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR																															

Bits	Field Name	Description	Type	Reset
63:40	Reserved	Reserved	R	0x000000
31:0	ADDR	Address of the command that caused the error	R	0x0000000000

5.6.5 Register Target (RT) Register Mapping Summary

This section describes the RT module.

Table 5-48 lists the RT registers and their physical addresses.

Table 5-49 through Table 5-51 describe the individual registers in the module instance.

Table 5-48. RT Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
L3_RT_NETWORK	R	64	0x010	0x6800 0010	Section 5.6.6.1
L3_RT_INITID_READBACK	R	64	0x070	0x6800 0070	Section 5.6.6.2
L3_RT_NETWORK_CONTROL	RW	64	0x078	0x6800 0078	Section 5.6.6.3

5.6.6 Register Target (RT) Register Descriptions

5.6.6.1 L3_RT_NETWORK

Table 5-49. L3_RT_NETWORK

Address Offset	0x010		
Physical address	0x6800 0010	**Instance**	RT
Description	This register identifies the interconnect and is present only in the register target.		
Type	R		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ID																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															

Bits	Field Name	Description	Type	Reset
63:32	ID	Unique Interconnect ID	R	0x00000000
31:0	Reserved	Reserved	R	0x00000000

5.6.6.2 L3_RT_INITID_READBACK

Table 5-50. L3_RT_INITID_READBACK

Address Offset	0x070		
Physical address	0x6800 0070	**Instance**	RT
Description	This register is used by initiators to discover their own identity.		
Type	R		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

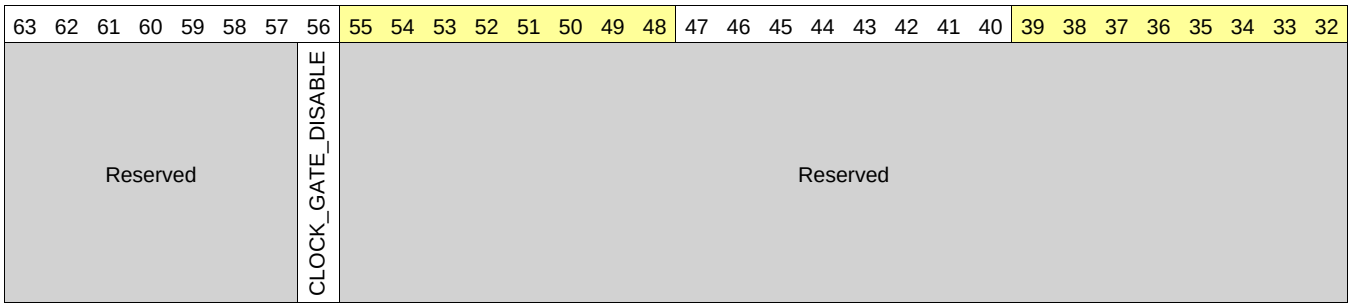
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								INITID							

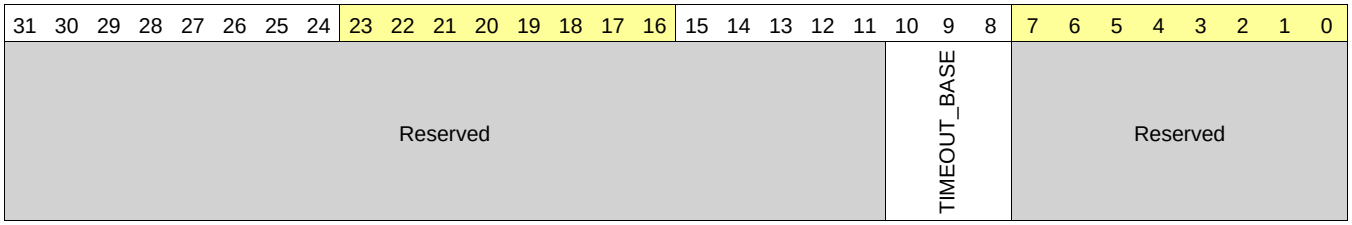
Bits	Field Name	Description	Type	Reset
63:8	Reserved	Reserved	R	0x00000000000000
7:0	INITID	Returns initiator ID of core thread that initiated the read	R	0x18

5.6.6.3 L3_RT_NETWORK_CONTROL

Table 5-51. L3_RT_NETWORK_CONTROL

Address Offset	0x068		
Physical address	0x6800 0078	**Instance**	RT
Description	It controls such interconnect wide functions as the timeout base scale and the disabling of fine grained hardware clock gating.		
Type	RW		





Bits	Field Name	Description	Type	Reset
63:57	Reserved	Reserved	R	0x00
56	CLOCK_GATE_ DISABLE	Overrides fine grained hardware clock gating	RW	0
55:11	Reserved	Reserved	R	0x000000000000
10:8	TIMEOUT_BASE	Timeout base period in register target clock cycles Program the timeout base period. Each of the agent timeout features is programmed as a multiple of the timeout base period. These timeout bases are: 0x0: Timeout disabled 0x1: L3 interconnect clock cycles divided by 64 0x2: L3 interconnect clock cycles divided by 256 0x3: L3 interconnect clock cycles divided by 1024 0x4: L3 interconnect clock cycles divided by 4096	RW	0x0
7:0	Reserved	Reserved	R	0x00

5.6.7 Protection Mechanism (PM) Register Mapping Summary

This section describes the protection mechanism register block.

The following are the protection mechanism registers:

- Register target (PM_RT module)
- General-purpose memory controller (PM_GPMC module)
- On-chip RAM (PM_OCM_RAM module)
- On-chip ROM (PM_OCM_ROM module)
- IP-Subsystem (PM_IPSS module)

Table 5-52 and Table 5-54 list the protection registers and their physical addresses, depending on the module instance.

Table 5-55 through Table 5-60 describe the individual common registers in the module instance.

Table 5-52. Protection Mechanism Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PM_RT Physical Address	PM_GPMC Physical Address	Section
L3_PM_ERROR_LOG	RW	64	0x6801 0020	0x6801 2420	Section 5.6.8.1
L3_PM_CONTROL	RW	64	0x6801 0028	0x6801 2428	Section 5.6.8.2
L3_PM_ERROR_CLEAR_SINGLE	R	64	0x6801 0030	0x6801 2430	Section 5.6.8.3
L3_PM_ERROR_CLEAR_MULTI	R	64	0x6801 0038	0x6801 2438	Section 5.6.8.4
L3_PM_REQ_INFO_PERMISSION_i[1]	RW	64	0x6801 0048 + (0x20*i)	0x6801 2448 + (0x20*i)	Section 5.6.8.5
L3_PM_READ_PERMISSION_i[1]	RW	64	0x6801 0050 + (0x20*i)	0x6801 2450 + (0x20*i)	Section 5.6.8.6
L3_PM_WRITE_PERMISSION_i[1]	RW	64	0x6801 0058 + (0x20*i)	0x6801 2458 + (0x20*i)	Section 5.6.8.7
L3_PM_ADDR_MATCH_k[2]	RW	64	0x6801 0060 + (0x20*k)	0x6801 2460 + (0x20*k)	Section 5.6.8.9

(1) i = 0 to 1 for PM_RT
i = 0 to 7 for PM_GPMC

(2) k = 1 to 1 for PM_RT
k = 1 to 7 for PM_GPMC

Table 5-53. Protection Mechanism Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PM_OCM_RAM Physical Address	PM_OCM_ROM Physical Address	Section
L3_PM_ERROR_LOG	RW	64	0x6801 2820	0x6801 2C20	Section 5.6.8.1
L3_PM_CONTROL	RW	64	0x6801 2828	0x6801 2C28	Section 5.6.8.2
L3_PM_ERROR_CLEAR_SINGLE	R	64	0x6801 2830	0x6801 2C30	Section 5.6.8.3
L3_PM_ERROR_CLEAR_MULTI	R	64	0x6801 2838	0x6801 2C38	Section 5.6.8.4
L3_PM_REQ_INFO_PERMISSION_i[1]	RW	64	0x6801 2848 + (0x20*i)	0x6801 2C48 + (0x20*i)	Section 5.6.8.5
L3_PM_READ_PERMISSION_i[1]	RW	64	0x6801 2850 + (0x20*i)	0x6801 2C50 + (0x20*i)	Section 5.6.8.6
L3_PM_WRITE_PERMISSION_i[1]	RW	64	0x6801 2858 + (0x20*i)	0x6801 2C58 + (0x20*i)	Section 5.6.8.7
L3_PM_ADDR_MATCH_k[2]	RW	64	0x6801 2860 + (0x20*k)	0x6801 2C60 + (0x20*k)	Section 5.6.8.9

(1) i = 0 to 1 for PM_OCM_ROM
i = 0 to 7 for PM_OCM_RAM

(2) k = 1 to 1 for PM_OCM_ROM
k = 1 to 7 for PM_OCM_RAM

Table 5-54. Protection Mechanism Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PM_IPSS Subsystem Physical Address	Section
L3_PM_ERROR_LOG	RW	64	0x6801 4020	Section 5.6.8.1

Table 5-54. Protection Mechanism Common Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	PM_IPSS Subsystem Physical Address	Section
L3_PM_CONTROL	RW	64	0x6801 4028	Section 5.6.8.2
L3_PM_ERROR_CLEAR_SINGLE	R	64	0x6801 4030	Section 5.6.8.3
L3_PM_ERROR_CLEAR_MULTI	R	64	0x6801 4038	Section 5.6.8.4
L3_PM_REQ_INFO_PERMISSION_i[(1)]	RW	64	0x6801 4048 + (0x20*i)	Section 5.6.8.5
L3_PM_READ_PERMISSION_i[(1)]	RW	64	0x6801 4050 + (0x20*i)	Section 5.6.8.6
L3_PM_WRITE_PERMISSION_i[(1)]	RW	64	0x6801 4058 + (0x20*i)	Section 5.6.8.7
L3_PM_ADDR_MATCH_k[(2)]	RW	64	0x6801 4060 + (0x20*k)	Section 5.6.8.9

(1) i = 0 to 3 for PM_IPSS
(2) k = 1 to 3 for PM_IPSS

5.6.8 Protection Mechanism (PM) Register Descriptions

5.6.8.1 L3_PM_ERROR_LOG

Table 5-55. L3_PM_ERROR_LOG

Address Offset	0x20
Physical address	Please refer from Table 5-52 to Table 5-54
Description	This register logs errors detected by the protection mechanism.
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MULTI	SECONDARY	Reserved		CODE				Reserved			REQ_INFO					INITID								Reserved	REGION			Reserved	CMD		

Bits	Field Name	Description	Type	Reset
63:32	Reserved	Reserved	R	0x00000000
31	MULTI	Multiple errors 0x0:Multiple errors not seen 0x1:Multiple errors seen	RW1toClr	0
30	SECONDARY	Secondary error present	RW1toClr	0
29:28	Reserved	Reserved	R	0x0
27:24	CODE	Error Code see Table 5-25	RW1toClr	0x0
23:21	Reserved	Reserved	R	0x0
20:16	REQ_INFO	MReqInfo bits of command selected for protection checking see Table 5-21	R	0x00
15:8	INITID	Initiator ID from which the command was launched see Table 5-18	R	0x00
7	Reserved	Reserved	R	0
6:4	REGION	Protection region number that command mapped to	R	0x0
3	Reserved	Reserved	R	0
2:0	CMD	Command that caused the error see Table 5-1	R	0x0

5.6.8.2 L3_PM_CONTROL

Table 5-56. L3_PM_CONTROL

Address Offset	0x28
Physical address	Please refer from Table 5-52 to Table 5-54
Description	This register controls protection mechanism functions such as error reporting.
Type	RW

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ERROR_SECONDARY_REP	ERROR_REP	Reserved																							

Bits	Field Name	Description	Type	Reset
63:26	Reserved	Reserved	R	0x0000000000
25	ERROR_SECONDARY_REP	Out of band error reporting 0x0: Out of band error reporting suppress 0x1: Out of band error report	RW	1
24	ERROR_REP	Out of band error reporting 0x0: Out of band error reporting suppress 0x1: Out of band error report	RW	1
23:0	Reserved	Reserved	R	0x000000

5.6.8.3 L3_PM_ERROR_CLEAR_SINGLE

Table 5-57. L3_PM_ERROR_CLEAR_SINGLE

Address Offset	0x30
Physical address	Please refer from Table 5-52 to Table 5-54
Description	Read to clear single errors from error log
Type	R

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															CLEAR

Bits	Field Name	Description	Type	Reset
63:1	Reserved	Reserved	R	0x0000000000000000
0	CLEAR	Clear single error from log	R	0

5.6.8.4 L3_PM_ERROR_CLEAR_MULTI

Table 5-58. L3_PM_ERROR_CLEAR_MULTI

Address Offset	0x38
Physical address	Please refer from Table 5-52 to Table 5-54
Description	Read to clear multiple errors from error log
Type	R

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															CLEAR

Bits	Field Name	Description	Type	Reset
63:1	Reserved	Reserved	R	0x0000000000000000
0	CLEAR	Clear multiple error from log	R	0

5.6.8.5 L3_PM_REQ_INFO_PERMISSION_i

Table 5-59. L3_PM_REQ_INFO_PERMISSION_i

Address Offset	0x38
Physical address	Please refer from Table 5-52 to Table 5-54
Description	It configures a protection region's permissions using the MReqInfo bits selected for the PM by the structural configuration.
Type	R/W

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																REQ_INFO															

Bits	Field Name	Description	Type	Reset
63:16	Reserved	Reserved	R	0x000000000000
15:0	REQ_INFO	Request info permission bits for region 0	RW	See Table 5-60.

Table 5-60. Reset Value for REQ_INFO_PERMISSION

	Regions							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
PM_RT	0xFFFF	0x000						
PM_GPMC	0x0000	0x----	0x----	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF
PM_OCM_RAM	0x0000	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF	0xFFFF
PM_OCM_ROM	0x----	0xFFFF						
PM_IPSS	0x0000	0xFFFF	0xFFFF	0xFFFF				

5.6.8.6 L3_PM_READ_PERMISSION_i

Table 5-61. L3_PM_READ_PERMISSION_i

Address Offset	0x050 + (0x20+i)	**Index**	i = 0 to 1 for PM_RT i = 0 to 7 for PM_GPMC i = 0 to 1 for PM_OCM_ROM i = 0 to 3 for PM_IPSS i = 0 to 7 for PM_OCM_RAM
Physical address	Please refer from Table 5-52 to Table 5-54		
Description	It configures protection region permissions for read incoming commands.		
Type	RW		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SGX	Reserved	DAP	Reserved			USB_HS_HOST	DISP_SS	Reserved			IPSS	SDMA	Reserved	MPU	Reserved

Bits	Field Name	Description	Type	Reset
63:15	Reserved	Reserved	R	0
14	SGX	Read permission for the SGX	RW	see Table 5-63
13	Reserved	Reserved	RW	0
12	DAP	Read permission for the DAP	RW	see Table 5-63

Bits	Field Name	Description	Type	Reset
11:10	Reserved	Reserved	R	0
9	USB_HS_HOST	Read permission for the USB_HS_Host	RW	see Table 5-63
8	DISP_SS	Write permission for the DISPLAY SS	RW	see Table 5-63
7:5	Reserved	Reserved	RW	0
4	IPSS	Read permission for the IP-Subsystem (Camera-VPFE, USBOTG, and EMAC).	RW	see Table 5-63
3	SDMA	Read permission for the system DMA	RW	see Table 5-63
2	Reserved	Reserved	R	0
1	MPU	Read permission for the MPU	RW	see Table 5-63
0	Reserved	Reserved	RW	0

5.6.8.7 L3_PM_WRITE_PERMISSION_i

Table 5-62. L3_PM_WRITE_PERMISSION_i

Address Offset	0x058 + (0x20+i)	**Index**	i = 0 to 1 for PM_RT i = 0 to 7 for PM_GPMC i = 0 to 1 for PM_OCM_ROM i = 0 to 3 for PM_IPSS i = 0 to 7 for PM_OCM_RAM
Physical address	Please refer from Table 5-52 to Table 5-54		
Description	It configures protection region permissions for write incoming commands.		
Type	RW		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																SGX	Reserved	DAP	Reserved			USB_HS_Host	DISP_SS	Reserved			IPSS	SDMA	Reserved	MPU	Reserved

Bits	Field Name	Description	Type	Reset
63:15	Reserved	Reserved	R	0
14	SGX	Write permission for the SGX	RW	see Table 5-63
13	Reserved	Reserved	RW	0
12	DAP	Write permission for the DAP	RW	see Table 5-63
11:10	Reserved	Reserved	R	0
9	USB_HS_Host	Write permission for the USB_HS_Host	RW	see Table 5-63
8	DISP_SS	Write permission for the DISPLAY SS	RW	see Table 5-63
7:5	Reserved	Reserved	RW	0
4	IPSS	IP-Subsystem (Camera-VPFE, USBOTG, and EMAC).	RW	see Table 5-63
3	SDMA	Write permission for the system DMA	RW	see Table 5-63
2	Reserved	Reserved	R	0
1	MPU	Write permission for the MPU	RW	see Table 5-63
0	Reserved	Reserved	RW	0

5.6.8.8 Bit Availability and Initialization Values for L3_PM_READ_PERMISSION_i and L3_PM_WRITE_PERMISSION_i

Table 5-63 shows bit available in L3_PM_READ_PERMISSION_i and L3_PM_WRITE_PERMISSION_i registers. All N/A are considered as reserved bits with a Read access.

Table 5-63. Bit Availability and Initialization Values for L3_PM_READ_PERMISSION_i and L3_PM_WRITE_PERMISSION_i[1]

PM	BITS							
	1: MPU	3: SDMA	4: IPSS_ OTG	8: DISP SS	9: USB_HS_ Host	12: DAP	14: SGX	63-15: Reserved
PM_RT region 0 to 1	1	N/A	N/A	N/A	N/A	1	N/A	0x0000000000000
PM_GPMC region 0 to 7	1	1	1	N/A	1	1	1	0x0000000000001
PM_OCM_RAM region 0 to 7	1	1	1	1	1	1	1	0x0000000000001
PM_OCM_ROM region 0 to 1 (ro)[2]	1	N/A	N/A	N/A	N/A	1	N/A	0x0000000000000
PM_IP Subsystem region 0 to 3	1	1	N/A	N/A	N/A	1	N/A	0x0000000000000

(1) Some firewall reset value are exported from the system control module. Values in this table are valid for GP device only. HS value may differ.

(2) ROM is a read only memory; therefore, the write permission is set to 0. The value for Region 0 is exported from control module (PLATFORM_MPU_OCMROM_DT_FW_RD).

5.6.8.9 L3_PM_ADDR_MATCH_k

Table 5-64. L3_PM_ADDR_MATCH_k

Address Offset	0x060 + (0x20+k)	**Index**	k = 1 to 1 for PM_RT k = 1 to 7 for PM_GPMC k = 1 to 1 for PM_OCM_ROM k = 1 to 3 for PM_IPSS k = 1 to 7 for PM_OCM_RAM
Physical address	Please refer from Table 5-52 to Table 5-54		
Description			
Type	R		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BASE_ADDR														LEVEL	Reserved	SIZE						ADDR_SPACE	

Bits	Field Name	Description	Type	Reset
63:20	Reserved	Reserved	R	0x00000000000
19:10	BASE_ADDR	Protection region base address	R	see Table 5-65
9	LEVEL	Protection region level.	R	see Table 5-65
8	Reserved	Reserved	R	0
7:3	SIZE	Protection region size	R	see Table 5-65
2:0	ADDR_SPACE	Protection region address space	R	see Table 5-65

Table 5-65. Reset Value for L3_PM_ADDR_MATCH_k[1]

PM	Region	BASE ADDRESS	LEVEL	SIZE	ADDR_SPACE
PM_RT	1	0x040	0x1	0x06	0x0
PM_GPMC	1	0x000	0x0	0x23	0x0
	2	0x000	0x0	0x00	0x0
	3	0x000	0x0	0x00	0x0
	4	0x000	0x0	0x00	0x0
	5	0x000	0x0	0x00	0x0
	6	0x000	0x0	0x00	0x0
	7	0x000	0x0	0x00	0x0
PM_OCM_RAM	1	0x000	0x0	0x00	0x0
	2	0x03E	0x0	0x02	0x0
	3	0x000	0x0	0x00	0x0
	4	0x000	0x0	0x00	0x0
	5	0x000	0x0	0x00	0x0
	6	0x000	0x0	0x00	0x0
	7	0x000	0x0	0x00	0x0
PM_OCM_ROM	1	0x050	0x0	0x05	0x0
	2	0x000	0x0	0x00	0x0
	3	0x000	0x0	0x00	0x0

(1) Some firewall reset value are exported from the system control module. Values in this table are valid for GP device only. HS value may differ.

Table 5-65. Reset Value for L3_PM_ADDR_MATCH_k[1] (continued)

PM	Region	BASE ADDRESS	LEVEL	SIZE	ADDR_SPACE
PM_IPSS	1	0x000	0x0	0x00	0x0
	2	0x000	0x0	0x00	0x0
	3	0x000	0x0	0x00	0x0

5.6.9 Sideband Interconnect (SI) Register Mapping Summary

This section describes the sideband interconnect register block.

Table 5-66 lists the SI registers and their physical addresses.

Table 5-67 through Table 5-69 describe the individual registers in the module instance.

Table 5-66. SI Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
L3_SI_CONTROL	RW	64	0x020	0x6800 0420	Section 5.6.10.1
L3_SI_FLAG_STATUS_0	R	64	0x110	0x6800 0510	Section 14.6.2.15
L3_SI_FLAG_STATUS_1	R	64	0x130	0x6800 0530	Section 5.6.10.3

5.6.10 Sideband Interconnect (SI) Register Descriptions

5.6.10.1 L3_SI_CONTROL

Table 5-67. L3_SI_CONTROL

Address Offset	0x020		
Physical address	0x6800 0420	**Instance**	SI
Description	Control of register and sideband interconnect		
Type	RW		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Reserved							CLOCK_GATE_DISABLE	Reserved																							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															

Bits	Field Name	Description	Type	Reset
63:57	Reserved	Reserved for future use	R	0x00
56	CLOCK_GATE_DISABLE	Overrides fine grained hardware clock gating in register and sideband interconnect 0x0: Normal clock gating 0x1: Clock gating disabled	RW	0
55:0	Reserved	Reserved for future use	R	0x00000000000000

5.6.10.2 L3_SI_FLAG_STATUS_0

Table 5-68. L3_SI_FLAG_STATUS_0

Address Offset	0x110		
Physical address	0x6800 0510	**Instance**	SI
Description	They are used to observe the individual bits that make up a composite interconnect flag.		
Type	R		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
STATUS																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATUS																															

Bits	Field Name	Description	Type	Reset
63:0	STATUS	Status of sideband signals making up composite interconnect flag for application. See Table 5-28	R	0x0000000000000000

5.6.10.3 L3_SI_FLAG_STATUS_1

Table 5-69. L3_SI_FLAG_STATUS_1

Address Offset	0x130		
Physical address	0x6800 0530	**Instance**	SI
Description	They are used to observe the individual bits that make up a composite interconnect flag.		
Type	R		

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
STATUS																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATUS																															

Bits	Field Name	Description	Type	Reset
63:0	STATUS	Status of sideband signals making up composite interconnect flag for debug. See Table 5-29.	R	0x0000000000000000

5.7 L4 Interconnects

5.7.1 Overview

To connect peripheral modules, the device uses four separate L4 interconnect structures. Although all L4 interconnects handle transfers with peripherals, the interconnects are in different power domains. The L4 interconnect is composed of the following:

- L4-Core: Includes the majority of the peripherals and the configuration interface for L3 interconnect system modules
- L4-Per: Includes peripherals that do not need to be mapped in the CORE power domain
- L4-Wakeup: Includes the peripherals attached to the WKUP power domain
- L4-Emu: Includes emulation peripherals attached to the EMU power domain

The following are the main features of the L4 interconnects:

- Single port to connect to L3 interconnect
  - L4-Core
  - L4-Per
- Dual ports for the following:
  - L4-Emu to connect to the L3 interconnect and DAP
  - L4-Wakeup to connect to the L4-Core and L4-Emu
- Single 32-bit initiator for the L3 port
- Multi target ports (one per target interface on the L4 interconnect)
- 8-, 16-, or 32-bit data, single, or burst transactions
- Little-endian
- Non-blocking with fair arbitration between threads
- Target interfaces: Fully synchronous or divided synchronous
- Peak bandwidth of 2x M bytes/sec of L4 frequency (in MHz)
- Latency: Three cycles on request, one cycle on response
- Security logic provides user-configurable access control to targets by each initiator:
  - Firewall in L4-Core protects the core and wake-up peripherals.
  - Firewall in L4-Per protects per peripherals.
  - Firewall in L4-Emu protects emulation and wake-up peripherals

NOTE: L4-Wakeup has two input ports from L4-Core and L4-Emu. Therefore, wake-up peripherals appear in two locations in the memory mapping. Normally, L4-Emu limits its access except when debugging.

Figure 5-12 shows an overview of the L4 interconnects and the peripherals attached to them.

Figure 5-12. L4 Interconnect Overview

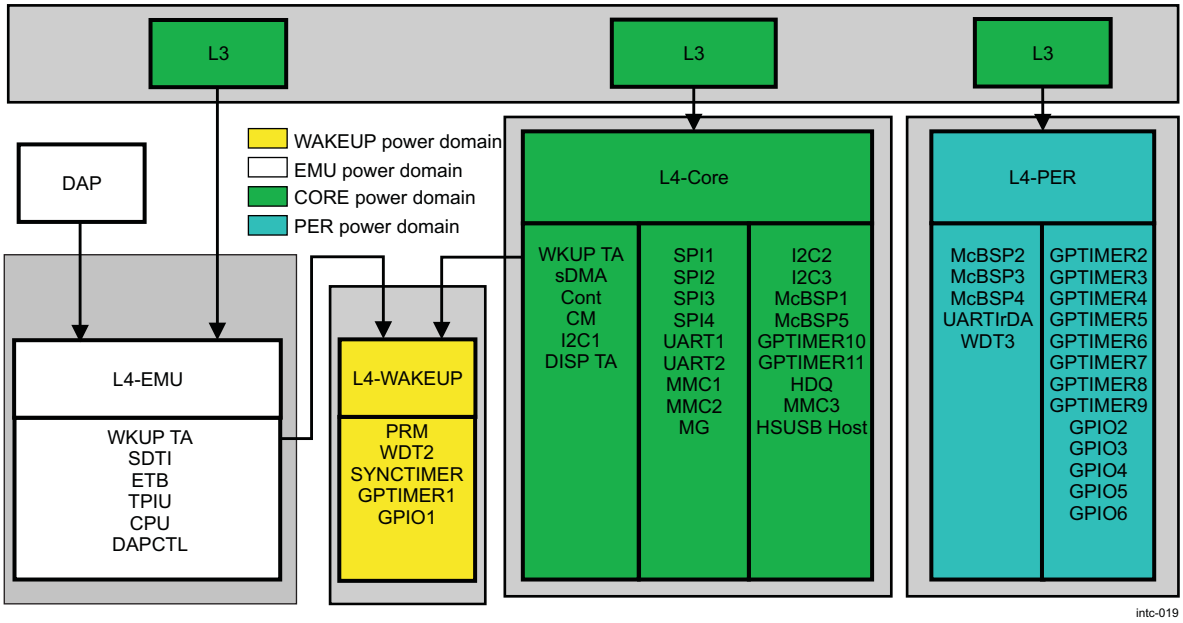
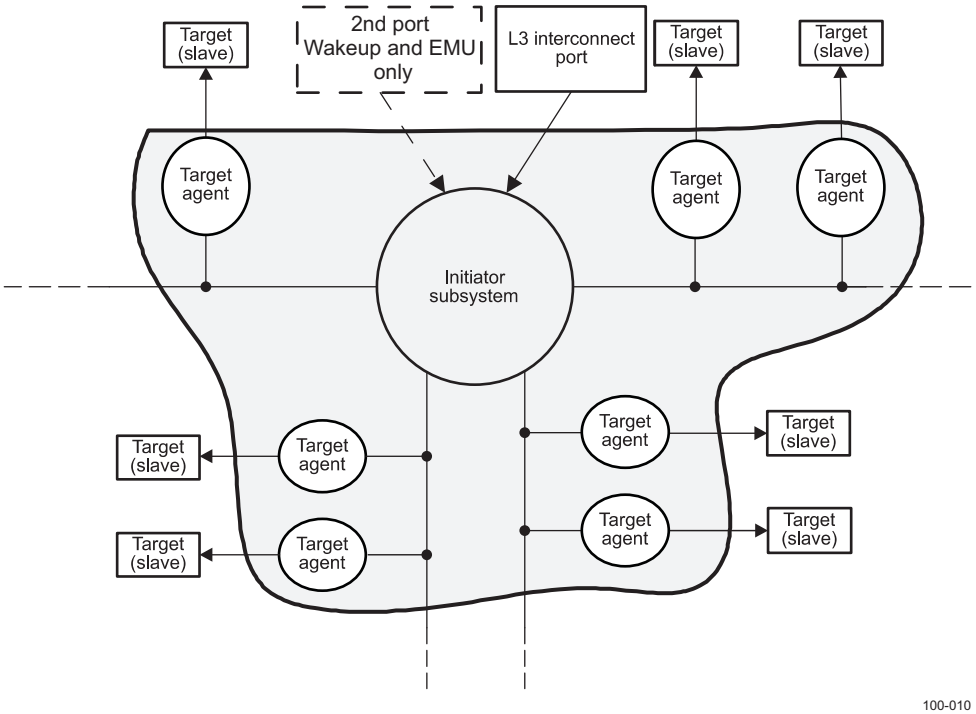


Figure 5-13 shows an internal view of the L4 interconnects in the overall interconnect. This architecture, with only one initiator module (the initiator subsystem) distributing transactions to all target modules (peripherals), enables the firewall functions of the L4 interconnects to be centralized at the L4 initiator level. The L4 firewall filters the accesses according to the configurable protection groups defined in the L4 address protection (AP) registers. Each module or agent is assigned to a protection group. Configuration is also defined in the L4 AP and is programmable on a module-per-module basis.

Figure 5-13. L4 Initiator-Target Connectivity for L4-Core and L4-Per



NOTE: As Figure 5-13 shows, targets are attached to branches. These branches have no functional effect and are present for timing closure reasons only.

5.7.1.1 L4-Core Interconnect

The L4-core interconnect handles only transfers to peripherals in the CORE power domain. Table 5-70 lists the TAs.

Table 5-70. L4-Core Target Agents

Module Name	Description
Display subsystem	Display subsystem configuration port
USBHS Host	Universal serial bus High-Speed port Host
USBTLL	USB Transceiver Less Link
UART1	Universal asynchronous receiver transmitter port 1
UART2	Universal asynchronous receiver transmitter port 2
I2C1	Multimaster inter-integrated circuit 1
I2C2	Multimaster inter-integrated circuit 2
I2C3	Multimaster inter-integrated circuit 3
McBSP1	Multichannel buffered serial port 1
McBSP5	Multichannel buffered serial port 5
GPTIMER10	General-purpose timer 10
GPTIMER11	General-purpose timer 11
SPI1	Serial peripheral interface 1
SPI2	Serial peripheral interface 2
MMCHS1	Multimedia memory controller SDIO 1
MMCHS2	Multimedia memory controller SDIO 2
MMCHS3	Multimedia memory controller SDIO 3
HDQ/1-Wire	Single wire serial link low rate
MG	MagicGate
SPI1	Serial peripheral interface 1
SPI2	Serial peripheral interface 2
SPI3	Serial peripheral interface 3
SPI4	Serial peripheral interface 4
sDMA	System DMA controller
L4-Wakeup	L4-Wakeup interconnect
CM	Clock manager
SCM	System control module

NOTE: A unique port is used for communication between the L3 interconnect and the L4-Core interconnect to allow the L3 initiators to access the L4-Core targets.

For the list of initiators authorized to access the L4-Core peripherals, see Table 5-14. For details on restricted access, see Section 5.9.3.1, *Protection Mechanism*.

5.7.1.2 L4-Per Interconnect

The L4-Per interconnect handles only transfers to peripherals in the PER power domain. Table 5-71 lists the TAs.

Table 5-71. L4-Per Target Agents

Module Name	Description
UARTIrDA	Universal asynchronous receiver/transmitter and infrared data association port
McBSP2	Multichannel buffered serial port 2
McBSP3	Multichannel buffered serial port 3
GPTIMER2	General-purpose timer 2
GPTIMER3	General-purpose timer 3
GPTIMER4	General-purpose timer 4
GPTIMER5	General-purpose timer 5
GPTIMER6	General-purpose timer 6
GPTIMER7	General-purpose timer 7
GPTIMER8	General-purpose timer 8
GPTIMER9	General-purpose timer 9
GPIO2	General-purpose I/O 2
GPIO3	General-purpose I/O 3
GPIO4	General-purpose I/O 4
GPIO5	General-purpose I/O 5
GPIO6	General-purpose I/O 6

NOTE: A unique port is used for communication between the L3 interconnect and the L4-Core interconnect to allow the L3 initiators to access the L4-Per targets.

For the list of initiators authorized to access the L4-Per peripherals, see Table 5-14. For details on restricted access, see Section 5.9.3.1, *Protection Mechanism*.

5.7.1.3 L4-Emu Interconnect

The L4-Emu interconnect handles only transfers to peripherals in the EMU power domain. Table 5-72 lists the TAs.

Table 5-72. L4-Emu Target Agents

Module Name	Description
L4-Wakeup	L4-Wakeup interconnect
SDTI	System debug trace interface
ETB	Embedded trace buffer
TPIU	Trace port interface unit
DAPCTL	Debug access port

Not all initiators can access all the targets in the L4-Emu interconnect. Additional restrictions affect the ability of these initiators to access the L4-Emu peripherals. Table 5-73 lists which initiators can access the L4-Emu interconnect.

NOTE: For the list of initiators authorized to access the L4-Emu peripherals, see Table 5-14. For details on restricted access, see Section 5.9.3.1, *Protection Mechanism*.

Table 5-73. L4-Emu Initiator Agents

Module Name	Description
L3 interconnect	L3 interconnect port
DAP	DAP port

5.7.1.4 L4-Wakeup Interconnect

The L4-Wakeup interconnect handles only transfers to peripherals in the WKUP power domain. Table 5-74 lists the TAs.

Table 5-74. L4-Wakeup Target Agents

Module Name	Description
PRM	Power reset manager
GPIO1	General-purpose I/O 1
GPTIMER1	General-purpose timer 1
GPTIMER12[1]	General-purpose timer 12
WDTIMER1[1]	Secure watchdog timer
WDTIMER2	MPU subsystem watchdog timer
32KTIMER	32-kHz timer

(1) Not available in GP device.

Initiators that can access the L4-Core or L4-Emu can access all the targets in the L4-Wakeup interconnect. Table 5-75 lists the initiators that can access the L4-Wakeup interconnect. For details on restricted access, see Section 5.9.3.1, *Protection Mechanism*.

Table 5-75. L4-Wakeup Initiator Agents

Module Name	Description
L4-Core interconnect	L4-Core interconnect port
L4-Emu interconnect	L4-Emu interconnect port

5.8 L4 Interconnects Integration

5.8.1 Clocking, Reset, and Power-Management Scheme

5.8.1.1 Clocks

Four functional clocks are used in each L4 interconnect (see Table 5-76).

Table 5-76. L4 Interconnect Clocks

Clock	Frequency	Name	Comments
L4-Core interconnect clock	Up to Core_L3_ICLK/2	CORE_L4_GICLK	Source, control, and gating handled by PRCM module
L4-Per interconnect clock	Up to Core_L3_ICLK/2	PER_L4_GICLK	Source, control, and gating handled by PRCM module
L4-Emu clock		L4_EMU	Clock for emulation
L4-Wakeup interconnect clock		WKUP_L4_GICLK	Source, control, and gating handled by PRCM module

5.8.1.2 Resets

5.8.1.2.1 Hardware Reset

L4 interconnects receive a reset signal from the PRCM module, which is the reset signal to the CORE power domain. For more details, see the *Power, Reset, and Clock Management* chapter.

Table 5-77 lists the hardware reset for the L4-Core interconnect.

Table 5-77. L4 Interconnect Hardware Reset

Interconnect	Reset Domain
L4-Core interconnect	CORE_RST
L4-Per interconnect	PER_RST
L4-Wakeup interconnect	WKUP_RST
L4-Emu interconnect	EMU_RST

5.8.1.2.2 Software Reset

The L4 interconnects have hardware reset capabilities, but do not have software reset capabilities. The hardware reset capabilities are controlled by the PRCM module and are applied to the L4 interconnects and the connected L4 TAs and L4 target modules.

5.8.1.3 Power Domain

For more details on power voltage scaling, see the *Power, Reset, and Clock Management* chapter.

Table 5-78 lists the power domains for the L4-Core and L4-Wakeup interconnects.

Table 5-78. L4 Interconnect Power Domains

Interconnect	Power Domain
L4-Core interconnect	CORE
L4-Per interconnect	PER
L4-Emu interconnect	EMU
L4-Wakeup interconnect	WKUP

5.8.1.4 Power Management

5.8.1.4.1 Module Power-Saving

The L4 interconnect automatically performs internal clock autogating to reduce power consumption. Though not recommended, it is possible to deactivate clock autogating by writing 1 to the CLOCK_GATE_DISABLE bit L4_LA_NETWORK_CONTROL_H[24] of each L4 interconnect. Clock autogating is enabled by default.

5.8.1.4.2 System Power Management and Wakeup

As part of the system-wide power-management scheme, the L4 interconnect enters an idle state at the request of the PRCM module (for more information, see the *Power, Reset, and Clock Management* chapter). The L4 interconnect is always in smart-idle mode; that is, it goes into idle state after receiving the request from the PRCM module once all the transfer requests are complete. This functionality is handled by hardware. The L4 interconnect sends an acknowledge signal back to the PRCM module when it enters idle state.

5.9 L4 Interconnects Functional Description

5.9.1 L4-Interconnects Initiator Identification

In the device interconnect, a ConnID is an initiator module identifier. The L4 interconnect uses the same ConnID as L3.

5.9.2 Endianness Management

Both L4 interconnects are little-endian only. Any initiator accessing the L4 interconnect module must consider byte ordering and perform a conversion, if necessary.

5.9.3 L4 Security and Firewalls

5.9.3.1 Protection Mechanism

The following two parameters are used to set up access permission because of the large address spaces and the number of peripherals connected to the L4 interconnects:

- Programmable groups for initiators:
  - 8 protection groups for the L4-Core interconnect
  - 8 protection groups for the L4-Per interconnect
  - 6 protection groups for the L4-Emu interconnect
- Each segment is divided into regions of 2K bytes:
  - 100 regions for the L4-Core interconnect
  - 43 regions for the L4-Per interconnect
  - 26 regions for the L4-Emu interconnect

NOTE: Regions and segments are present for the L4-Wakeup interconnect but cannot be programmed. The L4-Wakeup protection is done through the L4-Core and L4-Emu interconnects.

A protection group is a group of targets that have the same security settings. Initiator access is defined by CONNID_BIT_VECTOR field L4_AP_PROT_GROUP_k_L[15:0]. The initiators have the same access permission to all regions in this group.

A region is programmed to allow access to a unique selectable protection group by using PROT_GROUP_ID field L4_AP_REGION_l_H[22:20].

NOTE: k denotes the protection group number.

l denotes the region number.

5.9.3.2 Protection Group

A protection group defines which initiators with a given MReqInfo can access the targets agent protected by this group.

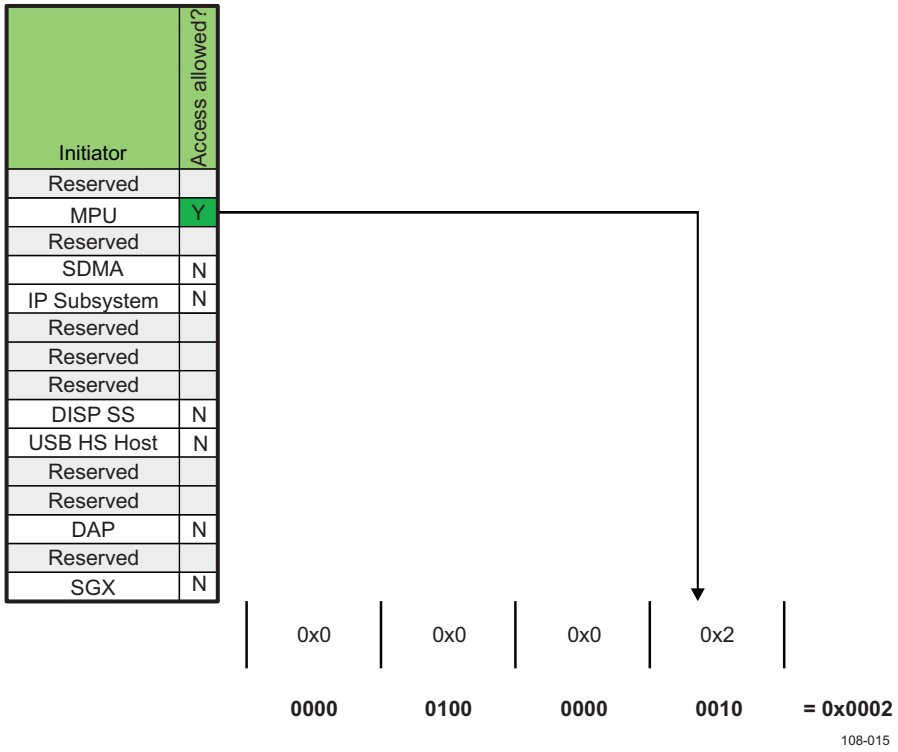
The CONNID_BIT_VECTOR field L4_AP_PROT_GROUP_MEMBERS_k[15:0] (see Figure 5-14) is a 1-bit vector that sets up initiator permission access regions. A protection group is accessible by an initiator if the bit position corresponding to its ConnID is set to one in the CONNID_BIT_VECTOR field.

The ENABLE field L4_AP_PROT_GROUP_ROLES_k[31:0] lists all possible MReqInfo combinations. Setting a Req bit in this register determines the type of access allowed to the initiator. See Section 5.4.3.4, *REQ_INFO_PERMISSION Configuration*, for more information. MReqInfo is used in L4 the same way it is used in the L3 firewall configuration.

NOTE: Permissions are identical for read and write accesses in L4 interconnect targets.

Figure 5-14 shows an example of CONNID_BIT_VECTOR.

Figure 5-14. Example of CONNID_BIT_VECTOR



Setting bits 1 and 4 in the PROT_GROUP_ID_1 defines a group initiator able to access targets in protection group 1, and includes:

- MPU SS

Protection group 1 (PG1) can be applied to multiple protection regions without limitation. Each protection region l that is configured with PG1 enables permission access to these two initiators only.

The firewall default configuration for the L4-Core interconnect contains eight protection groups to ensure security:

- Most regions are, by default, set with protection group 7 (PG7), which is configured for all access (see Table 5-79).
- Protection group 0 (PG0) is restricted to the MPU subsystem in public.
- By default, the hardware crypto-processors (DES, SHA1-MD5, AES, FPKA, RNG, and WDTIMER1 not available in GP device) are attached to PG1 and are publicly accessible by all the initiators that can access the L4-Core.
- The LCD is attached to protection group 2 (PG2).
- The GPTIMER12 of the L4-Wakeup interconnect is attached to protection group 4 (PG4) (not available in GP device).
- By default, PG5 to PG7 are configured for all access.

NOTE: System control module and secure watchdog timer have an embedded firewall.

The L4-Per interconnect contains eight protection groups:

- By default, PG0 to PG7 are configured for all access.
- By default, most regions are set with PG7, which is configured for all access (see Table 5-80).

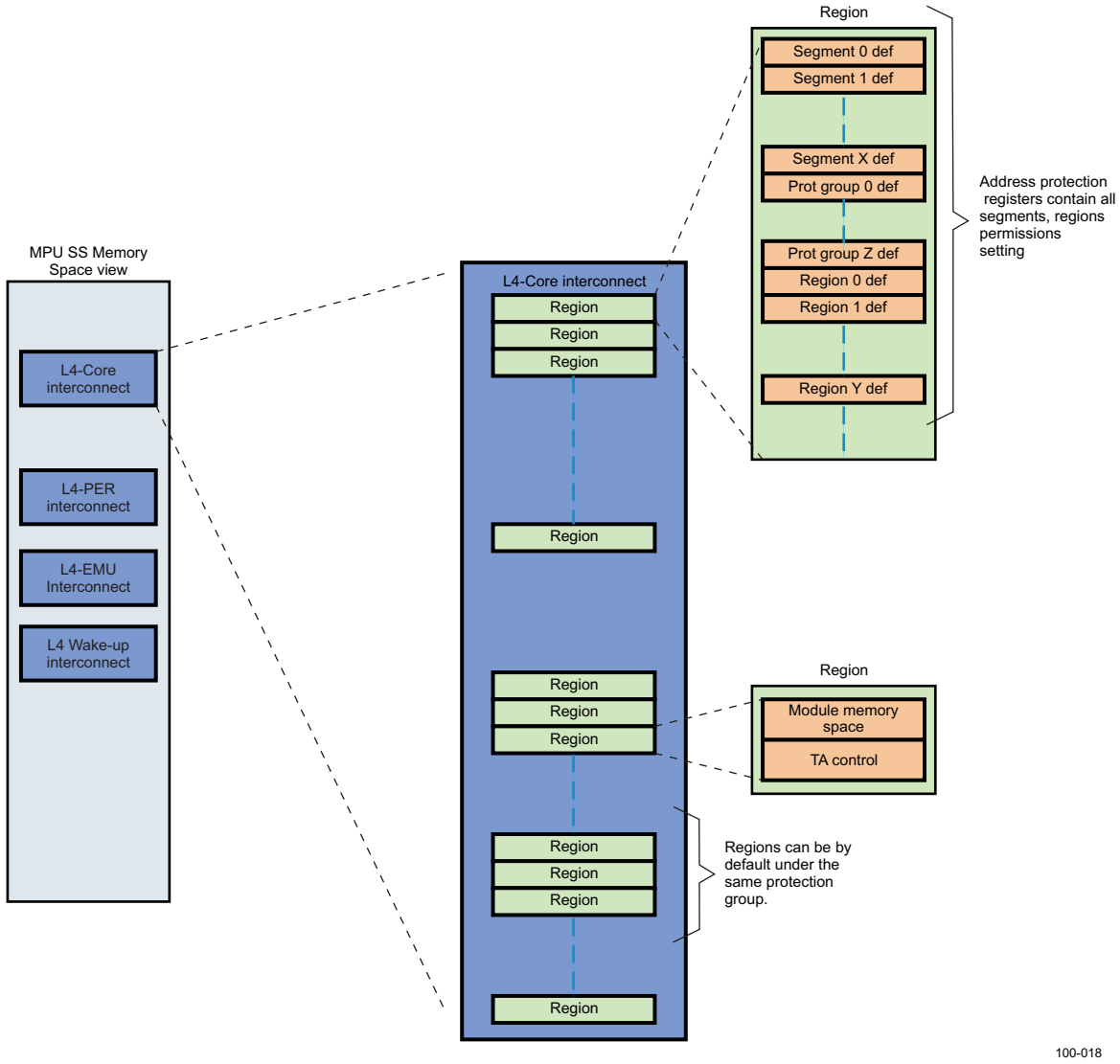
The L4-Emu interconnect contains six protection groups:

- By default, most regions are set with PG5, which is configured for all access (see Table 5-81).
- The AP is attached to PG0.
- The WDTIMER1 is attached to PG1 (not available in GP device).
- The GPTIMER12 is attached to PG2 (not available in GP device).
- The emulation organs are attached to PG3.

5.9.3.3 Segments and Regions

The protection mechanism for L4 interconnects is based on a hierarchical segmentation (see Figure 5-15). By default, some regions are attached to a specific protection group. This specificity allows the user to set up the permission access for certain types of modules that require the same access protection without managing the region allocation.

Figure 5-15. L4 Firewall Overview



All interconnect address spaces are covered by regions. Table 5-79to Table 5-81 list the module mapping, including the address, region number, and default protection group allocated to it.

By setting ENABLE bit L4_AP_REGION_y_H[0] to 0, the region becomes inactive and therefore inaccessible. Setting the bit to 1 activates the region.

Table 5-79. Region Allocation for L4-Core Interconnect

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
Reserved	0x4800 0000	Reserved		
System control module	0x4800 2000	Module	73	7
	0x4800 3000	L4 interconnect	75	7
Clock manager	0x4800 4000	Module region A	66	7
	0x4800 6000	Module region B	72	7
	0x4800 6800	Reserved		
	0x4800 7000	L4 interconnect	67	7
Reserved	0x4800 8000 0x4802 4000 0x4802 5000 0x4802 6000	Reserved		
L4-Core configuration	0x4804 0000	Address protection (AP)	0	7
	0x4804 0800	Initiator port (IP)	1	7
	0x4804 1000	Link agent (LA)	2	0
Reserved	0x4804 2000 0x4804 FC00	Reserved		
Display subsystem	0x4805 0400	Display subsystem top	4	2
	0x4805 0400	Display controller	4	2
	0x4805 0800	RFBI	5	2
	0x4805 0C00	Video encoder	6	2
	0x4805 1000	L4 interconnect	7	2
Reserved	0x4805 2000	Reserved		
sDMA	0x4805 6000	Module	9	7
	0x4805 7000	L4 interconnect	10	7
Reserved	0x4805 8000 0x4805 9000 0x4805 A000 0x4805 B000 0x4805 C000 0x4805 D000	Reserved		
I2C3	0x4806 0000	Module	73	7
	0x4806 1000	L4 interconnect	74	7
USBTLL	0x4806 2000	Module	100	7
	0x4806 3000	L4 interconnect	101	7
USBHS Host	0x4806 4000	Module	15	7
	0x4806 5000	L4 interconnect	16	7
UART1	0x4806 A000	Module	17	7
	0x4806 B000	L4 interconnect	18	7
UART2	0x4806 C000	Module	19	7
	0x4806 D000	L4 interconnect	20	7
Reserved	0x4806 E000	Reserved		
I2C1	0x4807 0000	Module	21	7
	0x4807 1000	L4 interconnect	22	7
I2C2	0x4807 2000	Module	23	7
	0x4807 3000	L4 interconnect	24	7
McBSP1	0x4807 4000	Module	25	7

Table 5-79. Region Allocation for L4-Core Interconnect (continued)

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
(digital base band data)	0x4807 5000	L4 interconnect	26	7
Reserved	0x4807 6000	Reserved		
GPTIMER10	0x4808 6000	Module	27	7
	0x4808 7000	L4 interconnect	28	7
GPTIMER11	0x4808 8000	Module	29	7
	0x4808 9000	L4 interconnect	30	7
Reserved	0x4808 A000 0x4808 B000 0x4808 C000 0x4809 4000 0x4809 5000	Reserved		
McBSP5	0x4809 6000	Module	59	7
(MIDI data)	0x4809 7000	L4 interconnect	60	7
SPI1	0x4809 8000	Module	35	7
	0x4809 9000	L4 interconnect	36	7
SPI2	0x4809 A000	Module	37	7
	0x4809 B000	L4 interconnect	38	7
MMC/SD/SDIO1	0x4809 C000	Module	39	7
	0x4809 D000	L4 interconnect	40	7
UART4	0x4809 E000	Module	43	7
	0x4809 F000	L4 interconnect	44	7
Reserved	0x480A 0000 0x480A 1000 0x480A 2000 0x480A 3000 0x480A 4000 0x480A 5000 0x480A 6000 0x480A 7000 0x480A 8000 0x480A A000 0x480A B000 0x480A C000	Reserved		
MMC/SD/SDIO3	0x480A D000	Module	98	7
	0x480A E000	L4 interconnect	99	7
MG	0x480B 0000	Module	55	7
	0x480B 1000	L4 interconnect	56	7
HDQ/1-Wire	0x480B 2000	Module	57	7
	0x480B 3000	L4 interconnect	58	7
MMC/SD/SDIO2	0x480B 4000	Module	41	7
	0x480B 5000	L4 interconnect	42	7
Reserved	0x480B 6000 0x480B 7000	Reserved		
SPI3	0x480B 8000	Module	66	7
	0x480B 9000	L4 interconnect	67	7

Table 5-79. Region Allocation for L4-Core Interconnect (continued)

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
SPI4	0x480B A000	Module	77	7
	0x480B B000	L4 interconnect	78	7
Reserved	0x480B C000	Reserved		
	0x480C 0000			
	0x480C 1000			
	0x480C 2000			
	0x480C 3000			
	0x480C 4000			
	0x480C 5000			
	0x480C 6000			
	0x480C 7000			
	0x480C 8000			
	0x480C D000			
	0x480C E000			
	0x480C F000			
MPU INTC	0x4820 0000	Nonshared device mapping		
Reserved	0x4820 1000	Reserved		
SSM	0x4828 0000	Nonshared device mapping		
Reserved	0x4828 1000	Reserved		
L4-Wakeup interconnect	0x4830 0000	L4 wakeup GPTIMER12[1]	76	7
	0x4830 6000	L4 wakeup	92	7
	0x4830 8000	L4 wakeup	93	7
	0x4830 9000	L4 wakeup	94	7
	0x4830 C000	L4 wakeup	95	7
	0x4831 0000	L4 wakeup	96	7
	0x4832 0000	L4 wakeup	97	7
	0x4834 0000	L4 wakeup	102	7
Reserved	0x4834 1000	Reserved		

(1) Not available in GP device.

Table 5-80. Region Allocation for L4-Per Interconnect

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
L4-Per configuration	0x4900 0000	Address protection (AP)	0	0
	0x4900 0800	Initiator port (IP)	1	7
	0x4900 1000	Link agent (LA)	2	7
Reserved	0x4900 2000	Reserved		
UART3	0x4902 0000	Module	3	7
(Infrared)	0x4902 1000	L4 interconnect	4	7
McBSP2	0x4902 2000	Module	5	7
(Audio for codec)	0x4902 3000	L4 interconnect	6	7
McBSP3	0x4902 4000	Module	7	7
(Bluetooth voice data)	0x4902 5000	L4 interconnect	8	7
McBSP4 (digital base band voice data)	0x4902 6000	Module	9	7
	0x4902 7000	L4 interconnect	10	7

Table 5-80. Region Allocation for L4-Per Interconnect (continued)

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
McBSP2 (Sidetone)	0x4902 8000	Module	39	7
	0x4902 9000	L4 interconnect	40	7
McBSP3 (Sidetone)	0x4902 A000	Module	41	7
	0x4902 B000	L4 interconnect	42	7
Reserved	0x4902 C000		Reserved	
WDTIMER3	0x4903 0000	Module	11	7
	0x4903 1000	L4 interconnect	12	7
GPTIMER2	0x4903 2000	Module	13	7
	0x4903 3000	L4 interconnect	14	7
GPTIMER3	0x4903 4000	Module	15	7
	0x4903 5000	L4 interconnect	16	7
GPTIMER4	0x4903 6000	Module	17	7
	0x4903 7000	L4 interconnect	18	7
GPTIMER5	0x4903 8000	Module	19	7
	0x4903 9000	L4 interconnect	20	7
GPTIMER6	0x4903 A000	Module	21	7
	0x4903 B000	L4 interconnect	22	7
GPTIMER7	0x4903 C000	Module	23	7
	0x4903 D000	L4 interconnect	24	7
GPTIMER8	0x4903 E000	Module	25	7
	0x4903 F000	L4 interconnect	26	7
GPTIMER9	0x4904 0000	Module	27	7
	0x4904 1000	L4 interconnect	28	7
Reserved	0x4904 2000		Reserved	
GPIO2	0x4905 0000	Module	29	7
	0x4905 1000	L4 interconnect	30	7
GPIO3	0x4905 2000	Module	31	7
	0x4905 3000	L4 interconnect	32	7
GPIO4	0x4905 4000	Module	33	7
	0x4905 5000	L4 interconnect	34	7
GPIO5	0x4905 6000	Module	35	7
	0x4905 7000	L4 interconnect	36	7
GPIO6	0x4905 8000	Module	37	7
	0x4905 9000	L4 interconnect	38	7
Reserved	0x4905 A000		Reserved	

Table 5-81. Region Allocation for L4-Emu Interconnect

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
Reserved	0x5400 0000		Reserved	
TEST–Chip-level TAP	0x5400 4000	Module	1	5
	0x5400 5000	L4 interconnect	2	5
L4-Emu configuration	0x5400 6000	Address protection (AP)	3	0
	0x5400 6800	Initiator port (IP) L4-Core	4	5
	0x5400 7000	Link agent (LA)	5	5
	0x5400 8000	Initiator port (IP) DAP	6	5

Table 5-81. Region Allocation for L4-Emu Interconnect (continued)

Device Name	Start Address (hex)	Description	Region Number	Default Protection Group
Reserved	0x5400 8800	Reserved		
MPU SS (emulation, trace, and debug)	0x5401 0000	Module	16	3
	0x5401 8400	L4 interconnect	7	3
TPIU	0x5401 9000	Module	8	3
	0x5401 A000	L4 interconnect	9	3
ETB	0x5401 B000	Module	10	3
	0x5401 C000	L4 interconnect	11	3
DAPCTL	0x5401 D000	Module	12	3
	0x5401 E000	L4 interconnect	13	3
SDTI	0x5401 F000	L4 interconnect	15	5
	0x5402 0000	Reserved		
	0x5450 0000	SDTI module (configuration)	14	5
	0x5451 0000	Reserved		
	0x5460 0000	SDTI module (window)	0	5
Reserved	0x5470 0000	Reserved		
GPTIMER12[1] (WKUP power domain)	0x5470 4000	Module	19	2
	0x5470 5000	L4 interconnect		
Power and reset manager • Power manager • Reset manager (WKUP power domain)	0x5470 6000	Module region A	17	5
	0x5470 8000	Module region B	18	5
	0x5470 8800	Reserved	20	5
	0x5470 9000	L4 interconnect	21	5
Reserved	0x5470 A000	Reserved		
WDTIMER1 (WKUP power domain)[1]	0x5470 C000	Module	22	1
	0x5470 D000	L4 interconnect	22	1
Reserved	0x5470 E000	Reserved		
GPIO1 (WKUP power domain)	0x5471 0000	Module	23	5
	0x5471 1000	L4 interconnect		
Reserved	0x5471 2000	Reserved		
WDTIMER2 (WKUP power domain)	0x5471 4000	Module		
	0x5471 5000	L4 interconnect		
Reserved	0x5471 6000	Reserved		
GPTIMER1 (WKUP power domain)	0x5471 8000	Module		
	0x5471 9000	L4 interconnect		
Reserved	0x5471 A000	Reserved		
32KTIMER (WKUP power domain)	0x5472 0000	Module	24	5
	0x5472 1000	L4 interconnect		
Reserved	0x5472 2000	Reserved		
L4-Wakeup configuration (WKUP power domain)	0x5472 8000	Address protection (AP)		
	0x5472 8800	Initiator port (IP) L4-Core		
	0x5472 9000	Link agent (LA)		
	0x5472 A000	Initiator port (IP) L4-Emu		
Reserved	0x5472 A800	Reserved		
L4-Wakeup	0x5473 0000	L4 interconnect	25	5

(1) Not available in GP device.

5.9.3.4 L4 Firewall Address and Protection Registers Setting

Table 5-82 lists the settings of the AP registers for an L4 interconnect firewall. These values are computed based on the physical implementation of each L4 interconnect.

Table 5-82. L4 Firewall Register Description Overview

Register Type	Register Name	Bits	Field	Description
Segment	L4_AP_SEGMENT_i_L	23:0	Base	Segment base address
	L4_AP_SEGMENT_i_H	4:0	SIZE	Segment size equals to 2 power of SIZE
Protection groups	L4_AP_PROT_GROUP_MEMBERS_k_L	15:0	CONNID_BIT_VECTOR	See Section 5.9.1 for L4ConnID).
	L4_AP_PROT_GROUP_ROLES_k_L	15:0	ENABLE	Refer to Table 5-21 for MReq description
Region setting	L4_AP_REGION_I_L	23:0	BASE	Define the base address of region in respect to its respective segment base address
	L4_AP_REGION_I_H	27:24	SEGMENT_ID	Segment ID number of the region
		22:20	PROT_GROUP_ID	The protection group attached to the region
		19:17	BYTE_DATA_WIDTH_EXP	Determine the number of bytes in an access
		5:1	SIZE	Size of the region equals to 2 power of SIZE
		0	ENABLE	Enable the region protection

5.9.4 Error Handling

5.9.4.1 Overview

The L4 interconnect provides mechanisms for handling either internally detected errors or errors reported by modules attached to the L4 target ports. Hardware support facilitates logging errors and cleaning up the state to allow error recovery software to treat the error.

As an L3 target, the L4 interconnect reports errors to the L3 interconnect in-band whenever possible. In-band error reporting is the default and recommended configuration. It is assumed that INBAND_ERROR_REP bit L4_IA_AGENT_CONTROL_L[27] is set to 1.

NOTE: *L4_IA* denotes which interconnect is considered: L4-Core, L4-Per, L4-Emu, or L4-Wakeup.

L4_TA denotes the module name, such as UART1, McBSP1, etc.

The L4 interconnects handle three types of errors:

- No target core found or address hole
- Request protection violation
- Failure of the target to service a request before a time-out expires

5.9.4.2 Error Logging

5.9.4.2.1 No Target Core Found/Address Hole

This error indicates that a request was addressed to a hole in the L4 address map.

When this occurs, an in-band error response is returned to the L3 level.

The error is also logged into the INBAND_ERROR bit L4_IA_AGENT_STATUS_L[27].

Additionally, an address hole error code is logged into the CODE field L4_IA_ERROR_LOG_L[25:24].

The L4_IA_ERROR_LOG register also includes MULTI bit L4_IA_ERROR_LOG_L[31], which is asserted when multiple errors are detected. In this case, the error code corresponds to the first error that occurs.

5.9.4.2.2 Protection Violation

This error indicates that an initiator has accessed a restricted region. This error is reported using an in-band error. It is written to the INBAND_ERROR field. A protection violation error code is also saved in the CODE field L4_IA_ERROR_LOG_L[25:24].

The security violation is also logged in the CONTROL_SEC_ERR_STATUS [7] register of the system control module.

NOTE: There is no specific error signal for the security violation that goes out from the L4 interconnects. The out-band error that goes to the system control module indicates a SECURITY_VIOLATION, but it also indicates normal interconnect errors.

5.9.4.2.3 Time-Out

This section gives information about all modules and features in the high-tier device. See the *Device Family* chapter to check availability of modules and features. To flag interconnect response being blocked, the time-out of target agents attached to unavailable modules can be enabled with the lowest setting.

A time-out mechanism can be enabled on a per-target basis in the L4_TA_AGENT_CONTROL_L register. If the mechanism is enabled for a TA, and commands are not accepted or responses are not returned within the expected delay, the L4 interconnect generates an error event.

The error is logged in the target agent REQ_TIMEOUT bit L4_TA_AGENT_STATUS_L[8]. The affected TA enters an error state that causes it to send error responses to any new request targeted at it. To recover from this state, the TA must be reset by system software. The time-out is counted starting from the moment a command is presented to the target, regardless of how the target responds to the command.

The L4 interconnect implements a centralized time-base circuit that broadcasts a set of four periodic pulse signals to all connected TAs. These four signals are referred to as 1X-time base, 4X-time base, 16X-time base, and 64X-time base.

The time-base circuit offers four possible sets of time-base signals. Selection is done by programming TIMEOUT_BASE field L4_LA_NETWORK_CONTROL_L[10:8]. Table 5-83 lists the values in the number of L4 clock cycles.

Table 5-83. L4 Time-Out Link and TA Programming

	REQ_TIMEOUT[2:0]				
TIMEOUT_BASE[2:0]	**0**	**1**	**2**	**3**	**4**
0	All L4 time-out features are disabled.				
1	Locally disabled	64	256	1024	4096
2		256	1024	4096	16384
3		1024	4096	16384	65536
4		4096	16384	65536	262144

The reset value is 0x4, resulting in the longest possible time-out.

The selected time-base signals are available at any TA. Each TA can be programmed to refer to one of these four time-base signals, by using REQ_TIMEOUT field L4_TA_AGENT_CONTROL_L[10:8] (see Table 5-84).

Table 5-84. L4 Time-Out TA Programming

REQ_TIMEOUT	Target Agent Time-out Reference
0	Time-out locally disabled
1	1X-base cycle
2	4X-base cycle
3	16X-base cycle
4	64X-base cycle

A time-out condition is detected when the command acceptance or the response is not received after a delay of between one and three time-base periods.

Example:

- L4 frequency = 100 MHz
- TIMEOUT_BASE = 4 in the L4_LA_NETWORK_CONTROL_L register
- REQ_TIMEOUT = 2 in the L4_TA_AGENT_CONTROL_L for target agent A
- REQ_TIMEOUT = 4 in the L4_TA_AGENT_CONTROL_L for target agent B

At agent A, the time-base unit is 16384 cycles. A time-out is issued when a request to the attached module is not accepted or no response is sent after a delay of 164 µs to 492 µs.

At agent B, the time-base unit is 262,144 cycles. A time-out is issued when a request to the attached module is not accepted or no response is sent after a delay of 2.6 ms to 7.8 ms.

On detection of a time-out condition, the agent automatically generates an error response to the inter_IA, which is forwarded to the L3. The agent also logs the time-out to REQ_TIMEOUT bit L4_TA_AGENT_STATUS_L[8].

After the time-out has been detected and logged, the behavior of the attached module is ignored. A new request targeting the module arriving at the timed out TA receives an error response. If the request is addressed to the agent internal registers, it is processed normally.

To recover from a time-out error, the software is assumed to reset first the faulty module by using its internal soft reset bit, and then the TA by using OCP_RESET bit L4_TA_AGENT_CONTROL_L[0].

5.9.4.3 TA Software Reset

Writing 1 to OCP_RESET bit L4_TA_AGENT_CONTROL_L[0] initiates the software reset period. The software reset must be asserted for at least 16 cycles of the target module OCP clock, which can be a divided clock with respect to the L4 clock.

During the software reset period the following occur:

- Requests sent to the target module receive error responses. Therefore, if the faulty request is part of a DMA transfer, it is necessary to stop the DMA to avoid getting unwanted errors.
- Requests sent to the TA register block are processed as usual.
- The L4_TA_AGENT_STATUS_L[8] REQ_TIMEOUT status bit is cleared.

Writing 0 to the L4_TA_AGENT_CONTROL_L[0] OCP_RESET bit terminates the software reset period.

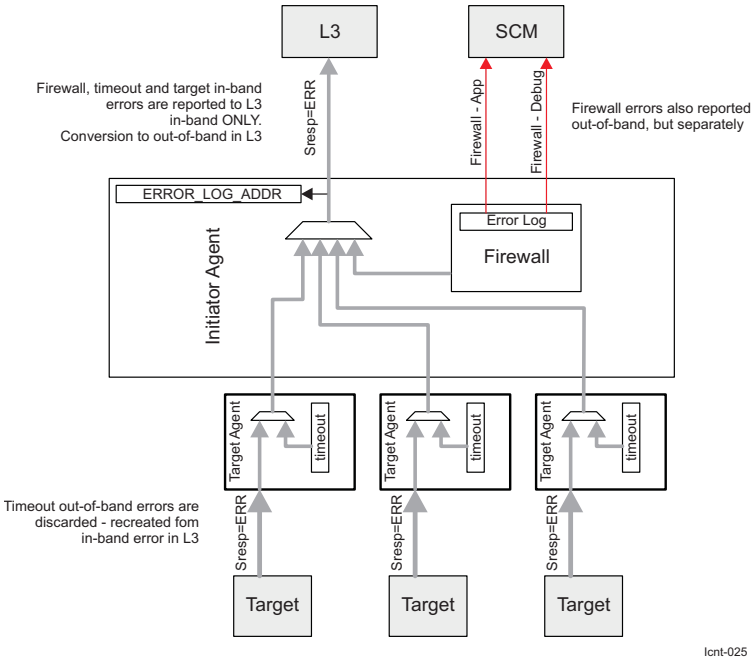
The attached module must then be reset to complete the recovery.

5.9.4.4 Error Reporting

Figure 5-16 illustrates the error-reporting scheme used in the L4_Core, L4_Per, and L4_EMU interconnects.

- Timeout error generation is enabled at each TA.
- Timeout error and target in-band errors are only reported in-band to the L3 initiator.
- Protection violations are reported out-of-band. Error steering is used to distinguish between application or debug errors.

Figure 5-16. L4 Error Reporting



5.10 L4 Interconnects Registers

A summary of the hardware interface for the L4-Core, L4-Per, L4-Emu and L4-Wkup interconnects is given in Table 5-85 to Table 5-88. Each module instance in the design is shown with the module register map and bit definitions for each bit field.

Table 5-85. L4- Core Instance Summary

Module Name	Base Address	Size
CORE_TA_CONTROL	0x4800 3000	4K bytes
CORE_TA_CM	0x4800 7000	4K bytes
CORE_AP	0x4804 0000	2K bytes
CORE_IA	0x4804 0800	2K bytes
CORE_LA	0x4804 1000	4K bytes
CORE_TA_DISPLAY_SS	0x4805 1000	4K bytes
CORE_TA_SDMA	0x4805 7000	4K bytes
CORE_TA_I2C3	0x4806 1000	4K bytes
CORE_TA_USB_HS_TLL	0x4806 3000	4K bytes
CORE_TA_USB_HS_Host	0x4806 5000	4K bytes
CORE_TA_UART1	0x4806 B000	4K bytes
CORE_TA_UART2	0x4806 D000	4K bytes
CORE_TA_I2C1	0x4807 1000	4K bytes
CORE_TA_I2C2	0x4807 3000	4K bytes
CORE_TA_MCBSP1	0x4807 5000	4K bytes
CORE_TA_GPTIMER10	0x4808 7000	4K bytes
CORE_TA_GPTIMER11	0x4808 9000	4K bytes
Reserved	0x4809 5000	4K bytes
CORE_TA_MCBSP5	0x4809 7000	4K bytes
CORE_TA_SPI1	0x4809 9000	4K bytes
CORE_TA_SPI2	0x4809 B000	4K bytes
CORE_TA_MMCHS1	0x4809 D000	4K bytes
CORE_TA_UART4	0x4809 F000	4K bytes
Reserved	0x480A 1000	4K bytes
Reserved	0x480A 3000	4K bytes
Reserved	0x480A 5000	4K bytes
Reserved	0x480A 7000	4K bytes
Reserved	0x480A A000	4K bytes
Reserved	0x480A C000	4K bytes
CORE_TA_MMCHS3	0x480A E000	4K bytes
CORE_TA_MG	0x480B 1000	4K bytes
CORE_TA_HD1W	0x480B 3000	4K bytes
CORE_TA_MMCHS2	0x480B 5000	4K bytes
Reserved	0x480B 7000	4K bytes
CORE_TA_SPI3	0x480B 9000	4K bytes
CORE_TA_SPI4	0x480B B000	4K bytes
Reserved	0x480C 0000	4K bytes
Reserved	0x480C 2000	4K bytes
Reserved	0x480C 4000	4K bytes
Reserved	0x480C 6000	4K bytes
CORE_TA_INTH	0x480C 8000	4K bytes
Reserved	0x480C E000	4K bytes
CORE_TA_WKUP	0x4834 0000	4K bytes

Table 5-86. L4-Per Instance Summary

Module Name	Base Address	Size
PER_AP	0x4900 0000	2K bytes
PER_IA	0x4900 0800	2K bytes
PER_LA	0x4900 1000	4K bytes
PER_TA_UART3	0x4902 1000	512 bytes
PER_TA_MCBSP2	0x4902 3000	1K byte
PER_TA_MCBSP3	0x4902 5000	1K byte
PER_TA_MCBSP4	0x4902 7000	1K byte
PER_TA_MCBSP_SIDETONE2	0x4902 9000	4K bytes
PER_TA_MCBSP_SIDETONE3	0x4902 B000	4K bytes
PER_TA_WDtimer3	0x4903 1000	2K bytes
PER_TA_GPTIMER2	0x4903 3000	1K byte
PER_TA_GPTIMER3	0x4903 5000	1K byte
PER_TA_GPTIMER4	0x4903 7000	1K byte
PER_TA_GPTIMER5	0x4903 9000	1K byte
PER_TA_GPTIMER6	0x4903 B000	1K byte
PER_TA_GPTIMER7	0x4903 D000	1K byte
PER_TA_GPTIMER8	0x4903 F000	1K byte
PER_TA_GPTIMER9	0x4904 1000	1K byte
PER_TA_GPIO2	0x4905 1000	1K byte
PER_TA_GPIO3	0x4905 3000	1K byte
PER_TA_GPIO4	0x4905 5000	1K byte
PER_TA_GPIO5	0x4905 7000	1K byte
PER_TA_GPIO6	0x4905 9000	1K byte

Table 5-87. L4-Emu Instance Summary

Module Name	Base Address	Size
EMU_TEST_TAP	0x5400 5000	4K bytes
EMU_AP	0x5400 6000	2K bytes
EMU_IA_0	0x5400 6800	2K bytes
EMU_LA	0x5400 7000	4K bytes
EMU_IA_1	0x5400 8000	2K bytes
EMU_TA_MPU	0x5401 8000	4K bytes
EMU_TA_TPIU	0x5401 A000	4K bytes
EMU_TA_ETB	0x5401 C000	4K bytes
EMU_TA_DAPCTL	0x5401 E000	4K bytes
EMU_TA_SDTI	0x5401 F000	4K bytes
EMU_TA_L4WKUP	0x5473 0000	4K bytes

Table 5-88. L4-WKUP Instance Summary

Module Name	Base Address	Size
WKUP_TA_GPTIMER12	0x4830 5000	4K bytes
WKUP_TA_PRM	0x4830 9000	4K bytes
WKUP_TA_WDTIMER1	0x4830 D000	4K bytes
WKUP_TA_GPIO1	0x4831 1000	4K bytes
WKUP_TA_WDTIMER2	0x4831 5000	4K bytes
WKUP_TA_GPTIMER1	0x4831 9000	4K bytes

Table 5-88. L4-WKUP Instance Summary (continued)

Module Name	Base Address	Size
WKUP_TA_SYNCTIMER32K	0x4832 1000	4K bytes
WKUP_AP	0x4832 8000	2K bytes
WKUP_IA_L4CORE	0x4832 8800	2K bytes
WKUP_LA	0x4832 9000	4K bytes
WKUP_IA_L4EMU	0x4832 A000	2K bytes

5.10.1 L4 Initiator Agent (L4 IA) Register Mapping Summary

This section provides information on the L4 IA module. Each of the registers within the module instance is described separately in Table 5-89 to Table 5-90.

The initiator OCP interface register block (IA) consists of the status, control, and error log registers that can be used to configure the interface of an initiator OCP. There can be only one register block for each initiator OCP interface.

Table 5-89. L4 IA Register Mapping Summary (1)

Register Name	Type	Register Width (Bits)	CORE_IA Physical Address	PER_IA Physical Address	EMU_IA_ L3 Physical Address	Section
L4_IA_AGENT_CONTROL_L	RW	32	0x4804 0820	0x4900 0820	0x5400 6820	Section 5.10.2.1
L4_IA_AGENT_STATUS_L	RW	32	0x4804 0828	0x4900 0828	0x5400 6828	Section 5.10.2.2
L4_IA_ERROR_LOG_L	RW	32	0x4804 0858	0x4900 0858	0x5400 6858	Section 5.10.2.3

Table 5-90. L4 IA Register Mapping Summary (2)

Register Name	Type	Register Width (Bits)	EMU_IA_DAP Physical Address	WKUP_IA_EMU Physical Address	WKUP_IA_CORE Physical Address	Section
L4_IA_AGENT_CONTROL_L	RW	32	0x5400 8020	0x4832 8820	0x4832 A020	Section 5.10.2.1
L4_IA_AGENT_STATUS_L	RW	32	0x5400 8028	0x4832 8828	0x4832 A028	Section 5.10.2.2
L4_IA_ERROR_LOG_L	RW	32	0x5400 8058	0x4832 8858	0x4832 A058	Section 5.10.2.3

5.10.2 L4 Initiator Agent (L4 IA) Register Descriptions

5.10.2.1 L4_IA_AGENT_CONTROL_L

Table 5-91. L4_IA_AGENT_CONTROL_L

Address Offset	0x020
Physical address	Please refer fromTable 5-89 to Table 5-90
Description	Enable error reporting on an initiator interface. The error reporting mechanism is enabled when the INBAND_ERROR_REP bit field is set to 1. The out-of-band OCP MError reporting mechanism is enabled when the MERROR_REP bit field is set to 1.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				INBAND_ERROR_REP	Reserved		MERROR_REP	Reserved																							

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Read returns 0.	R	0x0
27	INBAND_ERROR_REP	Setting this field to 1 reports on in-band errors using the INBAND_ERROR log bit of IA.AGENT_STATUS register.	R	1
26:25	Reserved	Read returns 0.	R	0x0
24	MERROR_REP	Enable MError reporting	R	0x0
23:0	Reserved	Read returns 0.	R	0x0000000

5.10.2.2 L4_IA_AGENT_STATUS_L

Table 5-92. L4_IA_AGENT_STATUS_L

Address Offset	0x028
Physical address	Please refer fromTable 5-89 to Table 5-90
Description	Stores status information for an initiator. The INBAND_ERROR and MERROR fields are read/write and are implemented as log bits.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				INBAND_ERROR_REP	Reserved		MERROR_REP	Reserved																							

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Read returns 0.	R	0x0
27	INBAND_ERROR_REP	0x0 No In-Band error present. 0x1 In-Band error present.	R 1toClr	0x0
26:25	Reserved	Read returns 0.	R	0x0
24	MERROR_REP	0x0 No MError error present. 0x1 MErorr error present.	R	0x0
23:0	Reserved	Read returns 0.	R	0x0000000

5.10.2.3 L4_IA_ERROR_LOG_L

Table 5-93. L4_IA_ERROR_LOG_L

Address Offset	0x058
Physical address	Please refer fromTable 5-89 to Table 5-90
Description	Log information about error conditions. The CODE field logs any protection violation or address hole errors detected by the initiator subsystem while decoding a request.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MULTI	Reserved						CODE	Reserved																							

Bits	Field Name	Description	Type	Reset
31	MULTI	While the CODE field is not zero, the MULTI bit is asserted whenever an additional error is detected. Once set by hardware, the MULTI bit can only be cleared by writing a 1 to it while writing a value other than zero to the CODE field.	RW	0
30:26	Reserved	Read returns 0.	R	0x00

Bits	Field Name	Description	Type	Reset
25:24	CODE	The error code of an initiator request. 0x00: No errors 0x01: Reserved 0x10: Address hole 0x11: Protection violation The CODE field, once set by hardware, can only be cleared by writing a non-zero value to it, in conjunction with writing a 1 to the MULTI bit field.	RW	0x0
23:0	Reserved	Read returns 0.	R	0x000000

5.10.3 L4 Target Agent (L4 TA) Register Mapping Summary

This section provides information on the L4 Target agent (TA) register module. Each of the registers within the module instance is described separately in Table 5-94 to Table 5-122.

Table 5-94 to Table 5-123 provide information on the L4 Target Agent having two supplementary registers OCP_CONTROL and OCP_STATUS

The TA register block consists of the status and control registers that can be used to configure the interfaces of the targets.

Table 5-94. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_CONTROL Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4800 3020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4800 3024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4800 3028	Section 5.10.4.3

Table 5-95. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_CM Physical Address	CORE_TA_ DISPLAY_SS Physical Address	CORE_ TA_SDMA Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4802 7020	0x4805 1020	0x4805 7020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4802 7024	0x4805 1024	0x4805 7024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4802 7028	0x4805 1028	0x4805 7028	Section 5.10.4.3

Table 5-96. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_I2C3 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4806 1020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4806 1024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4806 1028	Section 5.10.4.3

Table 5-97. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_ USB_TLL Physical Address	CORE_TA_ USB_HS_Host Physical Address	CORE_ TA_UART1 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4806 3020	0x4806 5020	0x4806 B020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4806 3024	0x4806 5024	0x4806 B024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4806 3028	0x4806 5028	0x4806 B028	Section 5.10.4.3

Table 5-98. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_ USB_TLL Physical Address	CORE_TA_ USB_HS_HOST Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4806 3020	0x4806 5020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4806 3024	0x4806 5024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4806 3028	0x4806 5028	Section 5.10.4.3

Table 5-99. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ TA_UART2 Physical Address	CORE_ TA_I2C1 Physical Address	CORE_ TA_I2C12 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4806 D020	0x4807 1020	0x4807 1020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4806 D024	0x4807 1024	0x4807 1024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4806 D028	0x4807 1028	0x4807 1028	Section 5.10.4.3

Table 5-100. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ TA_MCBSP1 Physical Address	CORE_TA_ GPTIMER10 Physical Address	CORE_TA_ GPTIMER11 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4807 5020	0x4808 7020	0x4809 9020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4807 5024	0x4808 7024	0x4809 9024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4807 5028	0x4808 7028	0x4809 9028	Section 5.10.4.3

Table 5-101. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ TA_MCBSP5 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4809 9020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4809 9024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4809 9028	Section 5.10.4.3

Table 5-102. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ TA_MCSPI1 Physical Address	CORE_ TA_MCSPI2 Physical Address	CORE_ TA_MMC1 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4809 9020	0x4809 B020	0x4809 D020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4809 9024	0x4809 B024	0x4809 D024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4809 9028	0x4809 B028	0x4809 D028	Section 5.10.4.3

Table 5-103. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ CORE_TA_UART4 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4809 F020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4809 F024	Section 5.10.4.2

Table 5-103. CORE_TA Common Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	CORE_ CORE_TA_UART4 Physical Address	Section
L4_TA_AGENT_STATUS_L	RW	32	0x4809 F028	Section 5.10.4.3

Table 5-104. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Section
L4_TA_AGENT_CONTROL_L	RW	32	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	Section 5.10.4.3

Table 5-105. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Section
L4_TA_AGENT_CONTROL_L	RW	32	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	Section 5.10.4.3

Table 5-106. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ TA_MMC3 Physical Address	TA_MG Physical Address	CORE_ TA_HDQ1 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x480A E020	0x480B 1020	0x480B 3020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x480A E024	0x480B 1024	0x480B 3024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x480A E028	0x480B 1028	0x480B 3028	Section 5.10.4.3

Table 5-107. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_ TA_MMC2 Physical Address	CORE_ TA_MCSPI3 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x480B 5020	0x480B 9020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x480B 5024	0x480B 9024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x480B 5028	0x480B 9028	Section 5.10.4.3

Table 5-108. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_ MCSPI4 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x480B B020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x480B B024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x480B B028	Section 5.10.4.3

Table 5-109. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_ INTH Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x480C 8020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x480C 8024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x480C 8028	Section 5.10.4.3

Table 5-110. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	Section
L4_TA_AGENT_CONTROL_L	RW	32	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	Section 5.10.4.3

Table 5-111. CORE_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_TA_WKUP Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4834 0020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4834 0024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4834 0028	Section 5.10.4.3

Table 5-112. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ UART3 Physical Address	PER_TA_ MCBSP2 Physical Address	PER_TA_ MCBSP3 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4902 1020	0x4902 3020	0x4902 5020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4902 1024	0x4902 3024	0x4902 5024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4902 1028	0x4902 3028	0x4902 5028	Section 5.10.4.3

Table 5-113. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ MCBSP4 Physical Address	PER_TA_ MCBSP2_ SIDETONE2 Physical Address	PER_TA_ MCBSP2_ SIDETONE3 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4902 7020	0x4902 9020	0x4902 B020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4902 7024	0x4902 9024	0x4902 B024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4902 7028	0x4902 9028	0x4902 B028	Section 5.10.4.3

Table 5-114. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ WDTIMER3 Physical Address	PER_TA_ GPTIMER2 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4903 1020	0x4903 3020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4903 1024	0x4903 3024	Section 5.10.4.2

Table 5-114. PER_TA Common Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	PER_TA_ WDTIMER3 Physical Address	PER_TA_ GPTIMER2 Physical Address	Section
L4_TA_AGENT_STATUS_L	RW	32	0x4903 1028	0x4903 3028	Section 5.10.4.3

Table 5-115. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ GPTIMER3 Physical Address	PER_TA_ GPTIMER4 Physical Address	PER_TA_ GPTIMER5 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4903 5020	0x4903 7020	0x4903 9020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4903 5024	0x4903 7024	0x4903 9024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4903 5028	0x4903 7028	0x4903 9028	Section 5.10.4.3

Table 5-116. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ GPTIMER6 Physical Address	PER_TA_ GPTIMER7 Physical Address	PER_TA_ GPTIMER8 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4903 B020	0x4903 D020	0x4903 F020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4903 B024	0x4903 D024	0x4903 F024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4903 B028	0x4903 D028	0x4903 F028	Section 5.10.4.3

Table 5-117. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ GPTIMER9 Physical Address	PER_TA_ GPIO2 Physical Address	PER_TA_ GPIO3 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4904 1020	0x4905 1020	0x4905 3020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4904 1024	0x4905 1024	0x4905 3024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4904 1028	0x4905 1028	0x4905 3028	Section 5.10.4.3

Table 5-118. PER_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	PER_TA_ GPIO4 Physical Address	PER_TA_ GPIO5 Physical Address	PER_TA_ GPIO6 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4905 5020	0x4905 7020	0x4905 9020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4905 5024	0x4905 7024	0x4905 9024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4905 5028	0x4905 7028	0x4905 9028	Section 5.10.4.3

Table 5-119. EMU_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	EMU_TA_ TESTCHIPLEVELTAP Physical Address	EMU_TA_ MPU Physical Address	EMU_TA_ TPUI Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x5400 5020	0x5401 8020	0x5401 A020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x5400 5024	0x5401 8024	0x5401 A024	Section 5.10.4.2

Table 5-119. EMU_TA Common Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	EMU_TA_ TESTCHIPLEVELTAP Physical Address	EMU_TA_ MPU Physical Address	EMU_TA_ TPUI Physical Address	Section
L4_TA_AGENT_STATUS_L	RW	32	0x5400 5028	0x5401 8028	0x5401 A028	Section 5.10.4.3

Table 5-120. EMU_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	EMU_TA_ ETB Physical Address	EMU_TA_ DAPCTL Physical Address	EMU_TA_ SDTI Physical Address	EMU_TA_ L4WKUP Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x5401 C020	0x5401 E020	0x5401 F020	0x5473 0020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x5401 C024	0x5401 E024	0x5401 F024	0x5473 0024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x5401 C028	0x5401 E028	0x5401 F028	0x5473 0028	Section 5.10.4.3

Table 5-121. WKUP_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	WKUP_TA_ GPTIMER12 Physical Address	WKUP_TA_ PRM Physical Address	WKUP_TA_ WDTIMER1 Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4830 5020	0x4830 9020	0x4830 D020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4830 5024	0x4830 9024	0x4830 D024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4830 5028	0x4830 9028	0x4830 D028	Section 5.10.4.3

Table 5-122. WKUP_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	WKUP_TA_ GPIO1 Physical Address	WKUP_TA_ WDTIMER Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4831 1020	0x4831 5020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4831 1024	0x4831 5024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x483 1028	0x4831 5028	Section 5.10.4.3

Table 5-123. WKUP_TA Common Register Mapping Summary

Register Name	Type	Register Width (Bits)	WKUP_TA_ GPTIMER1 Physical Address	WKUP_TA_ SYNCTIMER32K Physical Address	Section
L4_TA_AGENT_CONTROL_L	RW	32	0x4831 9020	0x4832 1020	Section 5.10.4.1
L4_TA_AGENT_CONTROL_H	RW	32	0x4831 9024	0x4832 1024	Section 5.10.4.2
L4_TA_AGENT_STATUS_L	RW	32	0x4831 9028	0x4832 1028	Section 5.10.4.3

5.10.4 L4 Target Agent (L4 TA) Register Descriptions

5.10.4.1 L4_TA_AGENT_CONTROL_L

Table 5-124. L4_TA_AGENT_CONTROL_L

Address Offset	0x020
Physical address	Please refer fromTable 5-94 to Table 5-123
Description	Enable error reporting
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SERROR_REP	Reserved													REQ_TIMEOUT			Reserved							OCP_RESET

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Read returns 0.	R	0x00
24	SERROR_REP	Enable logging of error	R	0x0
23:11	Reserved	Read returns 0.		
10:8	REQ_TIMEOUT	Timeout Bound. Values are:0 - No timeout 1 - 1x base cycles 2 - 4x base cycles 3 - 16x base cycles 4 - 64x base cycles	RW	0x2
7:1	Reserved	Read returns 0.	R	0x00
0	OCP_RESET	The OCP_RESET field controls the OCP reset signal to the attached core. Setting this bit clears any pending transfers and resets the OCP interface. The bit must be cleared to de-assert the OCP reset signal. When the software reset feature is available on a target agent, the target agent OCP must also have a reset signal directed to the target core.	RW	0

5.10.4.2 L4_TA_AGENT_CONTROL_H

Table 5-125. L4_TA_AGENT_CONTROL_H

Address Offset	0x024
Physical address	Please refer fromTable 5-94 to Table 5-123
Description	Enable clock power management
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							EXT_CLOCK	Reserved							

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Read returns 0.	R	0x000000
8	EXT_CLOCK	When set to 1, the ext_clk_off_i signal on a target agent indicates when the target agent should shut off.	R	0
7:0	Reserved	Read returns 0.	R	0x00

5.10.4.3 L4_TA_AGENT_STATUS_L

Table 5-126. L4_TA_AGENT_STATUS_L

Address Offset	0x028
Physical address	See Table 5-94 to Table 5-123
Description	Error reporting
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved															REQ_TIMEOUT	Reserved							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:9	Reserved	Read returns 0.	R	0x0001
8	REQ_TIMEOUT	0x0: No request timeout 0x1: A request timeout has occurred	R 1toCLR	0
7:0	Reserved	Read returns 0.	R	0x00

5.10.5 L4 Link Register Agent (LA) Register Mapping Summary

This section provides information on the L4-Core link agent (LA) register module. Each of the registers within the module instance is described separately in Table 5-127.

The LA register block contains the initiator subsystem information register and the composite sideband signal mask and status registers.

Table 5-127. L4 LA Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_LA Physical Address	PER_LA Physical Address	EMU_LA Physical Address	WKUP_LA Physical Address	
L4_LA_NETWORK_H	R	32	0x4804 1014	0x4900 1014	0x5400 7014	0x4832 9014	Section 5.10.6.1
L4_LA_INITIATOR_INFO_L	R	32	0x4804 1018	0x4900 1018	0x5400 7018	0x4832 9018	Section 5.10.6.2
L4_LA_INITIATOR_INFO_H	R	32	0x4804 101C	0x4900 101C	0x5400 701C	0x4832 901C	Section 5.10.6.3
L4_LA_NETWORK_CONTROL_L	RW	32	0x4804 1020	0x4900 1020	0x5400 7020	0x4832 9020	Section 5.10.6.4
L4_LA_NETWORK_CONTROL_H	RW	32	0x4804 1024	0x4900 1024	0x5400 7024	0x4832 9024	Section 5.10.6.5

5.10.6 L4 Link Register Agent (LA) Register Descriptions

5.10.6.1 L4_LA_NETWORK_H

Table 5-128. L4_LA_NETWORK_H

Address Offset	0x014
Physical address	Please refer to Table 5-127
Description	Identify the interconnect
Type	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															

Bits	Field Name	Description	Type	Reset
31:0	ID	The ID field uniquely identifies this interconnect, and can serve as a chip ID.	R	0x00010000

5.10.6.2 L4_LA_INITIATOR_INFO_L

Table 5-129. L4_LA_INITIATOR_INFO_L

Address Offset	0x018
Physical address	Please refer to Table 5-127
Description	Contain initiator subsystem information.
Type	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				PROT_GROUPS				NUMBER_REGIONS								Reserved												SEGMENTS			

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Read returns 0.	R	0x0
27:24	PROT_GROUPS	The number of protection groups. The PROT_GROUPS field contains read-only configuration information for the address mapping and security structure of the initiator subsystem. If the PROT_GROUPS field is set to 0, there are no protection group registers.	R	see Table 5-130
23:16	NUMBER_REGIONS	The number of regions. The NUMBER_REGIONS field contains read-only configuration information for the region register of the initiator subsystem.	R	see Table 5-130
15:4	Reserved	Read returns 0.	R	0x000
3:0	SEGMENTS	The number of segments. The SEGMENT fields contains read-only configuration information for the segment register of the initiator subsystem.	R	see Table 5-130

Table 5-130. Reset value for L4_LA_INITIATOR_INFO_L

Field Name	CORE_LA	PER_LA	EMU_LA	WKUP_LA
PROT_GROUPS	0x8	0x8	0x8	0x8
NUMBER_REGIONS	0x64	0x2B	0x1A	0x13
SEGMENTS	0x6	0x5	0x3	0x2

5.10.6.3 L4_LA_INITIATOR_INFO_H

Table 5-131. L4_LA_INITIATOR_INFO_H

Address Offset	0x01C
Physical address	Please refer to Table 5-127
Description	Contain initiator subsystem information.
Type	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													THREADS			Reserved	CONNID_WIDTH			Reserved	BYTE_DATA_WIDTH_EXP			Reserved			ADDR_WIDTH				

Bits	Field Name	Description	Type	Reset
31:19	Reserved	Read returns 0.	R	0x0000
18:16	THREADS	The THREADS field specifies the number of initiator threads connected to the interconnect. The field contains read-only configuration information for the initiator subsystem.	R	see Table 5-132
15	Reserved	Read returns 0.	R	0
14:12	CONNID_WIDTH	The initiator subsystem connID width. The CONNID_WIDTH field contains read-only configuration information for the initiator subsystem.	R	see Table 5-132
11	Reserved	Read returns 0.	R	0
10:8	BYTE_DATA_WIDTH_EXP	This field specifies the initiator subsystem data width. 1:2^1 bytes specifies a 16-bit data width and 2:2^2 bytes specifies a 32-bit data width. The BYTE_DATA_WIDTH_EXP field contains read-only configuration information for the initiator subsystem.	R	see Table 5-132
7:5	Reserved	Read returns 0.	R	0x0
4:0	ADDR_WIDTH	This field specifies the initiator subsystem address width. The ADDR_WIDTH field contains read-only configuration information for the initiator subsystem.	R	see Table 5-132

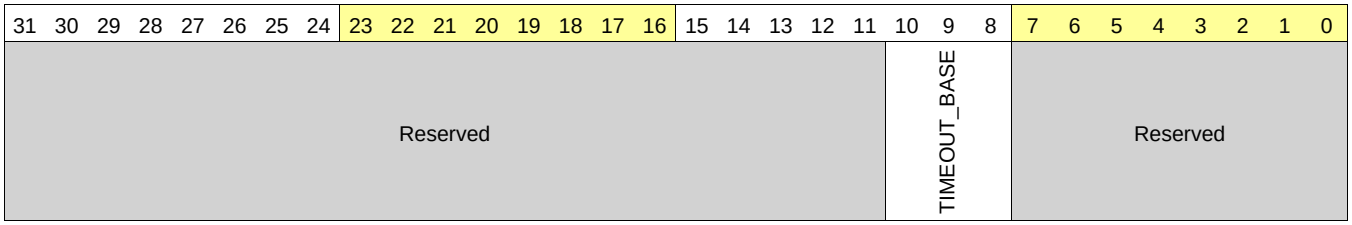
Table 5-132. Reset value for L4_LA_INITIATOR_INFO_H

Field Name	CORE_LA	PER_LA	EMU_LA	WKUP_LA
THREADS	0x4	0x4	0x2	0x2
CONNID_WIDTH	0x4	0x4	0x4	0x0
BYTE_DATA_WIDTH_EXP	0x2	0x2	0x2	0x2
ADDR_WIDTH	0x18	0x14	0x18	0x14

5.10.6.4 L4_LA_NETWORK_CONTROL_L

Table 5-133. L4_LA_NETWORK_CONTROL_L

Address Offset	0x020
Physical address	Please refer to Table 5-127
Description	Control interconnect minimum timeout values.
Type	RW



Bits	Field Name	Description	Type	Reset
31:11	Reserved	Read returns 0.	R	0x000000
10:8	TIMEOUT_BASE	The TIMEOUT_BASE field indicates the timeout period (that is, base cycles) for the highest frequency time-base signal sent from the L4 initiator subsystem to all target agents that have timeout enabled. Values for the field are: 0 - Timeout disabled 1 - L4 interconnect clock cycles divided by 64 2 - L4 interconnect clock cycles divided by 256 3 - L4 interconnect clock cycles divided by 1024 4 - L4 interconnect clock cycles divided by 4096	RW	0x4
7:0	Reserved	Read returns 0.	R	0x00

5.10.6.5 L4_LA_NETWORK_CONTROL_H

Table 5-134. L4_LA_NETWORK_CONTROL_H

Address Offset	0x024
Physical address	Please refer to Table 5-127
Description	Control interconnect global power control
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CLOCK_GATE_DISABLE	Reserved			THREAD0_PRI	Reserved											EXT_CLOCK	Reserved							

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Read returns 0.	R	0x00
24	CLOCK_GATE_DISABLE	When set to 1 this field disables all clock gating.	RW	0
23:21	Reserved	Read returns 0.	R	0x0
20	THREAD0_PRI	Sets thread priority. If the field is set to 0, the default, all initiator threads are treated the same. Setting the THREAD0_PRI field to 1 assigns a higher arbitration priority to thread 0 of the first initiator OCP interface. To avoid starvation, arbitration is imposed by the initiator subsystem. When multiple requests from different initiator threads are dispatched to targets simultaneously, the oldest request is dispatched first. If thread 0 is assigned a higher priority, a request on thread 0 always wins arbitration. Assigning thread 0 of the first initiator OCP the highest priority on a request or response can result in the starvation of other threads.	R	1
19:9	Reserved	Read returns 0.	R	0x000
8	EXT_CLOCK	When set to 1, the ext_clk_off_i signal on the initiator subsystem instructs the entire L4 to shut off.	R	1
7:0	Reserved	Read returns 0.	R	0x00

5.10.7 L4 Address Protection (AP) Register Mapping Summary

This section provides information on the L4-Core link agent (LA) register module. Each of the registers within the module instance is described separately in Table 5-135.

The AP register block contains the segment, address region, and protection group registers that can be used to specify the addressing scheme or restrict the access of target address regions.

Table 5-135. L4 AP Register Mapping Summary

Register Name	Type	Register Width (Bits)	CORE_AP Physical Address	PER_AP Physical Address	Section
L4_AP_SEGMENT_i_L[1]	RW	32	0x4804 0100 + (0x08*i)	0x4900 0100 + (0x08*i)	Section 5.10.8.1
L4_AP_SEGMENT_i_H[1]	RW	32	0x4804 0104 + (0x08*i)	0x4900 0104 + (0x08*i)	Section 5.10.8.2
L4_AP_PROT_GROUP_MEMBERS_k_L[2]	R	32	0x4804 0200 + (0x08*k)	0x4900 0200 + (0x08*k)	Section 5.10.8.3
L4_AP_PROT_GROUP_ROLES_k_L[2]	R	32	0x4804 0280 + (0x08*k)	0x4900 0280 + (0x08*k)	Section 5.10.8.4
L4_AP_REGION_l_L[3]	RW	32	0x4804 0300 + (0x08*l)	0x4900 0300 + (0x08*l)	Section 5.10.8.5
L4_AP_REGION_l_H[3]	RW	32	0x4804 0304 + (0x08*l)	0x4900 0304 + (0x08*l)	Section 5.10.8.6

(1) i = 0 to 5 for CORE_AP,
i = 0 to 4 for PER_AP,
i = 0 to 2 for EMU_AP,
i = 0 to 1 for WKUP_AP,

(2) k = 0 to 7 for CORE_ AP and PER_AP.
k = 0 to 5 for EMU_AP

(3) l = 0 to 99 for CORE_AP,
l = 0 to 42 for PER_AP,
l = 0 to 25 for EMU_AP,
l = 0 to 18 for WKUP_AP,

Table 5-136. L4 AP Register Mapping Summary

Register Name	Type	Register Width (Bits)	EMU_AP Physical Address	WKUP_AP Physical Address	Section
L4_AP_SEGMENT_i_L[1]	RW	32	0x5400 6100 + (0x08*i)	0x4832 8100 + (0x08*i)	Section 5.10.8.1
L4_AP_SEGMENT_i_H[1]	RW	32	0x5400 6104 + (0x08*i)	0x4832 8104 + (0x08*i)	Section 5.10.8.2
L4_AP_PROT_GROUP_MEMBERS_k_L[2]	R	32	0x5400 6200 + (0x08*k)	N/A	Section 5.10.8.3
L4_AP_PROT_GROUP_ROLES_k_L[2]	R	32	0x5400 6280 + (0x08*k)	N/A	Section 5.10.8.4
L4_AP_REGION_l_L[3]	RW	32	0x5400 6300 + (0x08*l)	0x4832 8300 + (0x08*l)	Section 5.10.8.5
L4_AP_REGION_l_H[3]	RW	32	0x5400 6304 + (0x08*l)	0x4832 8304 + (0x08*l)	Section 5.10.8.6

(1) i = 0 to 5 for CORE_AP,
i = 0 to 4 for PER_AP,
i = 0 to 2 for EMU_AP,
i = 0 to 1 for WKUP_AP,

(2) k = 0 to 7 for CORE_ AP and PER_AP.
k = 0 to 5 for EMU_AP

(3) l = 0 to 99 for CORE_AP,
l = 0 to 42 for PER_AP,
l = 0 to 25 for EMU_AP,
l = 0 to 18 for WKUP_AP,

5.10.7.1 Reset Values

Table 5-137. Reset Values for CORE_AP L4_AP_REGION_I_L and L4_AP_REGION_I_H

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
0	0x00 0000	1	0	2	0x0B
1	0x00 0800	1	7	2	0x0B
2	0x00 1000	1	7	2	0x0C
3	0x01 0000	1	2	2	0x0A
4	0x01 0400	1	2	2	0x0A
5	0x01 0800	1	2	2	0x0A
6	0x01 0C00	1	2	2	0x0A
7	0x01 1000	1	2	2	0X0C
8	0x00 0000	3	3	2	0x0C
9	0x01 6000	1	7	2	0x0C
10	0x01 7000	1	7	2	0x0C
11	0x01 8000	1	7	2	0x0C
12	0x01 C000	1	7	2	0x0C
13	0x02 B000	2	7	2	0x0C
14	0x02 C000	2	7	2	0x0C
15	0x01 E000	1	7	2	0x0C
16	0x01 F000	1	7	2	0x0C
17	0x02 A000	1	7	2	0x0C
18	0x02 B000	1	7	2	0x0C
19	0x02 C000	1	7	2	0X0C
20	0x02 D000	1	7	2	0x0C
21	0x03 0000	1	7	1	0x0C
22	0x03 1000	1	7	1	0x0C
23	0x03 2000	1	7	1	0x0C
24	0x03 3000	1	7	1	0x0C
25	0x03 4000	1	7	2	0x0C
26	0x03 5000	1	7	2	0x0C
27	0x00 6000	2	7	2	0x0C
28	0x00 7000	2	7	2	0x0C
29	0x00 8000	2	7	2	0x0C
30	0x00 9000	2	7	2	0x0C
31	0x00 9000	3	7	2	0x0C
32	0x00 A000	3	7	2	0x0C
33	0x01 2000	2	7	2	0x0C
34	0x01 3000	2	7	2	0x0C
35	0x01 4000	2	7	2	0x0C
36	0x01 5000	2	7	2	0x0C
37	0x01 8000	2	7	2	0x0C
38	0x01 9000	2	7	2	0x0C
39	0x01 A000	2	7	2	0x0C
40	0x01 B000	2	7	2	0x0C
41	0x01 C000	2	7	2	0x0C
42	0x01 D000	2	7	2	0x0C
43	0x03 4000	2	7	2	0x0C
44	0x03 5000	2	7	2	0x0C
45	0x01 E000	2	7	2	0x0C

Table 5-137. Reset Values for CORE_AP L4_AP_REGION_I_L and L4_AP_REGION_I_H (continued)

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
46	0x01 F000	2	7	2	0x0C
47	0x02 0000	2	1	2	0x0C
48	0x02 1000	2	1	2	0x0C
49	0x02 2000	2	1	2	0x0C
50	0x02 3000	2	1	2	0x0C
51	0x02 4000	2	1	2	0x0C
52	0x02 5000	2	1	2	0x0C
53	0x02 6000	2	1	2	0x0C
54	0x02 7000	2	1	2	0x0C
55	0x02 8000	2	1	2	0x0D
56	0x02 A000	2	1	2	0x0C
57	0x03 0000	2	7	2	0x0C
58	0x03 1000	2	7	2	0x0C
59	0x03 2000	2	7	2	0x0C
60	0x03 3000	2	7	2	0x0C
61	0x01 6000	2	7	2	0x0C
62	0x01 7000	2	7	2	0x0C
63	0x01 9000	1	7	2	0x0C
64	0x01 A000	1	7	2	0x0C
65	0x01 B000	1	7	2	0x0C
66	0x03 8000	2	7	2	0x0C
67	0x03 9000	2	7	2	0x0C
68	0x00 4000	0	7	2	0x0C
69	0x00 7000	0	7	2	0x0C
70	0x00 B000	3	7	2	0x0C
71	0x00 C000	3	7	2	0x0C
72	0x00 6000	0	7	2	0x0B
73	0x02 0000	1	7	1	0x0C
74	0x02 1000	1	7	1	0x0C
75	0x00 2000	0	7	2	0x0C
76	0x00 3000	0	7	2	0x0C
77	0x03 C000	2	3	3	0x0C
78	0x00 4000	4	4	2	0x0D
79	0x03 A000	2	7	2	0x0C
80	0x03 B000	2	7	2	0x0C
81	0x00 0000	5	7	2	0x0C
82	0x00 1000	3	1	2	0x0C
83	0x00 2000	3	1	2	0x0C
84	0x00 3000	3	1	2	0x0C
85	0x00 4000	3	1	2	0x0C
86	0x00 5000	3	1	2	0x0C
87	0x00 6000	3	1	2	0x0C
88	0x03 6000	2	7	2	0x0C
89	0x03 7000	2	7	2	0x0C
90	0x00 7000	3	7	2	0x0C
91	0x00 8000	3	7	2	0x0C
92	0x00 D000	3	7	2	0x0C

Table 5-137. Reset Values for CORE_AP L4_AP_REGION_I_L and L4_AP_REGION_I_H (continued)

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
93	0x00 E000	3	7	2	0x0C
94	0x00 6000	4	7	2	0x0D
95	0x00 8800	4	7	2	0x0B
96	0x00 9000	4	7	2	0x0C
97	0x00 C000	4	1	2	0x0D
98	0x01 0000	4	7	2	0x10
99	0x02 0000	4	7	2	0x10

Table 5-138. Reset Values for PER_AP L4_AP_REGION_I_L and L4_AP_REGION_I_H

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
0	0x00 0000	0	0	2	0x0B
1	0x00 0800	0	7	2	0x0B
2	0x00 1000	0	7	2	0x0C
3	0x00 0000	1	7	2	0x0C
4	0x00 1000	1	7	2	0X0C
5	0x00 2000	1	7	2	0X0C
6	0x00 3000	1	7	2	0X0C
7	0x00 4000	1	7	2	0X0C
8	0x00 5000	1	7	2	0x0C
9	0x00 6000	1	7	2	0x0C
10	0x00 7000	1	7	2	0x0C
11	0x00 0000	1	7	2	0x0C
12	0x00 1000	2	7	2	0x0C
13	0x00 2000	2	7	2	0x0C
14	0x00 3000	2	7	2	0x0C
15	0x00 4000	2	7	2	0x0C
16	0x00 5000	2	7	2	0x0C
17	0x00 6000	2	7	2	0x0C
18	0x00 7000	2	7	2	0x0C
19	0x00 8000	2	7	2	0X0C
20	0x00 9000	2	7	2	0x0C
21	0x00 A000	2	7	1	0x0C
22	0x00 B000	2	7	2	0x0C
23	0x00 C000	2	7	2	0x0C
24	0x00 D000	2	7	2	0x0C
25	0x00 E000	2	7	2	0x0C
26	0x00 F000	2	7	2	0x0C
27	0x00 0000	3	7	2	0x0C
28	0x00 1000	3	7	2	0x0C
29	0x00 0000	4	7	2	0x0C
30	0x00 1000	4	7	2	0x0C
31	0x00 2000	4	7	2	0x0C
32	0x00 3000	4	7	2	0x0C
33	0x00 4000	4	7	2	0x0C
34	0x00 5000	4	7	2	0x0C
35	0x00 6000	4	7	2	0x0C

Table 5-138. Reset Values for PER_AP L4_AP_REGION_l_L and L4_AP_REGION_l_H (continued)

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
36	0x00 7000	4	7	2	0x0C
37	0x00 8000	4	7	2	0x0C
38	0x00 9000	4	7	2	0x0C
39	0x00 8000	4	7	2	0x0C
40	0x00 9000	1	7	2	0x0C
41	0x00 A000	1	7	2	0x0C
42	0x00 B000	1	7	2	0x0C

Table 5-139. Reset Values for EMU_AP L4_AP_REGION_l_L and L4_AP_REGION_l_H

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
0	0x00 0000	2	5	2	0x14
1	0x00 4000	0	5	2	0x0C
2	0x00 5000	0	5	2	0x0C
3	0x00 6000	0	5	0	0x0B
4	0x00 6800	0	5	2	0x0B
5	0x00 7000	0	5	2	0x0B
6	0x00 8000	0	5	2	0x0C
7	0x01 8000	0	3	2	0x0C
8	0x01 9000	0	3	2	0x0C
9	0x01 A000	0	3	2	0x0C
10	0x01 B000	0	3	2	0x0C
11	0x01 C000	0	3	2	0x0C
12	0x01 D000	0	3	2	0x0C
13	0x01 E000	0	3	2	0x0C
14	0x10 0000	1	5	2	0x0C
15	0x01 F000	0	5	2	0x0C
16	0x01 0000	0	3	2	0x0E
17	0x10 6000	2	5	2	0x0C
18	0x10 8000	2	5	2	0x0C
19	0x10 4000	2	2	2	0X0D
20	0x10 8800	2	5	2	0x0B
21	0x10 9000	2	5	2	0x0C
22	0x10C000	2	1	2	0x0D
23	0x11 0000	2	5	2	0x10
24	0x12 0000	2	5	2	0x10
25	0x13 0000	2	5	2	0x0C

Table 5-140. Reset Values for WKPUP_AP L4_AP_REGION_I_L and L4_AP_REGION_I_H

y	BASE	SEGMENT_ID	PROT_GROUP_ID	BYTE_DATA_WIDTH_EXP	SIZE
0	0x00 0000	1	0	2	0x0B
1	0x00 6000	0	0	2	0x0D
2	0x00 9000	0	0	2	0x0C
3	0x00 C000	0	0	2	0x0C
4	0x00 D000	0	0	2	0x0C
5	0x01 8000	0	0	2	0x0C
6	0x01 9000	0	0	2	0x0C
7	0x01 4000	0	0	2	0x0C
8	0x01 5000	0	0	2	0x0C
9	0x00 9000	1	0	2	0x0C
10	0x00 4000	0	0	2	0x0C
11	0x00 5000	0	0	2	0x0C
12	0x00 8800	1	0	2	0x0B
13	0x00 8000	0	0	2	0x0B
14	0x01 0000	0	0	2	0x0C
15	0x01 1000	0	0	2	0x0C
16	0x00 0000	1	0	2	0x0C
17	0x00 1000	1	0	2	0x0C
18	0x00 A000	1	0	2	0x0C

5.10.8 L4 Address Protection (AP) Register Descriptions

5.10.8.1 L4_AP_SEGMENT_i_L

Table 5-141. L4_AP_SEGMENT_i_L

Address Offset	0x100 + (0x08*i)	**Index**	i = 0 to 5 for CORE_AP, i = 0 to 4 for PER_AP, i = 0 to 2 for EMU_AP, i = 0 to 1 for WKUP_AP,
Physical address	Please refer to Table 5-135		
Description	Define the base address of each segments		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	The base address of the segment (with 0s from bit 0 to bit SIZE-1).	R	see Table 5-142

Table 5-142. Reset Values for L4_AP_SEGMENT_i_L

BASE	CORE_AP	PER_AP	EMU_AP	WKUP_AP
i = 0	0x00 0000	0x00 0000	0x00 0000	0x00 0000
i = 1	0x04 0000	0x02 0000	0x40 0000	0x02 0000
i = 2	0x08 0000	0x03 0000	0x60 0000	N/A
i = 3	0x0C 0000	0x04 0000	N/A	N/A
i = 4	0x30 0000	0x05 0000	N/A	N/A
i = 5	0x32 0000	N/A	N/A	N/A

5.10.8.2 L4_AP_SEGMENT_i_H

Table 5-143. L4_AP_SEGMENT_i_H

Address Offset	0x104 + (0x08*i)	**Index**	i = 0 to 5 for CORE_AP, i = 0 to 4 for PER_AP, i = 0 to 2 for EMU_AP, i = 0 to 1 for WKUP_AP,
Physical address	Please refer to Table 5-135		
Description	Define the size of each segments		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								SIZE							

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Read returns 0.	R	0x0000000
4:0	SIZE	Segment size is a power of 2, where 2^SIZE is the byte size of a segment (all segment registers use the same size).	R	see Table 5-144

Table 5-144. Reset Values for L4_AP_SEGMENT_i_H

SIZE	CORE_AP	PER_AP	EMU_AP	WKUP_AP
For all i	0x12	0x10	0x15	0x11

5.10.8.3 L4_AP_PROT_GROUP_MEMBERS_k_L

Table 5-145. L4_AP_PROT_GROUP_MEMBERS_k_L

Address Offset	0x200 + (0x08*k)	**Index**	k = 0 to 7 for CORE_ AP and PER_AP. k = 0 to 5 for EMU_AP
Physical address	Please refer to Table 5-135		
Description	Define connID bit vectors for a protection group.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CONNID_BIT_VECTOR															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0	R	0x0000
15:0	CONNID_BIT_VECTOR	A bit of 1 in position n means that connID n is allowed in this protection group. Illegal connIDs have their bits set to 0s.	R	0xFFFF

5.10.8.4 L4_AP_PROT_GROUP_ROLES_k_L

Table 5-146. L4_AP_PROT_GROUP_ROLES_k_L

Address Offset	0x200 + (0x08*k)	**Index**	k = 0 to 7 for CORE_ AP and PER_AP. k = 0 to 5 for EMU_AP
Physical address	Please refer to Table 5-135		
Description	Define MReqInfo bit vectors for a protection group.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ENABLE															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0	R	0x0000
15:0	ENABLE	Setting of type access allowed for the group of initiators see Table 5-21.	R	0xFFFF

5.10.8.5 L4_AP_REGION_l_L

Table 5-147. L4_AP_REGION_l_L

Address Offset	0x300 + (0x08*l)	**Index**	l = 0 to 99 for CORE_AP, l = 0 to 42 for PER_AP, l = 0 to 25 for EMU_AP, l = 0 to 18 for WKUP_AP,
Physical address	Please refer to Table 5-135		
Description	Define the base address of the region in respect to the segment it belongs to.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BASE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Read returns 0.	R	0x00
23:0	BASE	Sets the base address of this region relative to its segment base.	R	See Table 5-137 to Table 5-140

5.10.8.6 L4_AP_REGION_I_H

Table 5-148. L4_AP_REGION_I_H

Address Offset	0x304 + (0x08*I)	**Index**	I = 0 to 99 for CORE_AP, I = 0 to 42 for PER_AP, I = 0 to 25 for EMU_AP, I = 0 to 18 for WKUP_AP,
Physical address	Please refer to Table 5-135		
Description	Define the size, protection group^and segment ID of the region		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SEGMENT_ID				Reserved	PROT_GROUP_ID			BYTE_DATA_WIDTH_EXP				Reserved								SIZE						ENABLE	

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Read returns 0	R	0x0
27:24	SEGMENT_ID	Identifies the segment to which the region is part of.	R	See Table 5-137 to Table 5-140
23	Reserved	Read returns 0.	R	0
22:20	PROT_GROUP_ID	The protection group containing this region.	RW	See Table 5-137 to Table 5-140
19:17	BYTE_DATA_WIDTH_EXP	The target OCP data byte width is 2^(BYTE_DATA_WIDTH_EXP) bytes. The value of this field is derived from the target OCP data_width parameter.	R	See Table 5-137 to Table 5-140
16:6	Reserved	Read returns 0.	R	0x0
5:1	SIZE	Define the size of the region in bytes. 2^SIZE equals the region.	R	See Table 5-137 to Table 5-140
0	ENABLE	0x0: Disable the region, no access allows 0x1: Enable the region, with access as define in registers	R	See Table 5-137 to Table 5-140

System Control Module

This document describes the system control module.

NOTE: This chapter gives information about all modules and features. See Chapter 1, to check availability of modules and features on your specific device.

Topic **Page**

6.1 System Control Module Overview

The system control module (SCM) allows software control of the various static modes supported by the device. The SCM is on the L4-Core interconnect, but it is sensitive only to the device internal power-on reset. It is not affected by the L4-Core reset.

For emulator devices, the power-on reset can also be controlled by the debugger through the JTAG interface.

The device uses the SCM as the primary point of control for these areas:

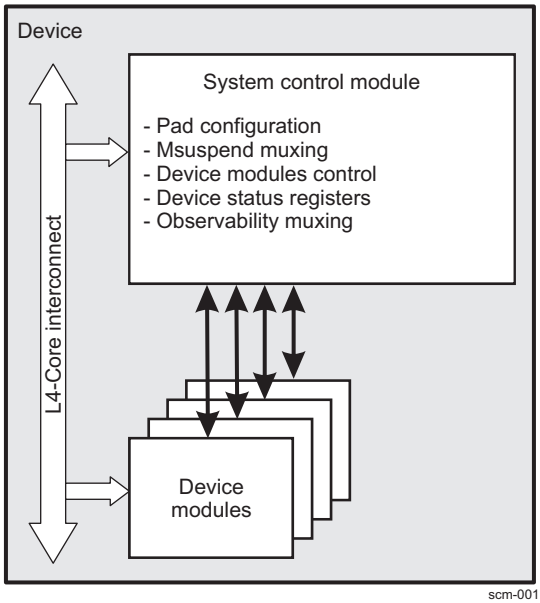
- Functional I/O multiplexing
- Pad configuration (pull-up or pull-down enable)
- Emulation controls
- Device status
- Peripheral sensitivity to the MPU MSuspend signals
- Static device configuration
- Debug and observability I/O multiplexing

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

NOTE: Power domains are not supported but the IP/Modules are still part of the logical design partition in the processor's domain partition architecture.

Figure 6-1 provides an overview of the SCM.

Figure 6-1. System Control Module Overview



The SCM primarily implements a bank of registers accessible by the software. Some are read-only registers that carry status information, and others are fully accessible (read/write).

The read/write registers are divided into the following classes:

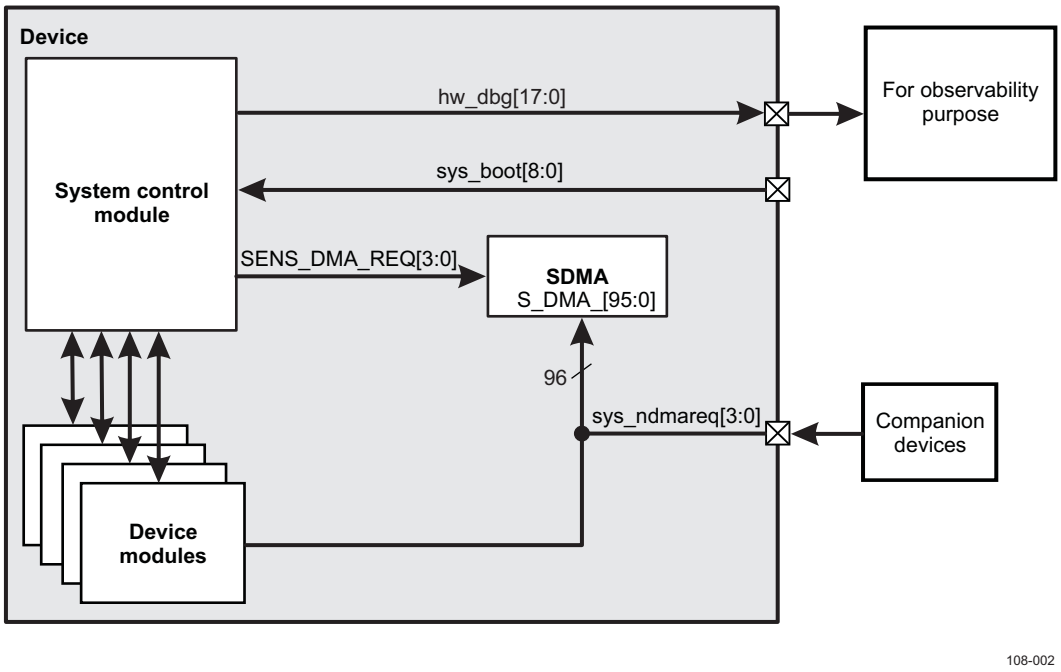
- Pad functional multiplexing and configuration registers (32-bit registers, one register per two pins)
- Debug and observability I/O multiplexing register (32-bit read/write register)
- Static device configuration registers (32-bit read/write registers for module specific configuration)
- Scratchpad memory bank (256* 32-bit)

6.2 System Control Module Environment

The SCM allows the setting of external system DMA (sDMA) request pin sensitivity and controls the multiplexing of device internal modules signals routed to external pins for hardware debug purposes. The SCM also integrates the decoding logic of sys_boot[8:0].

Figure 6-2 shows an overview of the SCM environment.

Figure 6-2. System Control Module Environment Overview



The seven sys_ndamreq[3:0] pins are optional external direct memory access (DMA) requests that can be managed directly by the sDMA controller. The SCM can configure these requests to either level sensitive (active low) or edge sensitive (falling edge) through the correct setting of the SENSDMAREQN bit (where N is between 0 and 3) in the CONTROL.CONTROL_DEVCONF0 and CONTROL.CONTROL_DEVCONF1 registers.

The sys_boot[5:0] pins are read-accessible in a status register (SYSBOOT field CONTROL.CONTROL_STATUS[5:0]) following a power-on reset. The SCM does not use sys_boot[6]. Sys_boot[8:7] are read accessible in CONTROL_DEVCONF2[18:19].

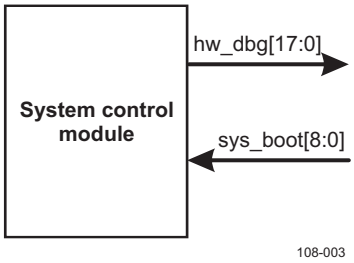
With the correct pad configuration, the SCM maps the hw_dbg[17:0] pins at the device boundary to observe hardware debug signals from device modules. The internal observable signals are PRCM signals, DMA requests, and interrupts.

6.2.1 Functional Interfaces

6.2.1.1 Basic System Control Module Pins

Figure 6-3 shows the SCM functional interface configured to observe device module debug signals.

Figure 6-3. System Control Module Interface Signals



6.2.1.2 System Control Module Interface Description

Table 6-1 lists the SCM input and output configured to observe device module debug signals.

Table 6-1. SCM I/O Description

Signal Name	I/O[1]	Description	Reset Value
hw_dbg[17:0]	O	Debug signals 0 to 17	N/A
sys_boot[8:0]	I	Boot configuration mode bits 0 to 8	Unknown

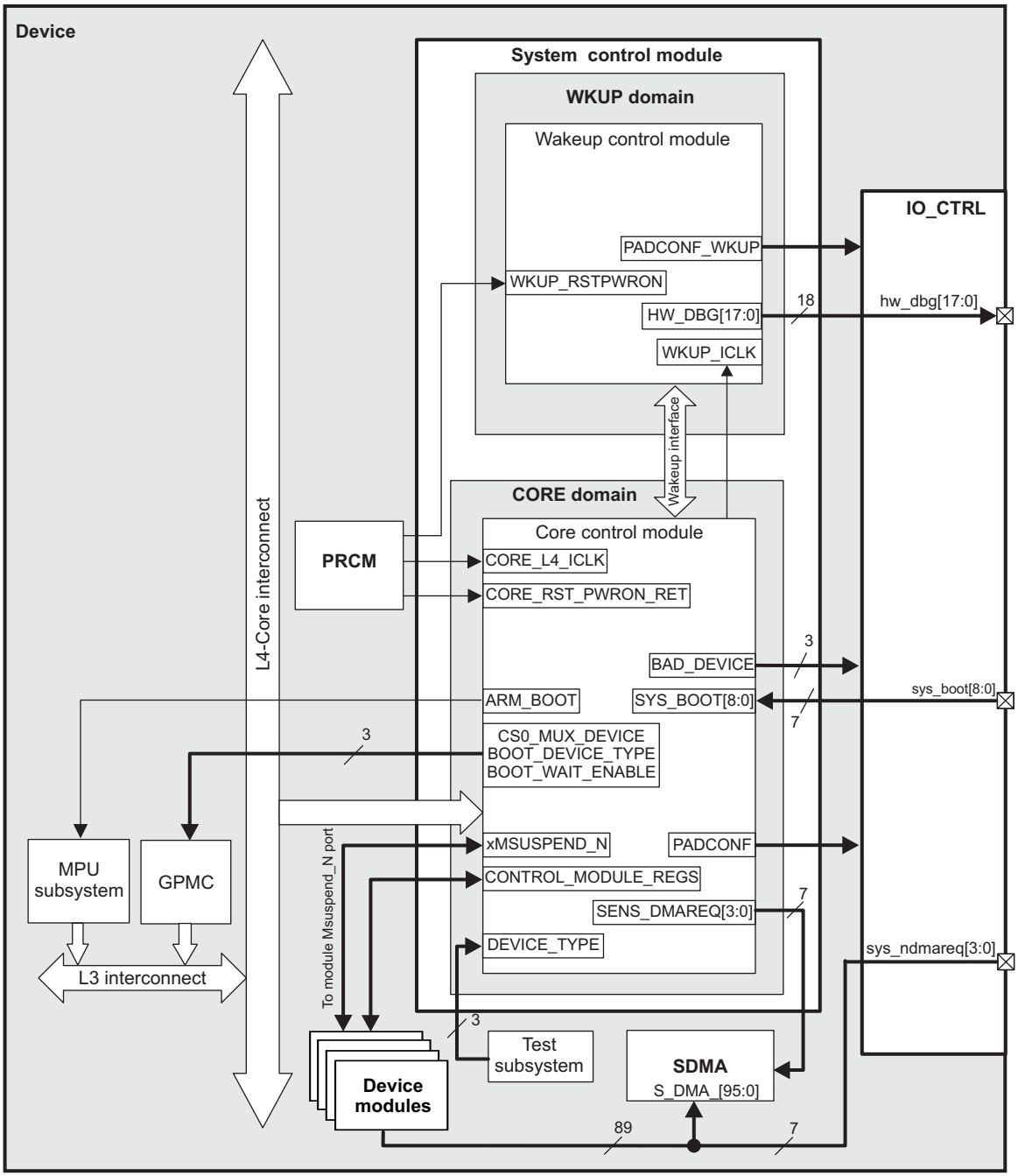
(1) I = Input, O = Output

6.3 System Control Module Integration

This section describes the integration of the SCM within the device.

Figure 6-4 shows the SCM integration in the device.

Figure 6-4. System Control Module Integration



The SCM is split into two blocks: the core control module in the CORE domain, and the wake-up control module in the WKUP domain. The wake-up control module contains some observability multiplexing, pad configurations, and security registers. For further information on the wake-up control module, see Section 6.4.3, *Wake-up Control Module*.

The software sets the configuration registers to the desired values depending on the configuration of the requested device. Static device configuration registers can be set by the software at any time and are effective immediately.

The SCM is connected on the L4-Core interconnect. The power, reset, and clock management (PRCM) module provides the module interface clock (CORE_L4_ICLK) and the power-on reset signal. The PRCM generates one global reset per domain (CORE_RST_PWRON_RET for the CORE domain and WKUP_RSTPWRON for the WKUP domain). The SCM does not respond to a warm reset or to an L4 reset.

6.3.1 Clocking, Reset, and Power-Management Scheme

6.3.1.1 Clock

The main sequential logic within the SCM is accessible in a register file through the L4-Core. The only clock provided to the SCM is the interface clock, CORE_L4_ICLK. This clock comes from the PRCM module and is controlled by the EN_SCMCTRL bit PRCM.CM_ICLKEN1_CORE[6] (0 = disables the clock, 1 = enables the clock) and the AUTO_SCMCTRL bit PRCM.CM_AUTOIDLE1_CORE[6] (enables/disables automatic control of the interface clock).

The wake-up control module is configured through the L4-Core interface of the core control module and is accessed from the core control module through a dedicated interface. This interface uses the L4-Core interface clock (CORE_L4_ICLK) divided by 4 or 2 according to the WKUPCTRLCLOCKDIV bit CONTROL.CONTROL_PADCONF_OFF[2]. Only this wake-up interface clock (WKUP_ICLK) is propagated to the wake-up control module.

For further information, see the *Power, Reset, and Clock Management* chapter.

6.3.1.2 Resets

The SCM responds only to the internal power-on reset and to the device type (secure, emulator, general-purpose, test, or bad). The SOFTRESET bit CONTROL.CONTROL_SYSCONFIG[1] has no effect; the SCM is not affected by a warm reset.

The internal power-on reset is not a direct image of the power-on reset input pin (SYS_NRESPWRON). The PRCM module generates an internal power-on reset signal per domain and activates the internal power-on reset when the eFuse-related settings (such as the device type) are initialized. The core control module of the CORE domain responds to the CORE power-on reset (CORE_RST_PWRON_RET). The wake-up control module of the WKUP domain responds to the power-on reset (WKUP_RSTPWRON).

NOTE: References in the TRM to the power-on reset refer to the internal power-on reset as seen by the SCM.

On emulator devices, the debugger can also control the internal power-on reset signal though the JTAG interface.

For further information, see the *Power, Reset, and Clock Management* chapter.

6.3.1.3 Power Management

6.3.1.3.1 System Power Management

The PRCM module can require the SCM to be idled to save power. The SCM enters idle mode through the IDLEMODE field CONTROL.CONTROL_SYSCONFIG[4:3].

The PRCM module requests idle mode, and the SCM always accepts an idle command. Idle mode can be configured to either force-idle or smart-idle mode.

When the SCM is in force-idle mode (IDLEMODE bits CONTROL.CONTROL_SYSCONFIG[4:3] = 0b00) and it receives an idle request from the PRCM module (PRCM.CM_ICLKEN1_CORE[6] set to 0 or PRCM.CM_AUTOIDLE1_CORE[6] set to 1 and the L4-Core interface clock idle transitions), the SCM waits unconditionally for active system clock gating by the PRCM module. Active system clock gating occurs only when all peripherals supplied by the same L4-Core interface clock domain are also ready for idle.

In smart-idle mode (IDLEMODE bits CONTROL.CONTROL_SYSCONFIG[4:3] = 0b10), an idle command is accepted only when the save mechanism completes.

In idle mode (when the PRCM module has gated the interface clock), the SCM is not active and the interface clock paths are gated.

NOTE: The ST_SCMCTRL bit PRCM.CM_IDLEST1_CORE[6] (0b0 = active, 0b1 = idle) can check the SCM idle state.

The SCM idle mode is a function of the EN_SCMCTRL bit PRCM.CM_ICLKEN1_CORE[6] configuration and can be controlled automatically with hardware, depending on the AUTO_SCMCTRL bit PRCM.CM_AUTOIDLE1_CORE[6] configuration.

6.3.1.3.2 Module Power Saving

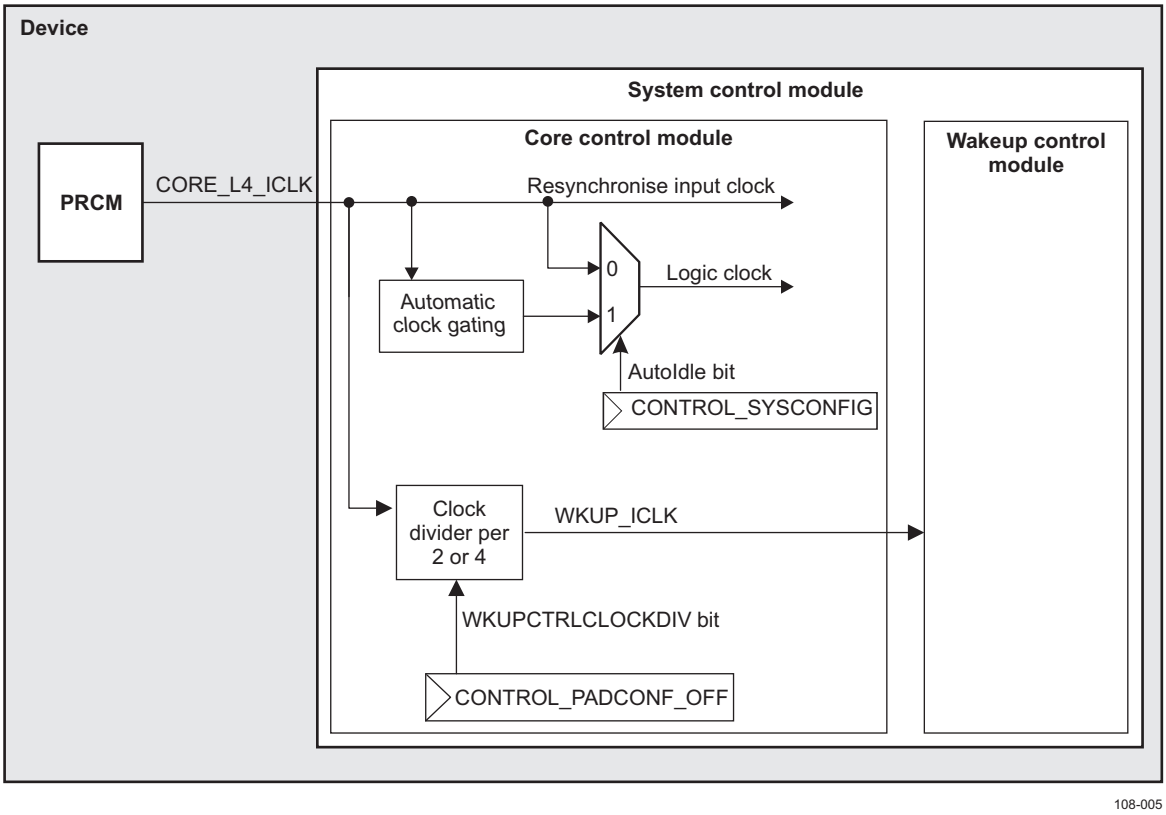
An internal interface clock-gating feature provides SCM local power management. The clock for the logic within the module can be gated when there is no access to the module according to the AUTOIDLE bit CONTROL.CONTROL_SYSCONFIG[0]. Otherwise, this logic is free-running on the interface clock CORE_L4_ICLK.

The L4-Core interface clock (CORE_L4_ICLK) is also used to synchronize and resample module inputs. The clock for those functions must always be free-running. Therefore, it is not gated.

The wake-up control module is clocked only by a local wake-up interface clock from the core control module. The wake-up interface clock (WKUP_ICLK) uses the L4-Core interface clock (CORE_L4_ICLK) divided by 4 or 2 to reduce power consumption according to the WKUPCTRLCLOCKDIV bit CONTROL.CONTROL_PADCONF_OFF[2] (0 = divided by 4, 1 = divided by 2).

Figure 6-5 shows the SCM internal clock implementation.

Figure 6-5. Internal Clock Implementation



6.3.2 Hardware Requests

The SCM does not generate interrupt or wake-up requests.

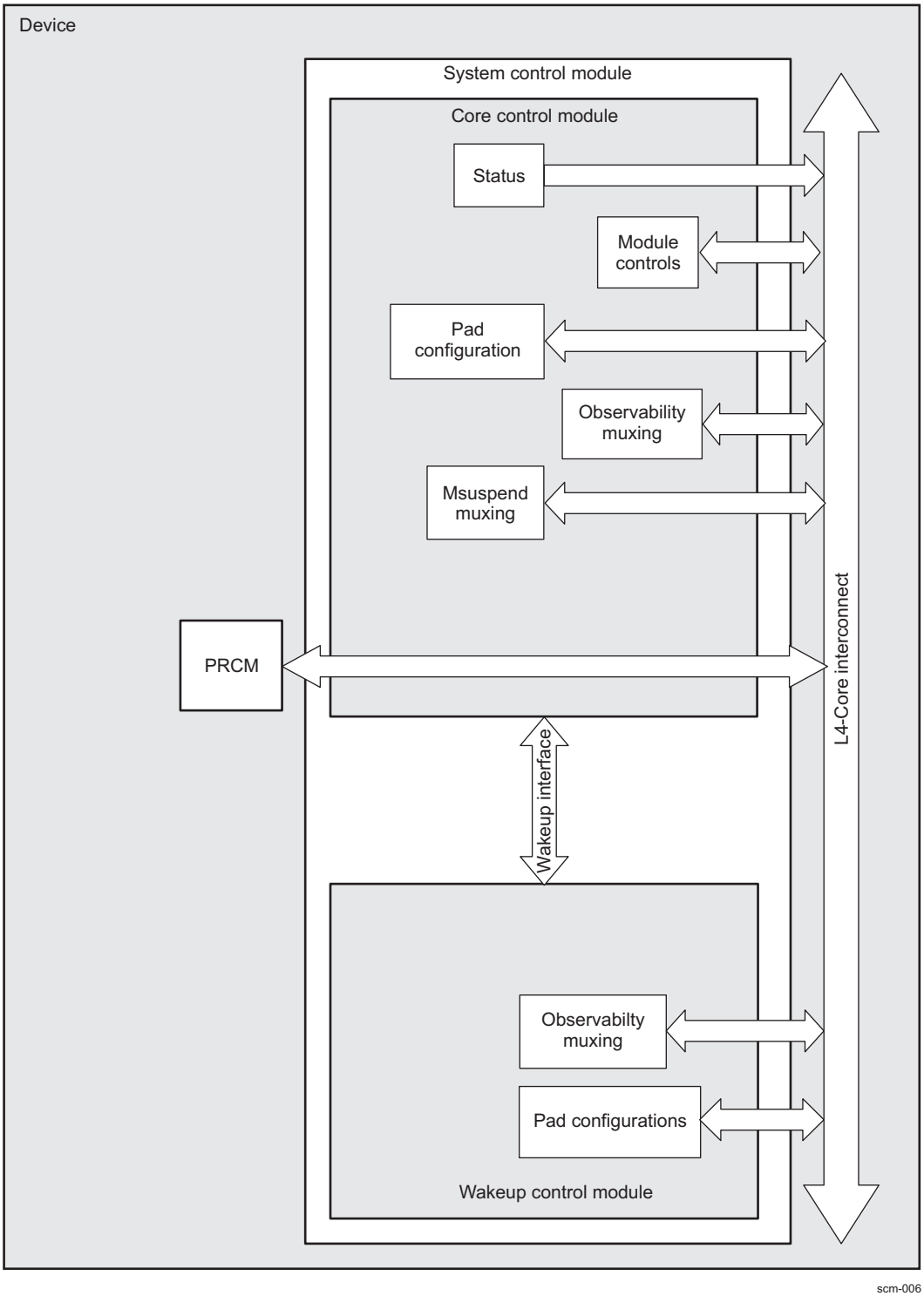
6.4 System Control Module Functional Description

6.4.1 Block Diagram

The SCM controls various device modules settings through register configuration and internal signals. It also controls the pad configuration and multiplexing and the routing of internal signals (such as PRCM signals or DMA requests) to observable pins for debug.

Figure 6-6 shows the SCM block diagram.

Figure 6-6. System Control Module Block Diagram



The following sections describe the functionality of the SCM registers.

6.4.2 System Control Module Initialization

The SCM responds only to the internal power-on reset and to the device type. At power-on, reset values for the registers define the safe state for the device. In the initialization mode, only modules used at boot time are associated with the pads. Other module inputs are internally tied, and outputs pads are turned off each time the feature is available.

For the pad configuration, pull-up/pull-down fields are set according to the device pin list. For further information, see the pin list tables in your device-specific data manual.

The device type (for example, emulator, secure, general-purpose) affects the reset values and access rights for some emulation and security-related configuration bits (see Section 6.6, *SCM Registers*).

General-purpose devices (that is, those with no secure mode available) include features that are inaccessible or unavailable. These inaccessible registers define the default or fixed device configuration or behavior.

This technical reference manual focuses only on GP devices. To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

6.4.3 Wake-Up Control Module

The wake-up control module in the SCM belongs to the WKUP domain. It contains a 1 K-byte memory which is used to save the pad configuration registers in the core control module before switching to off mode. Pad configuration registers driving the I/O pad control in the WKUP domain are also instantiated in the wake-up control module. In this module, there is also a subset of observability multiplexing registers dedicated to PRCM observable signals and security registers (CONTROL.CONTROL_SEC_TAP and CONTROL.CONTROL_SEC_EMU).

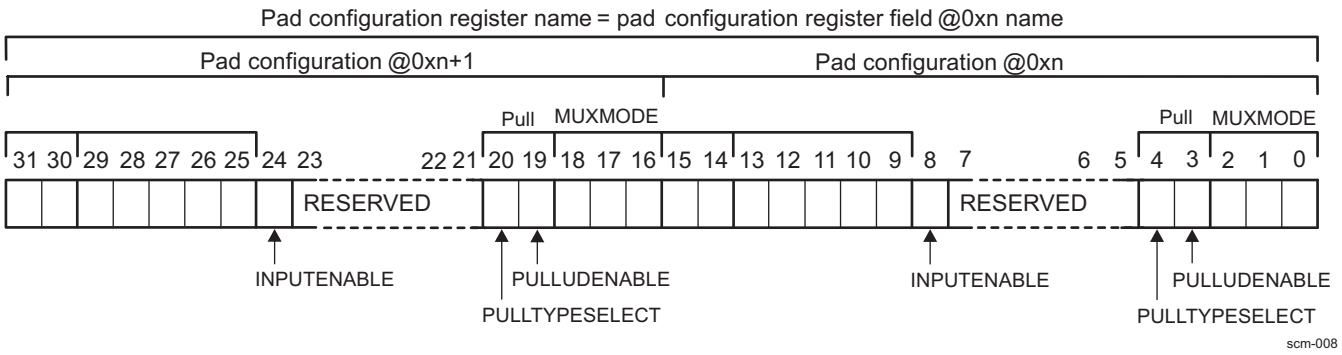
The wake-up control module is configured through the L4-Core interface in the core control module and is accessed from the core control module through a dedicated interface. This interface between the core control module and the wake-up control module uses the CORE_L4_ICLK clock divided by 4 or 2 to reduce power consumption in the WKUP domain.

6.4.4 Pad Functional Multiplexing and Configuration

After power-on reset, the software sets the pad functional multiplexing and configuration registers to the requested device pad configurations. Data written in these registers command directly the multiplexing of the pad configuration logic.

Each pin is configurable by software using its associated pad configuration register field, which is 16 bits wide (see Figure 6-7).

Figure 6-7. Pad Configuration Register Functionality



One pad configuration register field is available for each pin. Each 32-bit pad configuration register is grouped into two 16-bit pad configuration register fields. One pad configuration register provides control for two different pins.

Some pad configuration registers control the configuration of pads in the CORE domain. These registers are instantiated in the CORE domain of the SCM (core control module, physical addresses 0x4800 2030 to 0x4800 2260). Pad configuration registers also control the configuration of pads in the WKUP domain. These registers are instantiated in the WKUP domain of the SCM (wake-up control module, physical addresses 0x4800 2A00 to 0x4800 2A4C).

NOTE: These registers can be accessed using 8-, 16-, and 32-bit operations.

The functional bits of a pad configuration register field are divided into the following five fields:

- MUXMODE (3 bits) defines the multiplexing mode applied to the pin. A mode corresponds to the selection of the functionality mapped on the pin with six (0 to 5) possible functional modes for each pin.
- PULL (2 bits) for combinational pull-up/pull-down configuration:
  - PULLTYPESELECT: Pull-up/pull-down selection for the pin.
  - PULLUDENABLE: Pull-up/pull-down enable for the pin.
- INPUTENABLE (1 bit) drives an input enable signal to the I/O CTRL.
  - INPUTENABLE = 0: Input Disable. Pin is configured in output only mode.
  - INPUTENABLE = 1: Input Enable. Pin is configured in bi-directional mode.

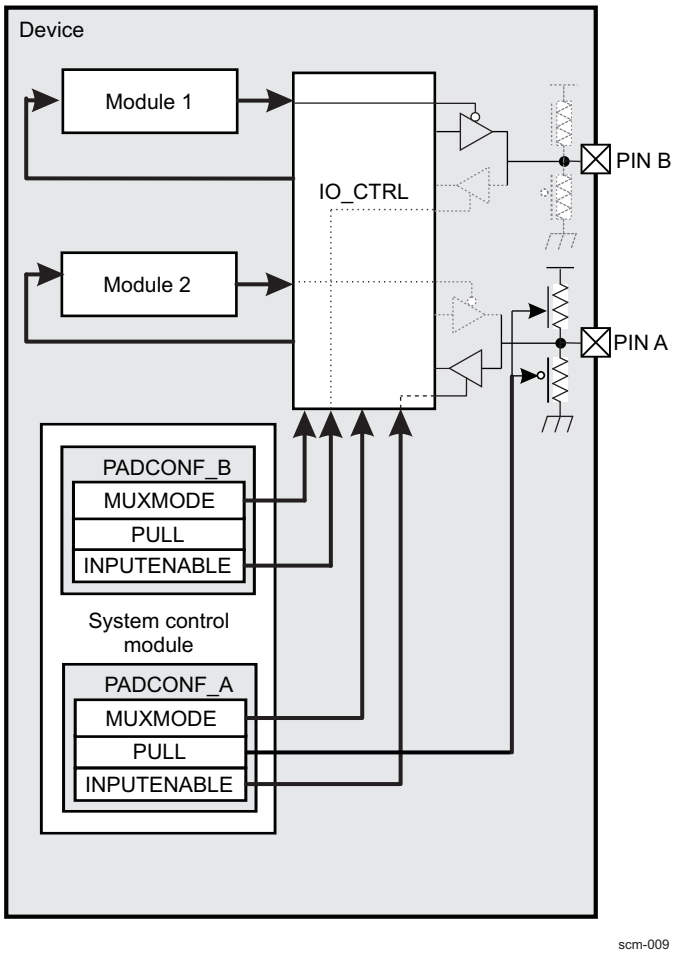
Table 6-2 lists the bit directions of the CONTROL_PADCONF_x registers.

Table 6-2. Bit Directions for CONTROL_PADCONF_x Registers

CONTROL_PADCONF_x Bit	Bit Direction	
	0	1
PULLUDENABLE	Not activated	Activated
PULLTYPESELECT	Pulldown	Pullup
INPUTENABLE	Input enable signal inactive	Input enable signal active

Figure 6-8 shows the pad configuration functionality when off mode is inactive.

Figure 6-8. Pad Configuration Diagram



6.4.4.1 Mode Selection

Table 6-3 lists the multiplexing modes and settings.

Table 6-3. Mode Selection

MUXMODE	Selected Mode
0b000	Primary mode = Mode 0
0b001	Mode 1
0b010	Mode 2
0b011	Mode 3
0b100	Mode 4
0b101	Mode 5
0b110	Mode 6
0b111	Safe mode = Mode 7

The MUXMODE field in the CONTROL_PADCONF_X register defines the multiplexing mode applied to the pad. Modes are referred to by their decimal (from 0 to 7) or binary (from 0b000 to 0b111) representation. Functional modes are defined from 0b000 to 0b101; mode 0b111 is referred to as the safe mode.

For most pads, the reset value for the MUXMODE field in the CONTROL_PADCONF_X register is 0b111. The exceptions are pads to be used at boot time to transfer data from selected peripherals to the external flash memory.

Mode 0 is the primary mode. When mode 0 is set, the function mapped to the pin to the name of the pin.

Mode 1 to mode 6 are possible modes for alternate functions. On each pin, some modes are used effectively for alternate functions, while other modes are unused and correspond to no functional configuration.

The safe mode avoids any risk of electrical contention by configuring the pin as an input with no functional interface mapped to it. The safe mode is used mainly as the default mode for all pins containing no mandatory interface at the release of power-on reset.

For more information about the configurable mode on each pin, see Table 6-5 .

6.4.4.2 Pull Selection

Whichever pull value is configured, pulls are automatically disabled when a pin is configured as an output (see Table 6-4).

Table 6-4. Pull Selection

PULL		Pin Behavior
PULLTYPESELECT	PULLUDENABLE	
0b0	0b0	Pull-down selected but not activated
0b0	0b1	Pull-down selected and activated if pin is NOT configured as OUTPUT
0b1	0b0	Pull-up selected but not activated
0b1	0b1	Pull-up selected and activated if pin is NOT configured as OUTPUT

For more information on the pull available on each pin, see Table 6-5 .

6.4.4.3 Pad Multiplexing Register Fields

Table 6-5 provide for each pad configuration register field the address offset and the associated signal name for each multiplexing mode (as set by bit field MUXMODE). Mode 0 is always defined. Modes with no signal name are undefined for the given pad.

NOTE: Table 6-5 lists the pad configuration registers instantiated in the CORE domain and WKUP domain that drive the pads in the CORE domain and WKUP domain respectively. An empty cell indicates that the mode or pull is not available for this pin.

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_SDRC_D0[15:0]	0x4800 2030	sdrc_d0							
CONTROL_PADCONF_SDRC_D0[31:16]	0x4800 2030	sdrc_d1							
CONTROL_PADCONF_SDRC_D2[15:0]	0x4800 2034	sdrc_d2							
CONTROL_PADCONF_SDRC_D2[31:16]	0x4800 2034	sdrc_d3							
CONTROL_PADCONF_SDRC_D4[15:0]	0x4800 2038	sdrc_d4							
CONTROL_PADCONF_SDRC_D4[31:16]	0x4800 2038	sdrc_d5							
CONTROL_PADCONF_SDRC_D6[15:0]	0x4800 203C	sdrc_d6							
CONTROL_PADCONF_SDRC_D6[31:16]	0x4800 203C	sdrc_d7							
CONTROL_PADCONF_SDRC_D8[15:0]	0x4800 2040	sdrc_d8							
CONTROL_PADCONF_SDRC_D8[31:16]	0x4800 2040	sdrc_d9							
CONTROL_PADCONF_SDRC_D10[15:0]	0x4800 2044	sdrc_d10							
CONTROL_PADCONF_SDRC_D10[31:16]	0x4800 2044	sdrc_d11							
CONTROL_PADCONF_SDRC_D12[15:0]	0x4800 2048	sdrc_d12							
CONTROL_PADCONF_SDRC_D12[31:16]	0x4800 2048	sdrc_d13							
CONTROL_PADCONF_SDRC_D14[15:0]	0x4800 204C	sdrc_d14							
CONTROL_PADCONF_SDRC_D14[31:16]	0x4800 204C	sdrc_d15							
CONTROL_PADCONF_SDRC_D16[15:0]	0x4800 2050	sdrc_d16							
CONTROL_PADCONF_SDRC_D16[31:16]	0x4800 2050	sdrc_d17							
CONTROL_PADCONF_SDRC_D18[15:0]	0x4800 2054	sdrc_d18							
CONTROL_PADCONF_SDRC_D18[31:16]	0x4800 2054	sdrc_d19							
CONTROL_PADCONF_SDRC_D20[15:0]	0x4800 2058	sdrc_d20							
CONTROL_PADCONF_SDRC_D20[31:16]	0x4800 2058	sdrc_d21							
CONTROL_PADCONF_SDRC_D22[15:0]	0x4800 205C	sdrc_d22							
CONTROL_PADCONF_SDRC_D22[31:16]	0x4800 205C	sdrc_d23							
CONTROL_PADCONF_SDRC_D24[15:0]	0x4800 2060	sdrc_d24							
CONTROL_PADCONF_SDRC_D24[31:16]	0x4800 2060	sdrc_d25							
CONTROL_PADCONF_SDRC_D26[15:0]	0x4800 2064	sdrc_d26							
CONTROL_PADCONF_SDRC_D26[31:16]	0x4800 2064	sdrc_d27							
CONTROL_PADCONF_SDRC_D28[15:0]	0x4800 2068	sdrc_d28							
CONTROL_PADCONF_SDRC_D28[31:16]	0x4800 2068	sdrc_d29							
CONTROL_PADCONF_SDRC_D30[15:0]	0x4800 206C	sdrc_d30							
CONTROL_PADCONF_SDRC_D30[31:16]	0x4800 206C	sdrc_d31							
CONTROL_PADCONF_SDRC_CLK[15:0]	0x4800 2070	sdrc_clk							
CONTROL_PADCONF_SDRC_CLK[31:16]	0x4800 2070	sdrc_dqs0p							
CONTROL_PADCONF_SDRC_CKE[31:16]	0x4800 2264	sdrc_cke0							sdrc_cke0_safe
CONTROL_PADCONF_SDRC_DQS1[15:0]	0x4800 2074	sdrc_dqs1p							

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_SDRC_DQS1[31:16]	0x4800 2074	sdrc_dqs2p							
CONTROL_PADCONF_SDRC_DQS3[15:0]	0x4800 2078	sdrc_dqs3p							
CONTROL_PADCONF_SYS_BOOT7[31:16]	0x4800 2218	sdrc_dqs0n							
CONTROL_PADCONF_SDRC_DQS1N[15:0]	0x4800 221C	sdrc_dqs1n							
CONTROL_PADCONF_SDRC_DQS1N[31:16]	0x4800 221C	sdrc_dqs2n							
CONTROL_PADCONF_SDRC_DQS3N[15:0]	0x4800 2220	sdrc_dqs3n							
CONTROL_PADCONF_SDRC_DQS3N[31:16]	0x4800 2220	sdrc_strben_dly0							
CONTROL_PADCONF_SDRC_STRBEN_DLY1[15:0]	0x4800 2224	sdrc_strben_dly1							
CONTROL_PADCONF_SDRC_DQS3[31:16]	0x4800 2078	gpmc_a1				gpio_34			
CONTROL_PADCONF_GPMC_A2[15:0]	0x4800 207C	gpmc_a2				gpio_35			
CONTROL_PADCONF_GPMC_A2[31:16]	0x4800 207C	gpmc_a3				gpio_36			
CONTROL_PADCONF_GPMC_A4[15:0]	0x4800 2080	gpmc_a4				gpio_37			
CONTROL_PADCONF_GPMC_A4[31:16]	0x4800 2080	gpmc_a5				gpio_38			
CONTROL_PADCONF_GPMC_A6[15:0]	0x4800 2084	gpmc_a6				gpio_39			
CONTROL_PADCONF_GPMC_A6[31:16]	0x4800 2084	gpmc_a7				gpio_40			
CONTROL_PADCONF_GPMC_A8[15:0]	0x4800 2088	gpmc_a8				gpio_41			
CONTROL_PADCONF_GPMC_A8[31:16]	0x4800 2088	gpmc_a9	SYS_NDMAREQ 2			gpio_42			
CONTROL_PADCONF_GPMC_A10[15:0]	0x4800 208C	gpmc_a10	SYS_NDMAREQ 3			gpio_43			
CONTROL_PADCONF_GPMC_A10[31:16]	0x4800 208C	gpmc_d0							
CONTROL_PADCONF_GPMC_D1[15:0]	0x4800 2090	gpmc_d1							
CONTROL_PADCONF_GPMC_D1[31:16]	0x4800 2090	gpmc_d2							
CONTROL_PADCONF_GPMC_D3[15:0]	0x4800 2094	gpmc_d3							
CONTROL_PADCONF_GPMC_D3[31:16]	0x4800 2094	gpmc_d4							
CONTROL_PADCONF_GPMC_D5[15:0]	0x4800 2098	gpmc_d5							
CONTROL_PADCONF_GPMC_D5[31:16]	0x4800 2098	gpmc_d6							
CONTROL_PADCONF_GPMC_D7[15:0]	0x4800 209C	gpmc_d7							
CONTROL_PADCONF_GPMC_D7[31:16]	0x4800 209C	gpmc_d8				gpio_44			
CONTROL_PADCONF_GPMC_D9[15:0]	0x4800 20A0	gpmc_d9				gpio_45			
CONTROL_PADCONF_GPMC_D9[31:16]	0x4800 20A0	gpmc_d10				gpio_46			
CONTROL_PADCONF_GPMC_D11[15:0]	0x4800 20A4	gpmc_d11				gpio_47			
CONTROL_PADCONF_GPMC_D11[31:16]	0x4800 20A4	gpmc_d12				gpio_48			
CONTROL_PADCONF_GPMC_D13[15:0]	0x4800 20A8	gpmc_d13				gpio_49			
CONTROL_PADCONF_GPMC_D13[31:16]	0x4800 20A8	gpmc_d14				gpio_50			
CONTROL_PADCONF_GPMC_D15[15:0]	0x4800 20AC	gpmc_d15				gpio_51			
CONTROL_PADCONF_GPMC_D15[31:16]	0x4800 20AC	gpmc_ncs0							

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_GPMC_NCS1[15:0]	0x4800 20B0	gpmc_ncs1				gpio_52			
CONTROL_PADCONF_GPMC_NCS1[31:16]	0x4800 20B0	gpmc_ncs2		gpt9_pwm_evt		gpio_53			
CONTROL_PADCONF_GPMC_NCS3[15:0]	0x4800 20B4	gpmc_ncs3	SYS_NDMAREQ 0	gpt10_pwm_evt		gpio_54			
CONTROL_PADCONF_GPMC_NCS3[31:16]	0x4800 20B4	gpmc_ncs4	SYS_NDMAREQ 1		gpt9_pwm_evt	gpio_55			
CONTROL_PADCONF_GPMC_NCS5[15:0]	0x4800 20B8	gpmc_ncs5	SYS_NDMAREQ 2		gpt10_pwm_evt	gpio_56			
CONTROL_PADCONF_GPMC_NCS5[31:16]	0x4800 20B8	gpmc_ncs6	SYS_NDMAREQ 3		gpt11_pwm_evt	gpio_57			
CONTROL_PADCONF_GPMC_NCS7[15:0]	0x4800 20BC	gpmc_ncs7	GPMC_IO_DIR		gpt8_pwm_evt	gpio_58			
CONTROL_PADCONF_GPMC_NCS7[31:16]	0x4800 20BC	gpmc_clk				gpio_59			
CONTROL_PADCONF_GPMC_NADV_ALE[15:0]	0x4800 20C0	gpmc_nadv_ale							
CONTROL_PADCONF_GPMC_NADV_ALE[31:16]	0x4800 20C0	gpmc_noe							
CONTROL_PADCONF_GPMC_NWE[15:0]	0x4800 20C4	gpmc_nwe							
CONTROL_PADCONF_GPMC_NWE[31:16]	0x4800 20C4	gpmc_nbe0_cle				gpio_60			
CONTROL_PADCONF_GPMC_NBE1[15:0]	0x4800 20C8	gpmc_nbe1				gpio_61			
CONTROL_PADCONF_GPMC_NBE1[31:16]	0x4800 20C8	gpmc_nwp				gpio_62			
CONTROL_PADCONF_GPMC_WAIT0[15:0]	0x4800 20CC	gpmc_wait0							
CONTROL_PADCONF_GPMC_WAIT0[31:16]	0x4800 20CC	gpmc_wait1	UART4_TX			gpio_63			
CONTROL_PADCONF_GPMC_WAIT2[15:0]	0x4800 20D0	gpmc_wait2	UART4_RX			gpio_64			
CONTROL_PADCONF_GPMC_WAIT2[31:16]	0x4800 20D0	gpmc_wait3	SYS_NDMAREQ 1	uart3_cts_rctx		gpio_65			
CONTROL_PADCONF_DSS_PCLK[15:0]	0x4800 20D4	dss_pclk				gpio_66	hw_dbg12		
CONTROL_PADCONF_DSS_PCLK[31:16]	0x4800 20D4	dss_hsync				gpio_67	hw_dbg13		
CONTROL_PADCONF_DSS_VSYNC[15:0]	0x4800 20D8	dss_vsync				gpio_68			
CONTROL_PADCONF_DSS_VSYNC[31:16]	0x4800 20D8	dss_acbias				gpio_69			
CONTROL_PADCONF_DSS_DATA0[15:0]	0x4800 20DC	dss_data0		uart1_cts	dssvenc656_data 0	gpio_70			
CONTROL_PADCONF_DSS_DATA0[31:16]	0x4800 20DC	dss_data1		uart1_rts	dssvenc656_data 1	gpio_71			
CONTROL_PADCONF_DSS_DATA2[15:0]	0x4800 20E0	dss_data2			dssvenc656_data 2	gpio_72			
CONTROL_PADCONF_DSS_DATA2[31:16]	0x4800 20E0	dss_data3			dssvenc656_data 3	gpio_73			
CONTROL_PADCONF_DSS_DATA4[15:0]	0x4800 20E4	dss_data4		uart3_rx_irrx	dssvenc656_data 4	gpio_74			
CONTROL_PADCONF_DSS_DATA4[31:16]	0x4800 20E4	dss_data5		uart3_tx_irtx	dssvenc656_data 5	gpio_75			

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_DSS_DATA6[15:0]	0x4800 20E8	dss_data6		uart1_tx	dssvenc656_data6	gpio_76	hw_dbg14		
CONTROL_PADCONF_DSS_DATA6[31:16]	0x4800 20E8	dss_data7		uart1_rx	dssvenc656_data7	gpio_77	hw_dbg15		
CONTROL_PADCONF_DSS_DATA8[15:0]	0x4800 20EC	dss_data8				gpio_78	hw_dbg16		
CONTROL_PADCONF_DSS_DATA8[31:16]	0x4800 20EC	dss_data9				gpio_79	hw_dbg17		
CONTROL_PADCONF_DSS_DATA10[15:0]	0x4800 20F0	dss_data10				gpio_80			
CONTROL_PADCONF_DSS_DATA10[31:16]	0x4800 20F0	dss_data11				gpio_81			
CONTROL_PADCONF_DSS_DATA12[15:0]	0x4800 20F4	dss_data12				gpio_82			
CONTROL_PADCONF_DSS_DATA12[31:16]	0x4800 20F4	dss_data13				gpio_83			
CONTROL_PADCONF_DSS_DATA14[15:0]	0x4800 20F8	dss_data14				gpio_84			
CONTROL_PADCONF_DSS_DATA14[31:16]	0x4800 20F8	dss_data15				gpio_85			
CONTROL_PADCONF_DSS_DATA16[15:0]	0x4800 20FC	dss_data16				gpio_86			
CONTROL_PADCONF_DSS_DATA16[31:16]	0x4800 20FC	dss_data17				gpio_87			
CONTROL_PADCONF_DSS_DATA18[15:0]	0x4800 2100	dss_data18		mcspi3_clk	dss_data4	gpio_88			
CONTROL_PADCONF_DSS_DATA18[31:16]	0x4800 2100	dss_data19		mcspi3_simo	dss_data3	gpio_89			
CONTROL_PADCONF_DSS_DATA20[15:0]	0x4800 2104	dss_data20		mcspi3_somi	dss_data2	gpio_90			
CONTROL_PADCONF_DSS_DATA20[31:16]	0x4800 2104	dss_data21		mcspi3_cs0	dss_data1	gpio_91			
CONTROL_PADCONF_DSS_DATA22[15:0]	0x4800 2108	dss_data22		mcspi3_cs1	dss_data0	gpio_92			
CONTROL_PADCONF_DSS_DATA22[31:16]	0x4800 2108	dss_data23			dss_data5	gpio_93			
CONTROL_PADCONF_CCDC_PCLK[15:0]	0x4800 21E4	ccdc_pclk				gpio_94	hw_dbg0		
CONTROL_PADCONF_CCDC_PCLK[31:16]	0x4800 21E4	ccdc_field	CCDC_DATA8	uart4_tx	i2c3_scl	gpio_95	hw_dbg1		
CONTROL_PADCONF_CCDC_HD[15:0]	0x4800 21E8	ccdc_hd		uart4_rts		gpio_96			
CONTROL_PADCONF_CCDC_HD[31:16]	0x4800 21E8	ccdc_vd		uart4_cts		gpio_97	hw_dbg2		
CONTROL_PADCONF_CCDC_WEN[15:0]	0x4800 21EC	ccdc_wen	CCDC_DATA9	uart4_rx		gpio_98	hw_dbg3		
CONTROL_PADCONF_CCDC_WEN[31:16]	0x4800 21EC	ccdc_data0			i2c3_sda	gpio_99			
CONTROL_PADCONF_CCDC_DATA1[15:0]	0x4800 21F0	ccdc_data1				gpio_100			
CONTROL_PADCONF_CCDC_DATA1[31:16]	0x4800 21F0	ccdc_data2				gpio_101	hw_dbg4		
CONTROL_PADCONF_CCDC_DATA3[15:0]	0x4800 21F4	ccdc_data3				gpio_102	hw_dbg5		
CONTROL_PADCONF_CCDC_DATA3[31:16]	0x4800 21F4	ccdc_data4				gpio_103	hw_dbg6		
CONTROL_PADCONF_CCDC_DATA5[15:0]	0x4800 21F8	ccdc_data5				gpio_104	hw_dbg7		
CONTROL_PADCONF_CCDC_DATA5[31:16]	0x4800 21F8	ccdc_data6				gpio_105			
CONTROL_PADCONF_CCDC_DATA7[15:0]	0x4800 21FC	ccdc_data7				gpio_106			
CONTROL_PADCONF_CCDC_DATA7[31:16]	0x4800 21FC	rmii_mdio_data	CCDC_DATA8			gpio_107			
CONTROL_PADCONF_RMII_MDIO_CLK[15:0]	0x4800 2200	rmii_mdio_clk	CCDC_DATA9			gpio_108			
CONTROL_PADCONF_RMII_MDIO_CLK[31:16]	0x4800 2200	rmii_rxd0	CCDC_DATA10			gpio_109	hw_dbg8		
CONTROL_PADCONF_RMII_RXD1[15:0]	0x4800 2204	rmii_rxd1	CCDC_DATA11			gpio_110	hw_dbg9		

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_RMII_RXD1[31:16]	0x4800 2204	rmii_crs_dv	CCDC_DATA12			gpio_111			
CONTROL_PADCONF_RMII_RXER[15:0]	0x4800 2208	rmii_rxer	CCDC_DATA13			gpio_167	hw_dbg10		
CONTROL_PADCONF_RMII_RXER[31:16]	0x4800 2208	rmii_txd0	CCDC_DATA14			gpio_126	hw_dbg11		
CONTROL_PADCONF_RMII_TXD1[15:0]	0x4800 220C	rmii_txd1	CCDC_DATA15			gpio_112			
CONTROL_PADCONF_RMII_TXD1[31:16]	0x4800 220C	rmii_txen				gpio_113			
CONTROL_PADCONF_RMII_50MHZ_CLK[15:0]	0x4800 2210	rmii_50mhz_clk				gpio_114			
CONTROL_PADCONF_MCBSP2_FSX[15:0]	0x4800 213C	mcbsp2_fsx				gpio_116			
CONTROL_PADCONF_MCBSP2_FSX[31:16]	0x4800 213C	mcbsp2_clkx				gpio_117			
CONTROL_PADCONF_MCBSP2_DR[15:0]	0x4800 2140	mcbsp2_dr				gpio_118			
CONTROL_PADCONF_MCBSP2_DR[31:16]	0x4800 2140	mcbsp2_dx				gpio_119			
CONTROL_PADCONF_MMC1_CLK[15:0]	0x4800 2144	mmc1_clk				gpio_120			
CONTROL_PADCONF_MMC1_CLK[31:16]	0x4800 2144	mmc1_cmd				gpio_121			
CONTROL_PADCONF_MMC1_DAT0[15:0]	0x4800 2148	mmc1_dat0	MCSPI2_CLK			gpio_122			
CONTROL_PADCONF_MMC1_DAT0[31:16]	0x4800 2148	mmc1_dat1	MCSPI2_SIMO			gpio_123			
CONTROL_PADCONF_MMC1_DAT2[15:0]	0x4800 214C	mmc1_dat2	MCSPI2_SOMI			gpio_124			
CONTROL_PADCONF_MMC1_DAT2[31:16]	0x4800 214C	mmc1_dat3	MCSPI2_CS0			gpio_125			
CONTROL_PADCONF_MMC1_DAT4[15:0]	0x4800 2150	mmc1_dat4				gpio_126			
CONTROL_PADCONF_MMC1_DAT4[31:16]	0x4800 2150	mmc1_dat5				gpio_127			
CONTROL_PADCONF_MMC1_DAT6[15:0]	0x4800 2154	mmc1_dat6				gpio_128			
CONTROL_PADCONF_MMC1_DAT6[31:16]	0x4800 2154	mmc1_dat7				gpio_129			
CONTROL_PADCONF_MMC2_CLK[15:0]	0x4800 2158	mmc2_clk	MCSPI3_CLK	uart4_cts		gpio_130			
CONTROL_PADCONF_MMC2_CLK[31:16]	0x4800 2158	mmc2_cmd	MCSPI3_SIMO	uart4_rts		gpio_131			
CONTROL_PADCONF_MMC2_DAT0[15:0]	0x4800 215C	mmc2_dat0	MCSPI3_SOMI	uart4_tx		gpio_132			
CONTROL_PADCONF_MMC2_DAT0[31:16]	0x4800 215C	mmc2_dat1		uart4_rx		gpio_133			
CONTROL_PADCONF_MMC2_DAT2[15:0]	0x4800 2160	mmc2_dat2	mcspi3_cs1			gpio_134			
CONTROL_PADCONF_MMC2_DAT2[31:16]	0x4800 2160	mmc2_dat3	mcspi3_cs0			gpio_135			
CONTROL_PADCONF_MMC2_DAT4[15:0]	0x4800 2164	mmc2_dat4	mmc2_dir_dat0		mmc3_dat0	gpio_136		mm_fsusb3_rxdp	
CONTROL_PADCONF_MMC2_DAT4[31:16]	0x4800 2164	mmc2_dat5	mmc2_dir_dat1		mmc3_dat1	gpio_137			
CONTROL_PADCONF_MMC2_DAT6[15:0]	0x4800 2168	mmc2_dat6	mmc2_dir_cmd		mmc3_dat2	gpio_138		mm_fsusb3_rxdm	
CONTROL_PADCONF_MMC2_DAT6[31:16]	0x4800 2168	mmc2_dat7	mmc2_clkin		mmc3_dat3	gpio_139			
CONTROL_PADCONF_MCBSP3_DX[15:0]	0x4800 216C	mcbsp3_dx	uart2_cts			gpio_140			
CONTROL_PADCONF_MCBSP3_DX[31:16]	0x4800 216C	mcbsp3_dr	uart2_rts			gpio_141			
CONTROL_PADCONF_MCBSP3_CLKX[15:0]	0x4800 2170	mcbsp3_clkx	uart2_tx			gpio_142			
CONTROL_PADCONF_MCBSP3_CLKX[31:16]	0x4800 2170	mcbsp3_fsx	uart2_rx			gpio_143			
CONTROL_PADCONF_UART2_CTS[15:0]	0x4800 2174	uart2_cts	mcbsp3_dx	gpt9_pwm_evt		gpio_144			

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_UART2_CTS[31:16]	0x4800 2174	uart2_rts	mcbsp3_dr	gpt10_pwm_evt		gpio_145			
CONTROL_PADCONF_UART2_TX[15:0]	0x4800 2178	uart2_tx	mcbsp3_clkx	gpt11_pwm_evt		gpio_146			
CONTROL_PADCONF_UART2_TX[31:16]	0x4800 2178	uart2_rx	mcbsp3_fsx	gpt8_pwm_evt		gpio_147			
CONTROL_PADCONF_UART1_TX[15:0]	0x4800 217C	uart1_tx				gpio_148			
CONTROL_PADCONF_UART1_TX[31:16]	0x4800 217C	uart1_rts				gpio_149			
CONTROL_PADCONF_UART1_CTS[15:0]	0x4800 2180	uart1_cts				gpio_150			
CONTROL_PADCONF_UART1_CTS[31:16]	0x4800 2180	uart1_rx		mcbsp1_clkr	mcspi4_clk	gpio_151		mm_fsusb3_txse0	
CONTROL_PADCONF_MCBSP4_CLKX[15:0]	0x4800 2184	mcbsp4_clkx				gpio_152		mm_fsusb3_rxrcv	
CONTROL_PADCONF_MCBSP4_CLKX[31:16]	0x4800 2184	mcbsp4_dr				gpio_153		mm_fsusb3_txdat	
CONTROL_PADCONF_MCBSP4_DX[15:0]	0x4800 2188	mcbsp4_dx				gpio_154		mm_fsusb3_txen_n	
CONTROL_PADCONF_MCBSP4_DX[31:16]	0x4800 2188	mcbsp4_fsx				gpio_155			
CONTROL_PADCONF_MCBSP1_CLKR[15:0]	0x4800 218C	mcbsp1_clkr	mcspi4_clk			gpio_156			
CONTROL_PADCONF_MCBSP1_CLKR[31:16]	0x4800 218C	mcbsp1_fsr	adpllv2d_ditherin g_en1			gpio_157			
CONTROL_PADCONF_MCBSP1_DX[15:0]	0x4800 2190	mcbsp1_dx	mcspi4_simo	mcbsp3_dx		gpio_158			
CONTROL_PADCONF_MCBSP1_DX[31:16]	0x4800 2190	mcbsp1_dr	mcspi4_somi	mcbsp3_dr		gpio_159			
CONTROL_PADCONF_MCBSP_CLKS[15:0]	0x4800 2194	mcbsp_clks				gpio_160	uart1_cts		
CONTROL_PADCONF_MCBSP_CLKS[31:16]	0x4800 2194	mcbsp1_fsx	mcspi4_cs0	mcbsp3_fsx		gpio_161			
CONTROL_PADCONF_MCBSP1_CLKX[15:0]	0x4800 2198	mcbsp1_clkx		mcbsp3_clkx		gpio_162			
CONTROL_PADCONF_MCBSP1_CLKX[31:16]	0x4800 2198	uart3_cts_rctx				gpio_163			
CONTROL_PADCONF_UART3_RTS_SD[15:0]	0x4800 219C	uart3_rts_sd				gpio_164			
CONTROL_PADCONF_UART3_RTS_SD[31:16]	0x4800 219C	uart3_rx_irrx				gpio_165			
CONTROL_PADCONF_UART3_TX_IRTX[15:0]	0x4800 21A0	uart3_tx_irtx				gpio_166			
CONTROL_PADCONF_RMII_50MHZ_CLK[31:16]	0x4800 2210	usb0_drvvbus		uart3_tx_irtx		gpio_125			
CONTROL_PADCONF_HECC1_TXD[15:0]	0x4800 2214	hecc1_txd		uart3_rx_irrx		gpio_130			
CONTROL_PADCONF_HECC1_RXD[31:16]	0x4800 2214	hecc1_rxd		uart3_rts_sd		gpio_131			
CONTROL_PADCONF_I2C1_SCL[31:15]	0x4800 21B8	i2c1_scl							
CONTROL_PADCONF_I2C1_SDA[15:0]	0x4800 21BC	i2c1_sda							
CONTROL_PADCONF_I2C1_SDA[31:16]	0x4800 21BC	i2c2_scl				gpio_168			
CONTROL_PADCONF_I2C2_SDA[15:0]	0x4800 21C0	i2c2_sda				gpio_183			
CONTROL_PADCONF_I2C2_SDA[31:16]	0x4800 21C0	i2c3_scl				gpio_184			
CONTROL_PADCONF_I2C3_SDA[15:0]	0x4800 21C4	i2c3_sda				gpio_185			
CONTROL_PADCONF_I2C3_SDA[31:16]	0x4800 21C4	hdq_sio	sys_altclk	i2c2_sccbe	i2c3_sccbe	gpio_170			
CONTROL_PADCONF_MCSPI1_CLK[15:0]	0x4800 21C8	mcspi1_clk	mmc2_dat4			gpio_171			

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_MCSPI1_CLK[31:16]	0x4800 21C8	mcspi1_simo	mmc2_dat5			gpio_172			
CONTROL_PADCONF_MCSPI1_SOMI[15:0]	0x4800 21CC	mcspi1_somi	mmc2_dat6			gpio_173			
CONTROL_PADCONF_MCSPI1_SOMI[31:16]	0x4800 21CC	mcspi1_cs0	mmc2_dat7			gpio_174			
CONTROL_PADCONF_MCSPI1_CS1[15:0]	0x4800 21D0	mcspi1_cs1	adpllv2d_dithering_en2		mmc3_cmd	gpio_175			
CONTROL_PADCONF_MCSPI1_CS1[31:16]	0x4800 21D0	mcspi1_cs2			mmc3_clk	gpio_176			
CONTROL_PADCONF_MCSPI1_CS3[15:0]	0x4800 21D4	mcspi1_cs3		hsusb2_tll_data2	hsusb2_data2	gpio_177	mm_fsusb2_txdat		
CONTROL_PADCONF_MCSPI1_CS3[31:16]	0x4800 21D4	mcspi2_clk		hsusb2_tll_data7	hsusb2_data7	gpio_178			
CONTROL_PADCONF_MCSPI2_SIMO[15:0]	0x4800 21D8	mcspi2_simo	gpt9_pwm_evt	hsusb2_tll_data4	hsusb2_data4	gpio_179			
CONTROL_PADCONF_MCSPI2_SIMO[31:16]	0x4800 21D8	mcspi2_somi	gpt10_pwm_evt	hsusb2_tll_data5	hsusb2_data5	gpio_180			
CONTROL_PADCONF_MCSPI2_CS0[15:0]	0x4800 21DC	mcspi2_cs0	gpt11_pwm_evt	hsusb2_tll_data6	hsusb2_data6	gpio_181			
CONTROL_PADCONF_MCSPI2_CS0[31:16]	0x4800 21DC	mcspi2_cs1	gpt8_pwm_evt	hsusb2_tll_data3	hsusb2_data3	gpio_182	mm_fsusb2_txen_n		
CONTROL_PADCONF_SYS_NIRQ[15:0]	0x4800 21E0	sys_nirq				gpio_0			
CONTROL_PADCONF_SYS_NIRQ[31:16]	0x4800 21E0	sys_clkout2				gpio_186			hw_dbg0
CONTROL_PADCONF_ETK_CLK[15:0]	0x4800 25D8	etk_clk	mcbsp5_clkx	mmc3_clk	hsusb1_stp	gpio_12			hw_dbg1
CONTROL_PADCONF_ETK_CLK[31:16]	0x4800 25D8	etk_ctl		mmc3_cmd	hsusb1_clk	gpio_13	mm_fsusb1_rxdp		hw_dbg2
CONTROL_PADCONF_ETK_D0[15:0]	0x4800 25DC	etk_d0	mcspi3_simo	mmc3_dat4	hsusb1_data0	gpio_14	mm_fsusb1_rxrcv		hw_dbg3
CONTROL_PADCONF_ETK_D0[31:16]	0x4800 25DC	etk_d1	mcspi3_somi		hsusb1_data1	gpio_15	mm_fsusb1_txse0		hw_dbg4
CONTROL_PADCONF_ETK_D2[15:0]	0x4800 25E0	etk_d2	mcspi3_cs0		hsusb1_data2	gpio_16	mm_fsusb1_txdat		hw_dbg5
CONTROL_PADCONF_ETK_D2[31:16]	0x4800 25E0	etk_d3	mcspi3_clk	mmc3_dat3	hsusb1_data7	gpio_17			hw_dbg6
CONTROL_PADCONF_ETK_D4[15:0]	0x4800 25E4	etk_d4	mcbsp5_dr	mmc3_dat0	hsusb1_data4	gpio_18			hw_dbg7
CONTROL_PADCONF_ETK_D4[31:16]	0x4800 25E4	etk_d5	mcbsp5_fsx	mmc3_dat1	hsusb1_data5	gpio_19			hw_dbg8
CONTROL_PADCONF_ETK_D6[15:0]	0x4800 25E8	etk_d6	mcbsp5_dx	mmc3_dat2	hsusb1_data6	gpio_20			hw_dbg9
CONTROL_PADCONF_ETK_D6[31:16]	0x4800 25E8	etk_d7	mcspi3_cs1	mmc3_dat7	hsusb1_data3	gpio_21	mm_fsusb1_txen_n		hw_dbg10
CONTROL_PADCONF_ETK_D8[15:0]	0x4800 25EC	etk_d8	sys_drm_msecure	mmc3_dat6	hsusb1_dir	gpio_22			hw_dbg11
CONTROL_PADCONF_ETK_D8[31:16]	0x4800 25EC	etk_d9	sys_secure_indicator	mmc3_dat5	hsusb1_nxt	gpio_23	mm_fsusb1_rxdm		hw_dbg12
CONTROL_PADCONF_ETK_D10[15:0]	0x4800 25F0	etk_d10		uart1_rx	hsusb2_clk	gpio_24			hw_dbg13
CONTROL_PADCONF_ETK_D10[31:16]	0x4800 25F0	etk_d11	mcspi3_clk		hsusb2_stp	gpio_25	mm_fsusb2_rxdp		hw_dbg14
CONTROL_PADCONF_ETK_D12[15:0]	0x4800 25F4	etk_d12			hsusb2_dir	gpio_26			hw_dbg15

Table 6-5. Core and Wakeup Control Module Pad Configuration Registers (continued)

Register Name	Physical Address	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
CONTROL_PADCONF_ETK_D12[31:16]	0x4800 25F4	etk_d13			hsusb2_nxt	gpio_27	mm_fsusb2_rxdm		hw_dbg16
CONTROL_PADCONF_ETK_D14[15:0]	0x4800 25F8	etk_d14			hsusb2_data0	gpio_28	mm_fsusb2_rxrv		hw_dbg17
CONTROL_PADCONF_ETK_D14[31:16]	0x4800 25F8	etk_d15			hsusb2_data1	gpio_29	mm_fsusb2_txse0		
CONTROL_PADCONF_SYS_32K[15:0]	0x4800 2A04	sys_32k							
CONTROL_PADCONF_SYS_32K[31:16]	0x4800 2A04	sys_clkreq				gpio_1			
CONTROL_PADCONF_SYS_NRESWARM[15:0]	0x4800 2A08	sys_nreswarm				gpio_30			
CONTROL_PADCONF_SYS_NRESWARM[31:16]	0x4800 2A08	sys_boot0				gpio_2			
CONTROL_PADCONF_SYS_BOOT1[15:0]	0x4800 2A0C	sys_boot1				gpio_3			
CONTROL_PADCONF_SYS_BOOT1[31:16]	0x4800 2A0C	sys_boot2				gpio_4			
CONTROL_PADCONF_SYS_BOOT3[15:0]	0x4800 2A10	sys_boot3				gpio_5			
CONTROL_PADCONF_SYS_BOOT3[31:16]	0x4800 2A10	sys_boot4	mmc2_dir_dat2			gpio_6			
CONTROL_PADCONF_SYS_BOOT5[15:0]	0x4800 2A14	sys_boot5	mmc2_dir_dat3			gpio_7			
CONTROL_PADCONF_SYS_BOOT5[31:16]	0x4800 2A14	sys_boot6				gpio_8			
CONTROL_PADCONF_SYS_BOOT7[15:0]	0x4800 2218	sys_boot7							
CONTROL_PADCONF_SDRC_STRBEN_DLY1[31:16]	0x4800 2224	sys_boot8							
CONTROL_PADCONF_SYS_CLKOUT1[31:16]	0x4800 2A18	sys_clkout1				gpio_10			
CONTROL_PADCONF_JTAG_NTRST[15:0]	0x4800 2A1C	jtag_ntrst							
CONTROL_PADCONF_JTAG_NTRST[31:16]	0x4800 2A1C	jtag_tck							
CONTROL_PADCONF_JTAG_RTCK[31:16]	0x4800 2A4C	jtag_rtck							
CONTROL_PADCONF_JTAG_TMS_TMSC[15:0]	0x4800 2A20	jtag_tms_tmsc							
CONTROL_PADCONF_JTAG_TMS_TMSC[31:16]	0x4800 2A20	jtag_tdi							
CONTROL_PADCONF_JTAG_TDO[15:0]	0x4800 2A50	jtag_tdo							
CONTROL_PADCONF_JTAG_EMU0[15:0]	0x4800 2A24	jtag_emu0				gpio_11			
CONTROL_PADCONF_JTAG_EMU0[31:16]	0x4800 2A24	jtag_emu1				gpio_31			

NOTE: Pad names are signal names available in mode 0.

For more information on the pad default states, please refer to your device-specific data manual.

6.4.5 Functional Register Description

6.4.5.1 Static Device Configuration Registers

Table 6-6 describes the static device configuration registers.

Table 6-6. Static Device Configuration Registers

Physical Address	Register Name	Description	Access
0x4800 2274	CONTROL_DEVCONF0	Module dedicated configurations	R/W
0x4800 22D8	CONTROL_DEVCONF1	Module dedicated configurations	R/W
0x4800 2580	CONTROL_DEVCONF2	Module dedicated configurations	R/W
0x4800 2584	CONTROL_DEVCONF3	Module dedicated configurations	R/W

These registers allow the static configuration of device modules such as USB, McBSP, and I^2C. For example, they allow selecting external or internal clocks for McBSP modules.

6.4.5.2 MPU MSuspend Configuration Registers

Table 6-7 describes the MPU MSuspend configuration registers.

Table 6-7. MSuspendMux Control Registers

Physical Address	Register Name	Description	Access
0x4800 2290	CONTROL_MSUSPENDMUX_0	Control the use of MSuspend signals at module level.	R/W
0x4800 2294	CONTROL_MSUSPENDMUX_1	Control the use of MSuspend signals at module level.	R/W
0x4800 2298	CONTROL_MSUSPENDMUX_2	Control the use of MSuspend signals at module level.	R/W
0x4800 22A0	CONTROL_MSUSPENDMUX_4	Control the use of MSuspend signals at module level.	R/W
0x4800 22A4	CONTROL_MSUSPENDMUX_5	Control the use of MSuspend signals at module level.	R/W
0x4800 22A8	CONTROL_MSUSPENDMUX_6	Control the use of MSuspend signals at module level.	R/W

These registers provide an entry for each module that must consider the MSuspend signals from the MPU processor . For each module, the sensitivity to the MSuspend signals is defined within possibilities (coded using 3 bits):

- 0b000: No sensitivity; no MSuspend signal reaches the module.
- 0b001: Sensitivity to the MPU MSuspend signal
- Other values: Reserved; other values are not supported.

The logic used to combine the MSuspend signals from the processor is implemented within the SCM. MSuspend signals are active low.

CAUTION

Not all modules use the MSUSPEND signal. See the TRM chapter for each module to determine whether the module supports the MSUSPEND signal.

All MSUSPEND signals coming out of the MPU and DSP are resynchronized within the SCM by using the control module interface clock.

6.4.5.3 Device Status Registers

Table 6-8 lists the status registers.

Table 6-8. Device Status Registers

Physical Address	Register Name	Description	Access
0x4800 22F0	CONTROL_STATUS	Device status register	Public privilege (R)

The CONTROL.CONTROL_STATUS register is read-only and has no access restriction.

The CONTROL_STATUS register bits[5:0] logs the status of sys_boot [5:0] pins at Power on Reset.

The lower three sys_boot[4:0] pins allow the selection of the booting sequence of interfaces or devices for ROM code when booting. Sys_boot[5] switches between memory (0) and peripheral (1) booting.

For further information on the configuration of the sys_boot pins, see the *Initialization* chapter.

After booting, these pins can be used optionally for other functions. When the pins are reused, the associated sysboot registers are not modified by the new functionality. It is recommended that the output function on the sys_boot pins be reused.

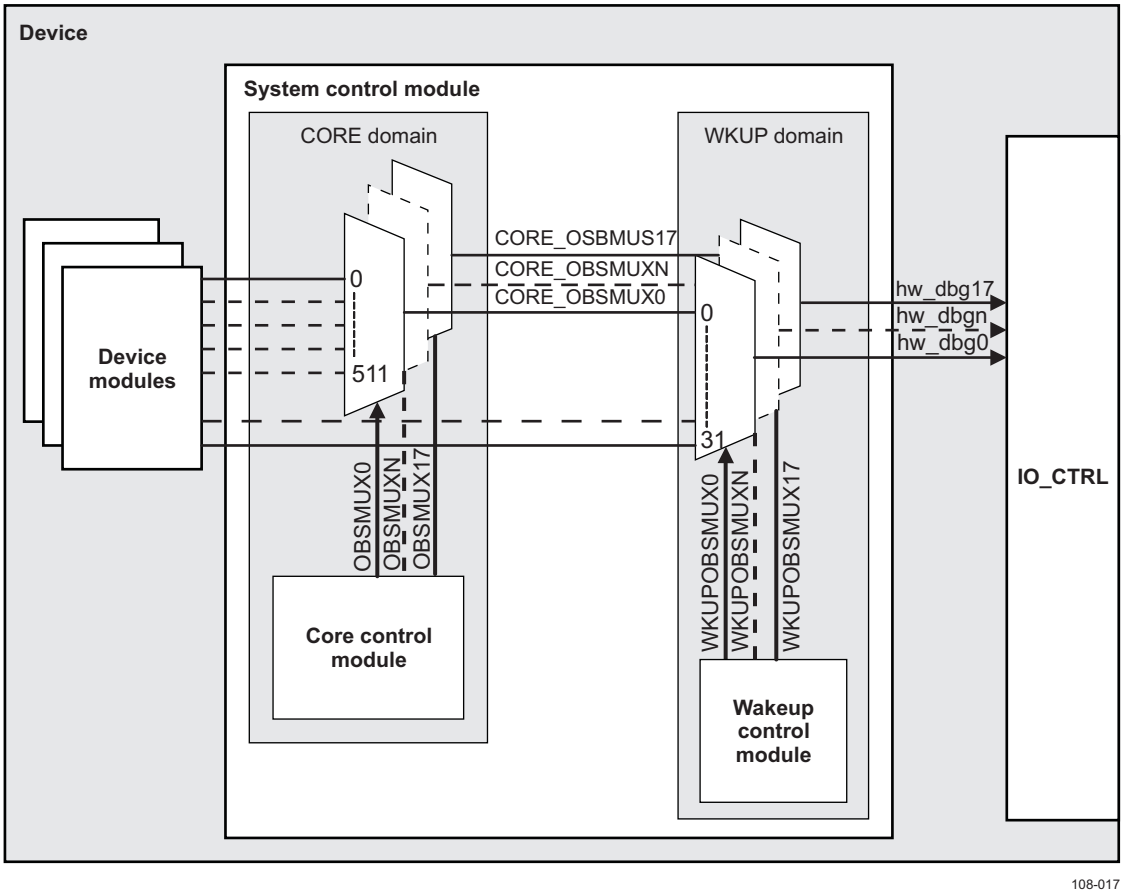
6.4.6 Debug and Observability

6.4.6.1 Description

NOTE: This feature is restricted to emulator and GP devices.

Figure 6-9 shows an overview of observability multiplexing, which minimizes the number of signals exchanged at the domain boundary.

Figure 6-9. Overview of the Debug and Observability Register Functionality



Two layers of multiplexer are used to select the set of internal observable signals (PRCM signals, DMA requests, and interrupts) to be routed to the pins dedicated to the hardware debug.

The first layer is in the CORE domain. It is controlled by the core control module registers and selects the set of internal signals from the CORE domain to be routed. The second layer is in the WKUP domain. It is controlled by the wake-up control module registers and selects the set of internal signals from the WKUP domain.

The pads used for the hardware debug must be properly configured by selecting the hardware debug function (hw_dbgn) of the pad. To configure the pads, select mode 5 (0b101) in the MUXMODE bit field of the CONTROL.CONTROL_PADCONF_ CCDC _x register (only for hw_dbg0 to hw_dbg11), or select mode 7 (0b111) in the MUXMODE bit field of the CONTROL.CONTROL_PADCONF_ETK_x register (for all hw_dbgn).

Before selecting the CORE signals, the WKUPOBSMUX field of the CONTROL.CONTROL_WKUP_DEBOBS_n registers must be set to 0.

NOTE: The pads used for the hardware debug must be properly configured by selecting the hardware debug function (hw_dbgn) of the pad. To configure the pads, select mode 5 (0b101) in the MUXMODE bit field of the CONTROL.CONTROL_PADCONF_ CCDC _x register (only for hw_dbg0 to hw_dbg11), or select mode 7 (0b111) in the MUXMODE bit field of the CONTROL.CONTROL_PADCONF_ETK_x register (for all hw_dbgn).

Table 6-9. Observability Registers

Physical Address	Register Name	Description	Access	
			Device Type	
			E/T/G	S/B
0x4800 2420	CONTROL_DEBOBS_0	Set and configure CORE observable signals 1 and 0	R/W	R
0x4800 2424	CONTROL_DEBOBS_1	Set and configure CORE observable signals 3 and 2	R/W	R
0x4800 2428	CONTROL_DEBOBS_2	Set and configure CORE observable signals 5 and 4	R/W	R
0x4800 242C	CONTROL_DEBOBS_3	Set and configure CORE observable signals 7 and 6	R/W	R
0x4800 2430	CONTROL_DEBOBS_4	Set and configure CORE observable signals 9 and 8	R/W	R
0x4800 2434	CONTROL_DEBOBS_5	Set and configure CORE observable signals 11 and 10	R/W	R
0x4800 2438	CONTROL_DEBOBS_6	Set and configure CORE observable signals 13 and 12	R/W	R
0x4800 243C	CONTROL_DEBOBS_7	Set and configure CORE observable signals 15 and 14	R/W	R
0x4800 2440	CONTROL_DEBOBS_8	Set and configure CORE observable signals 17 and 16	R/W	R
0x4800 2A68	CONTROL_WKUP_DEBOBS_0	Set and configure WKUP observable pins 3, 2, 1, 0	R/W	R
0x4800 2A6C	CONTROL_WKUP_DEBOBS_1	Set and configure WKUP observable pins 7, 6, 5, 4	R/W	R
0x4800 2A70	CONTROL_WKUP_DEBOBS_2	Set and configure WKUP observable pins 11, 10, 9, 8	R/W	R
0x4800 2A74	CONTROL_WKUP_DEBOBS_3	Set and configure WKUP observable pins 15, 14, 13, 12	R/W	R
0x4800 2A78	CONTROL_WKUP_DEBOBS_4	Set and configure WKUP observable pins 17, 16	R/W	R

The write capabilities of these registers differ according to the device type.

Perform the following steps to configure observability:

1. To configure the pads properly for hardware debug and observability, select the hardware debug (hw_dbg) function mode 5 in the MUXMODE field of the CONTROL.CONTROL_PADCONF_ CCDC _x or mode 7 in the MUXMODE field of the CONTROL.CONTROL_PADCONF_ETK_X registers.

2. For the observability pads, set the proper values of the WKUPOBSMUX field CONTROL.CONTROL_WKUP_DEBOBS_n. Up to 5 bits are used to select the signal set to be observed (0x00 selection sets the output to CORE_OSMUXn signal). For more information, see the description of each register in Section 6.4.6.2, *Observability Tables*.
3. To observe the CORE_OSMUXn signals from the first layer of the multiplexer, set the WKUPOBSMUX field CONTROL.CONTROL_WKUP_DEBOBS_n to 0x00, and then set the proper values of the OBSMUX field CONTROL.CONTROL_DEBOBS_n. A maximum of 7 bits is used to select the signal set to be observed (0b0000000 selection sets the output to 0).

For more information, see the description of each register in Section 6.4.6.2, *Observability Tables*.

6.4.6.2 Observability Tables

This section gives information about all modules and features in the high-tier device. See Chapter 1, the *Device Family* section, to check availability of modules and features. Unavailable module and feature pins are not functional.

Table 6-10 through Table 6-45 define the mapped internal signals for each OBSMUX and WKUPOBSMUX value.

Table 6-10. Internal Signals Multiplexed on OBSMUX0

Out Signal Name	Muxed Signal Name	OBSMUX0 Field CONTROL.CONTROL_DEBOBS_0 [22:16] (dec)	Description	High State	Low State
CORE_OBSMUX0[(1)]	tie_low	0	Signal tied low	-	Active state low
	CM_96_FCLK	1	96 MHz functional clock of the CM module.	-	-
	CM_32K_CLK	2	32 kHz functional clock of the CM module.	-	-
	PRCM_DPLL3_enable	3	Signal used to enable DPLL3.	DPLL is enabled	DPLL is disabled
	Reserved	4	-	-	-
	PRCM_NEON_forceWakeup	5	Indicates if a wakeup of the NEON domain is forced.	Wakeup is forced	Wakeup is not forced
	PRCM_COREL4_domainNready	6	Indicates if the CORE_L4 domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	PRCM_WKUP_domainNready	7	Indicates if the WKUP domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	Reserved	(16:8)	-	-	-
	sdma_PI_DMAREQ	(87:17)	DMA requests lines mapped to the System DMA module. See the *DMA* chapter for more information about the System DMA request mapping.	-	-
	sgx_SINTERRUPTN	(104:88)	Interrupt lines from the SGX. See the *SGX* chapter for more information about the Interrupt Requests mapping.	-	-
	Reserved	(116:105)	-	-	-
	SD2D _PICLKOCP	117	Interface clock of SD2D	-	-
	Reserved	118	-	-	-
	SD2D _PICLKOCP	119	Interface clock of SD2D	-	-
	Reserved	(127:120)	-	-	-

(1) 0x00 in WKUPOBSMUX0 field CONTROL.CONTROL_WKUP_DEBOBS_0[4:0]

Table 6-11. Internal Signals Multiplexed on OBSMUX1

Out Signal Name	Muxed Signal Name	OBSMUX1 Field CONTROL.CONTROL_DEBOBS_0[6:0] (dec)	Description	High State	Low State
CORE_OBSMUX1[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_DPLL3_M2_CLK	1	M2 clock generated by DPLL3.	-	-
	CM_SYS_CLK	2	System clock	-	-
	PRCM_DPLL3_enablediv	3	Signal used to enable the clock divisor of DPLL3.	DPLL clock divisor is enabled	DPLL clock divisor is disabled
	Reserved	4	-	-	-
	PRCM_NEON_domainNready	5	Indicates if the NEON domain is ready. In other words, is domain transition on-going?	Domain is not ready	Domain is ready
	Reserved	6	-	-	-
	PRCM_WKUP_domainNready	7	Indicates if the WKUP domain is ready. In other words, is domain transition on-going?	Domain is not ready	Domain is ready
	Reserved	(12:8)	-	-	-
	PRCM_CORE_96M_GFCLK	13	96 MHz functional clock of the CORE domain.	-	-
	Reserved	(16:14)	-	-	-
	sdma_PI_DMAREQ	(87:17)	DMA requests lines mapped to the System DMA module. See the *DMA* chapter for more information about the System DMA request mapping.	-	-
	Reserved	(127:88)	-	-	-

(1) 0x00 in WKUPOBSMUX1 field CONTROL.CONTROL_WKUP_DEBOBS_0[12:8]

Table 6-12. Internal Signals Multiplexed on OBSMUX2

Out Signal Name	Muxed Signal Name	OBSMUX2 Field CONTROL.CONTROL_DEBOBS_1[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX2(1)	tie_low	0	Signal tied low	-	Active state low
	PRCM_DPLL3_M2X2_CLK	1	M2X2 clock generated by DPLL3.	-	-
	PRCM_DPLL1_freqlock	2	Indicates if the frequency of DPLL1 is locked.	DPLL frequency is locked	DPLL frequency is not locked
	PRCM_DPLL4_freqlock	3	Indicates if the frequency of DPLL4 is locked.	DPLL frequency is locked	DPLL frequency is not locked
	PRCM_COREL3_IClkIsNotRunning	4	Indicates if the interface clock of the COREL3 domain is running or not.	Clock is not running	Clock is running
	PRCM_NEON_forceSleep	5	Indicates if the NEON domain is forced to sleep mode.	Sleep mode is forced	Sleep mode is not forced
	Reserved	6	-	-	-
	PRCM_EMU_domainIsIdle	7	Indicates if EMU domain is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	Reserved	8	-	-	-
	IVA2_WUGEN.IDLEACK	9	Indicates acknowledgement for idle mode of WUGEN module. See the *PRCM* chapter for more information about idle signal.	-	-
	SR[2].IDLEACK	10	Indicates acknowledgement for idle mode of SmartReflex2. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	(12:11)	-	-	-
	PRCM_CORE_48M_GFCLK	13	96 MHz functional clock of the CORE domain.	-	-
	Reserved	(16:14)	-	-	-
	sdma_PI_DMAREQ	(87:17)	DMA requests lines mapped to the System DMA module. See the*DMA* chapter for more information about the System DMA request mapping.	-	-
	Reserved	(127:88)	-	-	-

(1) 0x00 in WKUPOBSMUX2 field CONTROL.CONTROL_WKUP_DEBOBS_0[20:16]

Table 6-13. Internal Signals Multiplexed on OBSMUX3

Out Signal Name	Muxed Signal Name	OBSMUX3 Field CONTROL.CONTROL_DEBOBS_1[6:0] (dec)	Description	High State	Low State
CORE_OBSMUX3[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_L3_ICLK	1	Interface clock of the L3 interconnect.	-	-
	PRCM_DPLL1_bypass	2	Indicates if the DPLL1 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	PRCM_DPLL4_bypass	3	Indicates if the DPLL4 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	Reserved	(6:4)	-	-	-
	PRCM_EMU_domainMute	7	Generated by EMU domain to indicate that it does not require services from domain A (Domain A can go in INACTIVE state)	Domain is in standby mode	Domain is not in standby mode
	Reserved	8	-	-	-
	DSS.IDLEREQ	9	Indicates request for idle mode of DSS module. See the *PRCM* chapter 4 for more information about idle signal.	-	-
	MPU.MSTANDBY	10	MPU Mstandby assertion.	-	-
	Reserved	11	-	-	-
	SR2_UPDATECLK	12	Update clock of the SmartReflex2.	-	-
	PRCM_CORE_12M_GFCLK	13	12 MHz functional clock of the CORE domain.	-	-
	Reserved	(16:14)	-	-	-
	sdma_PI_DMAREQ	(87:17)	DMA requests lines mapped to the System DMA module. See the *DMA* chapter for more information about the System DMA request mapping.	-	-
	Reserved	(127:88)	-	-	-

(1) 0x00 in WKUPOBSMUX3 field CONTROL.CONTROL_WKUP_DEBOBS_0[28:24]

Table 6-14. Internal Signals Multiplexed on OBSMUX4

Out Signal Name	Muxed Signal Name	OBSMUX4 Field CONTROL.CONTROL_DEBOBS_2[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX4[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_L4_ICLK	1	Interface clock of the L4 interconnect.	-	-
	PRCM_DPLL1_idle	2	Indicates if DPLL1 is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_DPLL4_idle	3	Indicates if DPLL4 is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	MODEMINTC_IClkIsNotRunning	4	Indicates if interface clock to modem INTC is running.	-	-
	Reserved	(6:5)	-	-	-
	PRCM_EMU_forceWakeup	7	Indicates if a wakeup of the EMU domain is forced.	Wakeup is forced	Wakeup is not forced
	Reserved	8	-	-	-
	DSS.IDLEACK	9	Indicates acknowledgement for idle mode of DSS. See the *PRCM* chapter for more information about idle signal.	-	-
	IVA2.MSTANDBY	10	IVA2 Mstandby assertion.	-	-
	Reserved	(12:11)	-	-	-
	PRCM_CORE_L3_GICLK	13	Interface clock of the L3 interconnect.	-	-
	Reserved	(16:14)	-	-	-
	mpu_PIIRQ	(112:17)	Interrupt request lines mapped to the interrupt controller. See the *Interrupt Controller* chapter for more information about these interrupt lines.	-	-
	Reserved	(127:113)	-	-	-

(1) 0x00 in WKUPOBSMUX4 field CONTROL.CONTROL_WKUP_DEBOBS_1[4:0]

Table 6-15. Internal Signals Multiplexed on OBSMUX5

Out Signal Name	Muxed Signal Name	OBSMUX5 Field CONTROL.CONTROL_DEBOBS_2[6:0] (dec)	Description	High State	Low State
CORE_OBSMUX5[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_RM_ICLK	1	Interface clock of the RM block.	-	-
	PRCM_DPLL1_initz	2	Lock sequence initialization (HLH) of DPLL1	-	-
	PRCM_DPLL4_initz	3	Lock sequence initialization (HLH) of DPLL4	-	-
	CM_SysClkIsRunning	4	Indicates if the system clock is running or not.	The clock is running	The clock is not running
	Reserved	(6:5)	-	-	-
	PRCM_EMU_domainNready	7	Indicates if the EMU domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	SL_EMU.IDLEREQ	8	Indicates request for idle mode of SL_EMU. See the *PRCM* chapter for more information about idle signal.	-	-
	USBHOST.IDLERE Q	9	Indicates request for idle mode of USBHOST. See the *PRCM* chapter for more information about idle signal.	-	-
	SGX.MSTANDBY	10	SGX Mstandby assertion.	-	-
	Reserved	(12:11)	-	-	-
	PRCM_CORE_L4_GICLK	13	Interface clock of the L4 interconnect.	-	-
	Reserved	(16:14)	-	-	-
	mpu_PIIRQ	(112:17)	Interrupt request lines mapped to the interrupt controller. See the *Interrupt Controller* chapter for more information about these interrupt lines.	-	-
	Reserved	(127:113)	-	-	-

(1) 0x00 in WKUPOBSMUX5 field CONTROL.CONTROL_WKUP_DEBOBS_1[12:8]

Table 6-16. Internal Signals Multiplexed on OBSMUX6

Out Signal Name	Muxed Signal Name	OBSMUX6 Field CONTROL.CONTROL_DEBOBS_3[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX6	tie_low	0	Signal tied low	-	Active state low
	PRCM_FUNC_96M_FCLK	1	96 MHz functional clock.	-	-
	PRCM_DPLL1_enable	2	Signal used to enable DPLL1.	DPLL is enabled	DPLL is disabled
	PRCM_DPLL4_enable	3	Signal used to enable DPLL4.	DPLL is enabled	DPLL is disabled
	PRCM_WKUP_IClkIsRunning	4	Indicates if the interface clock of the WKUP domain is running or not.	The clock is running	The clock is not running
	Reserved	5	-	-	-
	SMX_L3.IDLEREQ	6	Indicates request for idle mode of SMX_L3. See the *PRCM* chapter for more information about idle signal.	-	-
	PRCM_USBHOST_domainIsIdle	7	Indicates if the USBHOST domain is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	SL_EMU.SIDLEACK	8	Indicates acknowledgement for idle mode of SL_EMU See the *PRCM* chapter for more information about idle signal.	-	-
	USBHOST.I DLEACK	9	Indicates acknowledgement for idle mode of USBHOST. See the *PRCM* chapter for more information about idle signal.	-	-
	CAM.MSTANDBY	10	CAM Mstandby assertion.	-	-
	Reserved	11	-	-	-
	PRCM_DSS_GICLK	12	Interface clock of the DSS module.	-	-
	PRCM_SECURITY_L3_GICLK	13	Interface clock of the L3 interconnect in the SECURITY clock domain.	-	-
	Reserved	(16:14)	-	-	-
	mpu_PIIRQ	(112:17)	Interrupt request lines mapped to the interrupt controller. See the *Interrupt Controller* chapter for more information about these interrupt lines.	-	-
	Reserved	(114:113)	-	-	-
	PRCM_DPLL1_LOSSREF	115	Reference input loss acknowledge of DPLL1	Signal Acknowledged	Signal not Acknowledged
	PRCM_DPLL2_LOSSREF	116	Reference input loss acknowledge of DPLL2	Signal Acknowledged	Signal not Acknowledged
	Reserved	(123:117)	-	-	-
	PRCM_DPLL3_LOSSREF	124	Reference input loss acknowledge of DPLL3	Signal Acknowledged	Signal not Acknowledged
	PRCM_DPLL4_LOSSREF	125	Reference input loss acknowledge of DPLL4	Signal Acknowledged	Signal not Acknowledged

Table 6-16. Internal Signals Multiplexed on OBSMUX6 (continued)

Out Signal Name	Muxed Signal Name	OBSMUX6 Field CONTROL.CONTROL_DEBOBS_3[22:16] (dec)	Description	High State	Low State
	[(1)]Reserved	(127:126)	-	-	-

(1) 0x00 in WKUPOBSMUX6 field CONTROL.CONTROL_WKUP_DEBOBS_1[20:16]

Table 6-17. Internal Signals Multiplexed on OBSMUX7

Out Signal Name	Muxed Signal Name	OBSMUX7 Field CONTROL.CONTROL_DEBOBS_3[6:0] (dec)	Description	High State	Low State
CORE_OBSMUX7[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_FUNC_48M_FCLK	1	48 MHz functional clock.	-	-
	PRCM_DPLL1_enablediv	2	Signal used to enable the clock divisor of DPLL1.	DPLL frequency divisor is enabled	DPLL frequency divisor is disabled
	PRCM_DPLL4_enablediv	3	Signal used to enable the clock divisor of DPLL4.	DPLL frequency divisor is enabled	DPLL frequency divisor is disabled
	PRCM_SRL4_IClkIsRunning	4	Indicates if the interface clock of the L4 SmartReflex domain is running or not.	The clock is running	The clock is not running
	Reserved	5	-	-	-
	PRCM_DSS_domainIsIdle	6	Indicates if the DSS domain is in Idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_USBHOST_forceWakeup	7	Indicates if a wakeup of the USBHOST domain is forced.	Wakeup is forced	Wakeup is not forced
	Reserved	8	-	-	-
	MCBSP[2].IDLEREQ	9	Indicates request for idle mode of MCBSP. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	(12:10)	-	-	-
	PRCM_SECURITY_L4_GICLK	13	Interface clock of the L4 interconnect in the SECURITY clock domain.	-	-
	Reserved	(16:14)	-	-	-
	mpu_PIIRQ	(112:17)	Interrupt request lines mapped to the interrupt controller. See the *Interrupt Controller* chapter for more information about these interrupt lines.	-	-
	Reserved	(114:113)	-	-	-
	PRCM_DPLL1_BREAKLOCKZ	115	Indicates if DPLL1 is in frequency lock condition.	Lock conditions not reached	Lock conditions reached
	Reserved	(122:116)	-	-	-
	PRCM_DPLL3_BREAKLOCKZ	123	Indicates if DPLL3 is in frequency lock condition.	Lock conditions not reached	Lock conditions reached
	PRCM_DPLL4_BREAKLOCKZ	124	Indicates if DPLL4 is in frequency lock condition.	Lock conditions not reached	Lock conditions reached
	Reserved	(127:125)	-	-	-

(1) 0x00 in WKUPOBSMUX7 field CONTROL.CONTROL_WKUP_DEBOBS_1[28:24]

Table 6-18. Internal Signals Multiplexed on OBSMUX8

Out Signal Name	Muxed Signal Name	OBSMUX8 Field CONTROL.CONTROL_DEBOBS_4[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX8[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_DSS_TV_FCLK	1	Functional clock of the DSS module for TV output.	-	-
	Reserved	2	-	-	-
	PRCM_DPLL5_freqlock	3	Indicates if the frequency of DPLL5 is locked.	DPLL frequency is locked	DPLL frequency is not locked
	PRCM_SGX_IClkIsNotRunning	4	Indicates if the interface clock of the 2D/3D Graphics Accelerator is running or not.	Clock is not running	Clock is running
	PRCM_SGX_domainIsIdle	5	Indicates if SGX domain is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_DSS_forceWakeup	6	Indicates if a wakeup of the DSS domain is forced.	Wakeup is forced	Wakeup is not forced
	PRCM_USBHOST_domainNready	7	Indicates if the DSS domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	Reserved	(12:8)	-	-	-
	SSI_L4.GICLK	13	SSI interface clock	-	-
	Reserved	(18:14)	-	-	-
	PRCM_DPLL1_PHASELOCK	19	Indicates if DPLL1 is in phase lock condition.	Lock conditions reached	Lock conditions not reached
	Reserved	(26:20)	-	-	-
	PRCM_DPLL3_PHASELOCK	27	Indicates if DPLL3 is in phase lock condition.	Lock conditions reached	Lock conditions not reached
	PRCM_DPLL4_PHASELOCK	28	Indicates if DPLL4 is in phase lock condition.	Lock conditions reached	Lock conditions not reached
	Reserved	29	-	-	-
	SR2_UPDATECLK	30	Update clock of SmartReflex2	-	-
	Reserved	(127:31)	-	-	-

(1) 0x00 in WKUPOBSMUX8 field CONTROL.CONTROL_WKUP_DEBOBS_2[4:0]

Table 6-19. Internal Signals Multiplexed on OBSMUX9

Out Signal Name	Muxed Signal Name	OBSMUX9 Field CONTROL.CONTROL_DEBOBS_4[8:0] (dec)	Description	High State	Low State
CORE_OBSMUX9[1]	tie_low	0	Signal tied low	-	Active state low
	Reserved	(2:1)	-	-	-
	PRCM_DPLL5_bypass	3	Indicates if the DPLL5 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	PRCM_DSS_IClkIsNotRunning	4	Indicates if the DSS interface clock is running or not.	Clock is not running	Clock is running
	PRCM_SGX_forceWakeup	5	Indicates if a wakeup of the SGX domain is forced.	Wakeup is forced	Wakeup is not forced
	PRCM_DSS_domainNready	6	Indicates if the DSS domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	PRCM_USBHOST_forceSleep	7	Indicates if the USBHOST domain forced to sleep mode.	Sleep mode is forced	Sleep mode is not forced
	Reserved	8	-	-	-
	USBHS.IDLEREQ	9	Request for idle mode of USBHOST See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	(11:10)	-	-	-
	PRCM_PER_48M_GFCLK	12	48 MHz functional clock of the PER domain.	-	-
	PRCM_DSS_96M_GFCLK	13	96 MHz functional clock of the DSS domain.	-	-
	Reserved	(42:14)	-	-	-
	PRCM_DPLL1_RECAL	43	Indicates that DPLL1 needs to perform internal re-calibration.	re-calibration required	re-calibration not required
	Reserved	44	-	-	-
	PRCM_DPLL3_RECAL	45	Indicates that DPLL3 needs to perform internal re-calibration.	re-calibration required	re-calibration not required
	PRCM_DPLL4_RECAL	46	Indicates that DPLL4 needs to perform internal re-calibration.	re-calibration required	re-calibration not required
	Reserved	(127:47)	-	-	-

(1) 0x00 in WKUPOBSMUX9 field CONTROL.CONTROL_WKUP_DEBOBS_2[12:8]

Table 6-20. Internal Signals Multiplexed on OBSMUX10

Out Signal Name	Muxed Signal Name	OBSMUX10 Field CONTROL.CONTROL_DEBOBS_5[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX10(1)	tie_low	0	Signal tied low	-	Active state low
	PRCM_CPEFUSE_FCLK	1	Functional clock of the EFUSE module.	-	-
	Reserved	2	-	-	-
	PRCM_DPLL5_idle	3	Indicates if the DPLL5 is idle.	Domain is in Idle mode	Domain is not in Idle mode
	Reserved	4	-	-	-
	PRCM_SGX_domainNready	5	Indicates if the MPU domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	PRCM_DSS_forceSleep	6	Indicates if the DSS domain is forced to sleep mode.	Sleep mode is forced	Sleep mode is not forced
	PRCM_USBHOST_domainFreeze	7	Indicates if the USBHOST domain is frozen.	Domain is frozen	Domain is not frozen
	Reserved	8	-	-	-
	USBHS.IDLEACK	9	Indicates acknowledgement for idle mode of USBHOST. See the *PRCM* chapter for more information about idle signal.	-	-
	USBHOST.MSTANDBY	10	USBHOST Mstandby assertion.	-	-
	PRCM_SR1_UPDATECLK	11	Update clock of the SmartReflex1	-	-
	PRCM_PER_96M_GFCLK	12	96 MHz functional clock of the PER domain.	-	-
	PRCM_MPU_EMU_CLK	13	Functional clock of the MPU module in the EMU domain.	-	-
	Reserved	(25:14)	-	-	-
	Reserved	(30:26)	-	-	-
	PRCM_DPLL1_BYPASS	31	Indicates if DPLL1 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	Reserved	32	-	-	-
	PRCM_DPLL3_BYPASS	33	Indicates if DPLL3 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	PRCM_DPLL4_BYPASS	34	Indicates if DPLL4 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	Reserved	(127:35)	-	-	-

(1) 0x00 in WKUPOBSMUX10 field CONTROL.CONTROL_WKUP_DEBOBS_2[20:16]

Table 6-21. Internal Signals Multiplexed on OBSMUX11

Out Signal Name	Muxed Signal Name	OBSMUX11 Field CONTROL.CONTROL_DEBOBS_5[6:0] (dec)	Description	High State	Low State
CORE_OBSMUX11(1)	tie_low	0	Signal tied low	-	Active state low
	PRCM_DPLL5_M2_CLK	1	M2 clock generated by DPLL5.	-	-
	Reserved	2	-	-	-
	PRCM_DPLL5_initz	3	Lock sequence initialization (HLH) of DPLL5	-	-
	PRCM_PERL4_IClkIsNotRunning	4	Indicates if the interface clock of the L4 interconnect in the PER domain is running or not.	Clock is not running	Clock is running
	PRCM_SGX_forceSleep	5	Indicates if the SGX domain is forced to sleep mode.	Sleep mode is forced	Sleep mode is not forced
	PRCM_DSS_domainFreeze	6	Indicates if the DSS domain is frozen.	Domain is frozen	Domain is not frozen
	PRCM_emudpll3srcclkactivitystatus	7	Indicates (when enabled and required) if DPLL3 is forced in lock mode in order to ensure high speed emulation or trace clocks.	Lock mode forced	Lock mode not forced
	Reserved	8	-	-	-
	SDRC.IDLEREQ	9	Request for idle mode of SDRC. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	10	-	-	-
	PRCM_SR1_FREQERRINTZ	11	Interrupt for error interface of SmartReflex1	-	-
	PRCM_PER_L4_GICLK	12	Interface clock of the L4 interconnect in the PER clock domain.	-	-
	Reserved	(29:13)	-	-	-
	PRCM_DPLL1_TICOPWDN	30	Indicates a Core DCO power down condition for DPLL1.	Conditions reached	Conditions not reached
	Reserved	31	-	-	-
	PRCM_DPLL3_TICOPWDN	32	Indicates a Core DCO power down condition for DPLL3.	Conditions reached	Conditions not reached
	PRCM_DPLL4_TICOPWDN	33	Indicates a Core DCO power down condition for DPLL4.	Conditions reached	Conditions not reached
	Reserved	(127:34)	-	-	-

(1) 0x00 in WKUPOBSMUX11 field CONTROL.CONTROL_WKUP_DEBOBS_2[28:24]

Table 6-22. Internal Signals Multiplexed on OBSMUX12

Out Signal Name	Muxed Signal Name	OBSMUX12 Field CONTROL.CONTROL_DEBOBS_6[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX12(1)	tie_low	0	Signal tied low	-	Active state low
	PRCM_DPLL1_FCLK	1	Functional clock of DPLL1.	-	-
	Reserved	2	-	-	-
	PRCM_DPLL5_enable	3	Signal used to enable DPLL5.	DPLL is enabled	DPLL is disabled
	PRCM_EMU_ClkIsRunning	4	Indicates if the EMU clock is running or not.	The clock is running	The clock is not running
	PRCM_COREL3_domainIsIdle	5	Indicates if the COREL3 domain is in Idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_PER_domainIsIdle	6	Indicates if the PER domain is in Idle.	Domain is in Idle mode	Domain is not in Idle mode
	Reserved	(8:7)	-	-	-
	SDRC.IDLEACK	9	Indicates acknowledgement for idle mode of SDRC. See the *PRCM* chapter for more information about idle signal.	-	-
	SDMA.MSTANDBY	10	SDMA Mstandby assertion.	-	-
	SR1_IRQ_CLEAR	11	Indicates signal for clear IRQ.	-	-
	PRCM_SR_L4_GICLK	12	Interface clock of the L4 interconnect in the SmartReflex domain.	-	-
	Reserved	(18:13)	-	-	-
	d2dinth_irq_input	(50:19)	D2D input interrupt	-	-
	Reserved	(127:51)	-	-	-

(1) 0x00 in WKUPOBSMUX12 field CONTROL.CONTROL_WKUP_DEBOBS_3[4:0]

Table 6-23. Internal Signals Multiplexed on OBSMUX13

Out Signal Name	Muxed Signal Name	OBSMUX13 Field CONTROL.CONTROL_DEBOBS_6 [6:0] (dec)	Description	High State	Low State
CORE_OBSMUX13(1)	tie_low	0	Signal tied low	-	Active state low
	Reserved	(2:1)	-	-	-
	PRCM_DPLL5_enablediv	3	Signal used to enable the clock divisor of DPLL5.	DPLL clock divisor is enabled	DPLL clock divisor is disabled
	PRCM_CPEFUSE_FClkIsNotRunning	4	Indicates if the CPEFUSE functional clock is running or not.	Clock is not running	Clock is running
	PRCM_COREL4_domainIsIdle	5	Indicates if the COREL4 domain is in Idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_PER_forceWakeup	6	Indicates if a wakeup of the PER domain is forced.	Wakeup is forced	Wakeup is not forced
	Reserved	(8:7)	-	-	-
	MODEM.IDLEREQ	9	Request for idle mode. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	(11:10)	-	-	-
	PRCM_USBHOST_GICLK	12	Interface clock of the L4 interconnect in the USBHOST module.	-	-
	PRCM_EMU_CORE_ALWON_CLK	13	Emulation clock used by PRCM in order to run the emulation trace. Supplied by DPLL3.	-	-
	Reserved	(18:14)	-	-	-
	d2dinth_irq_input	(50:19)	D2D input interrupt	-	-
	Reserved	(127:51)	-	-	-

(1) 0x00 in WKUPOBSMUX13 field CONTROL.CONTROL_WKUP_DEBOBS_3[12:8]

Table 6-24. Internal Signals Multiplexed on OBSMUX14

Out Signal Name	Muxed Signal Name	OBSMUX14 Field CONTROL.CONTROL_DEBOBS_7 [22:16] (dec)	Description	High State	Low State
CORE_OBSMUX14(1)	tie_low	0	Signal tied low	-	Active state low
	Reserved	1	-	-	-
	PRCM_DPLL3_freqlock	2	Indicates if the frequency of DPLL3 is locked.	DPLL frequency is locked.	DPLL frequency is not locked.
	PRCM_MPU_domainFreeze	3	Indicates if the MPU domain is frozen.	Domain is frozen.	Domain is not frozen.
	PRCM_MPU_domainIsIdle	4	Indicates if MPU domain is Idle.	Domain is in Idle mode.	Domain is not in Idle mode.
	Reserved	5	-	-	-
	PRCM_PER_domainNready	6	Indicates if the PER domain is ready. In other words, is domain transition on-going ?	Domain is not ready.	Domain is ready.
	PRCM_EMU_MStandby	7	EMU asserts this signal to initiate a transition to standby mode.	Transition initiated.	Transition not initiated.
	Reserved	8	-	-	-
	MODEM.IDLEACK	9	Indicates acknowledgement for idle mode of MODEM. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	(11:10)		-	-
	PRCM_USBHOST_48M_GFCLK	12	48 MHz functional clock of the USBHOST module.	-	-
	PRCM_EMU_PER_ALWON_CLK	13	High-frequency always-on clock in the PER domain used to generate for emulation clocks.	-	-
	Reserved	(18:14)	-	-	-
	d2dinth_irq_input	(50:19)	D2D input interrupt	-	-
	Reserved	(127:51)	-	-	-

(1) 0x00 in WKUPOBSMUX14 field CONTROL.CONTROL_WKUP_DEBOBS_3[20:16]

Table 6-25. Internal Signals Multiplexed on OBSMUX15

Out Signal Name	Muxed Signal Name	OBSMUX15 Field CONTROL.CONTROL_DEBOBS_7 [6:0] (dec)	Description	High State	Low State
CORE_OBSMUX15(1)	tie_low	0	Signal tied low	-	Active state low
	PRCM_GPT10_FCLK	1	Functional clock of GP-Timer #10.	-	-
	PRCM_DPLL3_bypass	2	Indicates if DPLL3 is bypassed, in other words if the clock divisor is 1.	DPLL is bypassed	DPLL is not bypassed
	Reserved	3	-	-	-
	PRCM_MPU_domainNready	4	Indicates if the MPU domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	PRCM_CORECM_domainIsIdle	5	Indicates if CORECM domain is in idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_PER_forceSleep	6	Indicates if the PER domain is forced to sleep mode.	Sleep mode is forced	Sleep mode is not forced
	PRCM_STATE_IS_ON_MPU	7	Indicates to the global Power Manager FSM that the MPU domain power state is ON.	FSM state is ON	FSM state is not ON
	Reserved	8	-	-	-
	SL_WAKEUP.IDLEREQ	9	Request for idle mode. See the *PRCM* chapter for more information about idle signal.	-	-
	SL_CORE.IDLEACK	10	Indicates acknowledgement for idle mode of SL_CORE. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	11	-	-	-
	PRCM_USBHOST_120M_GFCLK	12	96 MHz functional clock of the USBHOST module.	-	-
	PRCM_DPLL4_M2X2_CLK	13	M2X2 clock generated by DPLL4.	-	-
	Reserved	(127:14)	-	-	-

(1) 0x00 in WKUPOBSMUX15 field CONTROL.CONTROL_WKUP_DEBOBS_3[28:24]

Table 6-26. Internal Signals Multiplexed on OBSMUX16

Out Signal Name	Muxed Signal Name	OBSMUX16 Field CONTROL.CONTROL_DEBOBS_8[22:16] (dec)	Description	High State	Low State
CORE_OBSMUX16(1)	tie_low	0	Signal tied low	-	Active state low
	PRCM_GPT11_FCLK	1	Functional clock of GP-Timer #11.	-	-
	PRCM_DPLL3_idle	2	Indicates if the DPLL3 is idle.	Domain is in Idle mode	Domain is not in Idle mode
	Reserved	3	-	-	-
	PRCM_MPU_domainFreeze	4	Indicates if the MPU domain is frozen.	Domain is frozen	Domain is not frozen
	PRCM_CORE_domainNready	5	Indicates if the CORE domain is ready. In other words, is domain transition on-going?	Domain is not ready	Domain is ready
	PRCM_WKUP_domainIsIdle	6	Indicates if WKUP domain is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	Reserved	(8:7)	-	-	-
	SL_ WAKEUP.IDLEACK	9	Indicates acknowledgement for idle mode of SL_WAKEUP. See the *PRCM* chapter for more information about idle signal.	-	-
	USBHS.MSTANDBY	10	USBHOST Mstandby assertion.	-	-
	Reserved	11	-	-	-
	PRCM_SGX_GICLK	12	Interface clock of the L4 interconnect in the SGX clock module.	-	-
	PRCM_DPLL4_M4X2_CLK	13	M4X2 clock generated by DPLL4.	-	-
	Reserved	(127:14)	-	-	-

(1) 0x00 in WKUPOBSMUX16 field CONTROL.CONTROL_WKUP_DEBOBS_4[4:0]

Table 6-27. Internal Signals Multiplexed on OBSMUX17

Out Signal Name	Muxed Signal Name	OBSMUX17 Field CONTROL.CONTROL_DEBOBS_8 [6:0] (dec)	Description	High State	Low State
CORE_OBSMUX17[1]	tie_low	0	Signal tied low	-	Active state low
	PRCM_SGX_GFCLK	1	Functional clock of the L4 interconnect in the SGX clock module.	-	-
	PRCM_DPLL3_initz	2	Lock sequence initialization (HLH) of DPLL3	-	-
	PRCM_DSS_domainFreeze	3	Indicates if the DSS domain is frozen.	Domain is frozen	Domain is not frozen
	PRCM_NEON_domainIsIdle	4	Indicates if the NEON domain is Idle.	Domain is in Idle mode	Domain is not in Idle mode
	PRCM_COREL3_domainNready	5	Indicates if the COREL3 domain is ready. In other words, is domain transition on-going ?	Domain is not ready	Domain is ready
	Reserved	(7:6)	-	-	-
	MPU_INTC.IDLEREQ	8	Indicates request for idle mode of MPU_INTC. See the *PRCM* chapter for more information about idle signal.	-	-
	SR[1].IDLEREQ	9	Indicates request for idle mode of Smart Reflex1. See the *PRCM* chapter for more information about idle signal.	-	-
	Reserved	(11:10)	-	-	-
	MDM_INTC.GICLK	12	Interface clock to Modem INTC.	-	-
	PRCM_DPLL4_M5X2_CLK	13	M5X2 clock generated by DPLL4.	-	-
	Reserved	(127:14)	-	-	-

(1) 0x00 in WKUPOBSMUX17 field CONTROL.CONTROL_WKUP_DEBOBS_4[12:8]

Table 6-28. Internal Signals Multiplexed on WKUPOBSMUX0

Pin Name	Observed Signal Name	WKUPOBSMUX0 Field CONTROL.CONTROL_WKUP_DEBOBS_0[4:0] (dec)	Description	High State	Low State
hw_dbg0[1]	CORE_OBSMUX0	0	Signal multiplexed by OBSMUX0.	-	-
	PRCM_SYS_CLK	1	System clock running at the system clock frequency.	-	-
	PRCM_GPT5_ALWON_FCLK	2	Always-on functional clock of GP-Timer #5.	-	-
	PRCM_SGX_RST	3	Reset signal for SGX.	Reset not active	Reset active
	PRCM_USBHOST_RST	4	Reset signal for USBHOST.	Reset not active	Reset active
	Reserved	(18:5)	-	-	-
	FORCEEMUCM	19	Indicates if the EMULATION mode of the clock manager is forced.	Forced	Not forced
	PRCM_MPU_INTC_SWAKEUP	20	MPU Interrupt Controller asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_CCDC_PCLK[2:0]

Table 6-29. Internal Signals Multiplexed on WKUPOBSMUX1

Pin Name	Observed Signal Name	WKUPOBSMUX1 Field CONTROL.CONTROL_WKUP_DEBOBS_0[12:8] (dec)	Description	High State	Low State
hw_dbg1[1]	CORE_OBSMUX1	0	Signal multiplexed by OBSMUX1.	-	-
	PRCM_DPLL1_ALWON_FCLK	1	Always-on functional clock of DPLL1.	-	-
	PRCM_GPT6_ALWON_FCLK	2	Always-on functional clock of GP-Timer #6.	-	-
	PRCM_DSS_RST	3	Reset signal for DSS.	Reset not active	Reset active
	PRCM_ICEPICK_RSTPWRON	4	Cold reset signal for ICEPICK.	Reset not active	Reset active
	PRCM_eFuseReady_n	5	Indicates if the device eFuse scan is completed.	Scan complete	Scan not complete
	Reserved	(18:6)	-	-	-
	PRCM_emudpll3srcclkactivityctrl	19	Indicates (when enabled and required) if DPLL3 can be forced in lock mode in order to ensure high speed emulation or trace clocks.	DPLL lock can be forced	DPLL lock can not be forced
	Reserved	(31:20)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_CCDC_PCLK[18:16]

Table 6-30. Internal Signals Multiplexed on WKUPOBSMUX2

Pin Name	Observed Signal Name	WKUPOBSMUX2 Field CONTROL.CONTROL_WKUP_DEBOBS_0[20:16] (dec)	Description	High State	Low State
hw_dbg2[1]	CORE_OBSMUX2	0	Signal multiplexed by OBSMUX2.	-	-
	Reserved	1	-	-	-
	PRCM_GPT7_ALWON_FCLK	2	Always-on functional clock of GP-Timer #7.	-	-
	Reserved	3	-	-	-
	PRCM_BANDGAP_RSTPWRON	4	Reset signal for BANDGAP.	Reset not active	Reset active
	Reserved	(18:5)	-	-	-
	PRCM_EMU_block_reset_cm	19	Reset signal of the block that controls the EMU domain.	Reset not active	Reset active
	Reserved	(31:20)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_CCDC_HD[18:16]

Table 6-31. Internal Signals Multiplexed on WKUPOBSMUX3

Pin Name	Observed Signal Name	WKUPOBSMUX3 Field CONTROL.CONTROL_WKUP_DEBOBS_0[28:24] (dec)	Description	High State	Low State
hw_dbg3[1]	CORE_OBSMUX3	0	Signal multiplexed by OBSMUX3.	-	-
	PRCM_DPLL3_ALWON_FCLK	1	Always-on functional clock of DPLL3.	-	-
	PRCM_GPT8_ALWON_FCLK	2	Always-on functional clock of GP-Timer #8.	-	-
	Reserved	3	-	-	-
	PRCM_MPU_WD_RST	4	Reset signal for MPU from MPU-Watchdog.	Reset not active	Reset active
	Reserved	5	-	-	-
	PRCM_efuseClk_is_not_running	6	Indicates if the interface clock of eFuse is running or not.	Clock is not running	Clock is running
	Reserved	(8:7)	-	-	-
	PRCM_SYS_CLKREQ_out	9	Value of the SYS_CLKREQ pad when used as an output.	-	-
	Reserved	(18:10)	-	-	-
	PRCM_FORCEACTIVEWKUP	19	If asserted, the WKUP domain is unable to start a sleep transition.	Sleep transition impossible	Sleep transition possible
	PRCM_USBHOST_SWAKEUP	20	USBHOST asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_CCDC_WEN[2:0]

Table 6-32. Internal Signals Multiplexed on WKUPOBSMUX4

Pin Name	Observed Signal Name	WKUPOBSMUX4 Field CONTROL.CONTROL_WKUP_DEBOBS_1[4:0] (dec)	Description	High State	Low State
hw_dbg4[1]	CORE_OBSMUX4	0	Signal multiplexed by OBSMUX4.	-	-
	PRCM_DPLL4_ALWON_FCLK	1	Always-on functional clock of DPLL4.	-	-
	PRCM_GPT9_ALWON_FCLK	2	Always-on functional clock of GP-Timer #9.	-	-
	Reserved	(5:3)	-	-	-
	PRCM_mpu_domain_is_inactive	6	Indicates if the MPU domain is active or not.	Domain is inactive	Domain is active
	Reserved	(8:7)	-	-	-
	PRCM_SYS_CLKREQ_in	9	When SYS_CLKREQ is used as an input, it is used to control the SYS.CLKOUT1 (and the internal oscillator).	-	-
	Reserved	(18:10)	-	-	-
	PRCM_FORCEACTIVECORE	19	If asserted, the CORE domain is unable to start a sleep transition.	Sleep transition impossible	Sleep transition possible
	PRCM_GPIO2_SWAKEUP	20	GPIO2 asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_CCDC_DATA1[18:16]

Table 6-33. Internal Signals Multiplexed on WKUPOBSMUX5

Pin Name	Observed Signal Name	WKUPOBSMUX5 Field CONTROL.CONTROL_WKUP_DEBOBS_1[12:8] (dec)	Description	High State	Low State
hw_dbg5[1]	CORE_OBSMUX5	0	Signal multiplexed by OBSMUX5.	-	-
	PRCM_DPLL5_ALWON_FCLK	1	Always-on functional clock of DPLL5.	-	-
	PRCM_MPU_RST	2	Reset signal for MPU.	Reset not active	Reset active
	PRCM_PER_RST	3	Reset signal for PER domain.	Reset not active	Reset active
	PRCM_SYNCT_RST	4	Reset signal for SYNCT domain.	Reset not active	Reset active
	Reserved	(8:5)	-	-	-
	PRCM_SYS_CLKREQ_oen	9	Indicates if the SYS_CLKREQ pin is used as an output (Output enable).	Output enabled	Output disabled
	Reserved	(17:10)	-	-	-
	PRCM_PER_POWER_ISO	18	PM command for PER domain isolation.	Isolation is ON	Isolation is OFF
	Reserved	19	-	-	-
	PRCM_GPT2_SWAKEUP	20	GP-Timer2 asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_CCDC_DATA3[2:0]

Table 6-34. Internal Signals Multiplexed on WKUPOBSMUX6

Pin Name	Observed Signal Name	WKUPOBSMUX6 Field CONTROL.CONTROL_WKUP_DEBOBS_1[20:16] (dec)	Description	High State	Low State
hw_dbg6[1]	CORE_OBSMUX6	0	Signal multiplexed by OBSMUX6.	-	-
	PRCM_SR_ALWAYSON_FCLK	1	Always-on functional clock of SmartReflex.	-	-
	PRCM_MPU_RSTPWRON	2	Cold reset signal for MPU.	Reset not active	Reset active
	PRCM_PER_RSTRET	3	Retention reset signal for PER domain.	Reset not active	Reset active
	PRCM_ICEPICK_POR	4	This signal acts as a power-on reset and is activated when emulation tools need to use an EMULATION device as a SECURITY device.	Reset not active	Reset active
	Reserved	5	-	-	-
	PRCM_neon_domain_is_inactive	6	Indicates if the NEON domain is active or not.	Domain is inactive	Domain is active
	Reserved	(18:7)	-	-	-
	PRCM_IPPWR	19	Indicates if the emulation domain is fully operational.	EMU power operational	EMU power not operational
	PRCM_MCBSP2_SWAKEUP	20	MCBSP2 asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_CCDC_DATA3[18:16]

Table 6-35. Internal Signals Multiplexed on WKUPOBSMUX7

Pin Name	Observed Signal Name	WKUPOBSMUX7 Field CONTROL.CONTROL_WKUP_DEBOBS_1[28:24] (dec)	Description	High State	Low State
hw_dbg7[1]	CORE_OBSMUX7	0	Signal multiplexed by OBSMUX7.	-	-
	Reserved	(2:1)	-	-	-
	PRCM_WKUP_RST	3	Reset signal for WKUP domain.	Reset not active	Reset active
	PRCM_ICEPICK_RST	4	Reset signal for ICEPICK module.	Reset not active	Reset active
	Reserved	5	-	-	-
	PRCM_usbhost_domain_is_inactive	6	Indicates if the USBHOST domain is active or not.	Domain is inactive	Domain is active
	Reserved	(18:7)	-	-	-
	PRCM_ActLikeSecure	19	Signal asserted after a power-on reset in order to "detect" the emulation devices which will be then configured as SECURE devices.	Detection possible	Detection impossible
	Reserved	(31:20)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_CCDC_DATA5[2:0]

Table 6-36. Internal Signals Multiplexed on WKUPOBSMUX8

Pin Name	Observed Signal Name	WKUPOBSMUX8 Field CONTROL.CONTROL_WKUP_DEBOBS_2[4:0] (dec)	Description	High State	Low State
hw_dbg8[1]	CORE_OBSMUX8	0	Signal multiplexed by OBSMUX8.	-	-
	Reserved	1	-	-	-
	PRCM_NEON_RST	2	Reset signal for NEON.	Reset not active	Reset active
	PRCM_WKUP_RSTPWRON	3	Cold reset signal for WKUP domain.	Reset not active	Reset active
	Reserved	(19:4)	-	-	-
	PRCM_GPT10_SWAKEUP	20	GP-Timer10 asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_RMII_MDIO_CLK[18:16]

Table 6-37. Internal Signals Multiplexed on WKUPOBSMUX9

Pin Name	Observed Signal Name	WKUPOBSMUX9 Field CONTROL.CONTROL_WKUP_DEBOBS_2[12:8] (dec)	Description	High State	Low State
hw_dbg9[1]	CORE_OBSMUX9	0	Signal multiplexed by OBSMUX9.	-	-
	Reserved	(2:1)	-	-	-
	PRCM_EMU_RST	3	Reset signal for EMU domain.	Reset not active	Reset active
	Reserved	(31:4)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_RMII_RXD1[2:0]

Table 6-38. Internal Signals Multiplexed on WKUPOBSMUX10

Pin Name	Observed Signal Name	WKUPOBSMUX10 Field CONTROL.CONTROL_WKUP_DEBOBS_2[20:16] (dec)	Description	High State	Low State
hw_dbg10[1]	CORE_OBSMUX10	0	Signal multiplexed by OBSMUX 10.	-	-
	PRCM_FUNC_96M_ALWON_CLK	1	96 MHz Always-on functional clock.	-	-
	Reserved	2	-	-	-
	PRCM_EMU_RSTPWRON	3	Cold Reset signal for EMU domain.	Reset not active	Reset active
	PRCM_COREL3_domainIsOn	5	Indicates to the global Power Manager FSM that the COREL3 domain power state is ON.	State is ON	State is not ON
	Reserved	(5:4)	-	-	-
	PRCM_sgx_domain_is_inactive	6	Indicates if the SGX domain is active or not.	Domain is inactive	Domain is active
	Reserved	(19:7)	-	-	-
	PRCM_GPIO1_SWAKEUP	20	GPIO1 asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_RMII_RXER[2:0]

Table 6-39. Internal Signals Multiplexed on WKUPOBSMUX11

Pin Name	Observed Signal Name	WKUPOBSMUX11 Field CONTROL.CONTROL_WKUP_DEBOBS_2[28:24] (dec)	Description	High State	Low State
hw_dbg11[1]	CORE_OBSMUX11	0	Signal multiplexed by OBSMUX11.	-	-
	PRCM_DSS2_ALWON_FCLK	1	Always-on functional clock 2 of DSS module.	-	-
	Reserved	2	-	-	-
	PRCM_EFUSE_RSTPWRON	3	Cold reset signal for EFUSE.	Reset not active	Reset active
	PRCM_DPLL3_rstdata	4	Reset signal for DPLL3.	Reset active	Reset not active
	Reserved	(7:5)	-	-	-
	PRCM_PRM2MPU_IRQ	8	Interrupt line from PRM to the MPU.	-	-
	Reserved	(20:9)	-	-	-
	PRCM_WAITINRESET_WK	21	Indicates if the device is booting in Wait-In-Reset mode.	Wait-In-Reset mode	Standard boot
	Reserved	(31:22)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_RMII_RXER[18:16]

Table 6-40. Internal Signals Multiplexed on WKUPOBSMUX12

Pin Name	Observed Signal Name	WKUPOBSMUX12 Field CONTROL.CONTROL_WKUP_DEBOBS_3[4:0] (dec)	Description	High State	Low State
hw_dbg12[1]	CORE_OBSMUX12	0	Signal multiplexed by OBSMUX12.	-	-
	PRCM_EFUSE_ALWON_FCLK	1	Always-on functional clock of Efuse.	-	-
	Reserved	2	-	-	-
	PRCM_SR_RST	3	Reset signal for SmartReflex.	Reset not active	Reset active
	PRCM_GlbRstData	4	Global reset signal.	Reset not active	Reset active
	Reserved	5	-	-	-
	PRCM_core_domain_is_inactive	6	Indicates if the CORE domain is active or not.	Domain is inactive	Domain is active
	Reserved	(8:7)	-	-	-
	PRCM_IO_WUCLK	9	I/O Wakeup clock.	-	-
	Reserved	(31:10)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_DSS_PCLK[2:0]

Table 6-41. Internal Signals Multiplexed on WKUPOBSMUX13

Pin Name	Observed Signal Name	WKUPOBSMUX13 Field CONTROL.CONTROL_WKUP_DEBOBS_3[12:8] (dec)	Description	High State	Low State
hw_dbg13[1]	CORE_OBSMUX13	0	Signal multiplexed by OBSMUX13.	-	-
	PRCM_PER_32K_ALWON_FCLK	1	Always-on 23KHz functional clock of PER domain.	-	-
	PRCM_CORE_RST	2	Reset signal for CORE domain.	Reset not active	Reset active
	PRCM_DPLL1_RSTPWRON	3	Cold reset signal for DPLL1.	Reset active	Reset not active
	PRCM_GlblWarmRst_n	4	Global warm reset signal.	Reset active	Reset not active
	Reserved	(19:5)	-	-	-
	PRCM_GPT1_SWAKEUP	20	FP-Timer1 asserts this signal to initiate a transition to WAKEUP mode.	Transition initiated	Wakeup conditions not reached
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_DSS_PCLK[18:16]

Table 6-42. Internal Signals Multiplexed on WKUPOBSMUX14

Pin Name	Observed Signal Name	WKUPOBSMUX14 Field CONTROL.CONTROL_WKUP_DEBOBS_3[20:16] (dec)	Description	High State	Low State
hw_dbg14[1]	CORE_OBSMUX14	0	Signal multiplexed by OBSMUX14.	-	-
	PRCM_GPT1_FCLK	1	Functional clock of GP-Timer1.	-	-
	PRCM_CORE_RSTPWRONRET	2	Cold retention reset signal for CORE domain.	Reset not active	Reset active
	PRCM_DPLL2_RSTPWRON	3	Cold reset signal for DPLL2.	Reset active	Reset not active
	PRCM_GlblPwrOnRst_n	4	Global cold reset signal.	Reset active	Reset not active
	Reserved	(19:5)	-	-	-
	PRCM_EMU_SYS_GCLK	20	System clock of the EMU block of the PRCM module.	-	-
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_DSS_DATA6[2:0]

Table 6-43. Internal Signals Multiplexed on WKUPOBSMUX15

Pin Name	Observed Signal Name	WKUPOBSMUX15 Field CONTROL.CONTROL_WKUP_DEBOBS_3[28:24] (dec)	Description	High State	Low State
hw_dbg15[1]	CORE_OBSMUX15	0	Signal multiplexed by OBSMUX15.	-	-
	PRCM_GPT2_ALWON_FCLK	1	Always-on functional clock of GP-Timer #2.	-	-
	PRCM_CORE_RSTRET	2	Retention reset signal for CORE domain.	Reset not active	Reset active
	PRCM_DPLL3_RSTPWRON	3	Cold reset signal for DPLL3.	Reset active	Reset not active
	Reserved	(6:4)	-	-	-
	PRCM_dss_domain_is_inactive	7	Indicates if the DSS domain is active or not.	Domain is inactive	Domain is active
	Reserved	(31:8)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_DSS_DATA6[18:16]

Table 6-44. Internal Signals Multiplexed on WKUPOBSMUX16

Pin Name	Observed Signal Name	WKUPOBSMUX16 Field CONTROL.CONTROL_WKUP_DEBOBS_4[4:0] (dec)	Description	High State	Low State
hw_dbg16[1]	CORE_OBSMUX16	0	Signal multiplexed by OBSMUX16.	-	-
	PRCM_GPT3_ALWON_FCLK	1	Always-on functional clock of GP-Timer #3.	-	-
	PRCM_CPEFUSE_RST	2	Reset signal for Efuse.	Reset not active	Reset active
	PRCM_DPLL4_RSTPWRON	3	Cold reset signal for DPLL4.	Reset active	Reset not active
	Reserved	(19:4)	-	-	-
	PRCM_WKUP_32K_GFCLK	20	32KHz functional clock of the WKUP domain.	-	-
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE0 field CONTROL.CONTROL_PADCONF_DSS_DATA8[2:0]

Table 6-45. Internal Signals Multiplexed on WKUPOBSMUX17

Pin Name	Observed Signal Name	WKUPOBSMUX17 Field CONTROL.CONTROL_WKUP_DEBOBS_4[12:8] (dec)	Description	High State	Low State
hw_dbg17[1]	CORE_OBSMUX17	0	Signal multiplexed by OBSMUX17.	-	-
	PRCM_GPT4_ALWON_FCLK	1	Always-on functional clock of GP-Timer #4.	-	-
	PRCM_USBTLL_RST	2	Reset signal for USBTLL.	Reset not active	Reset active
	PRCM_DPLL5_RSTPWRON	3	Cold reset signal for DPLL5.	Reset active	Reset not active
	PRCM_ICECRUSHER_RST	4	Reset signal for ICECRUSHER.	Reset not active	Reset active
	PRCM_CORECM_domain Wakeup	5	Indicates that a wake-up condition is detected for the CORECM domain.	Conditions reached	Conditions not reached
	Reserved	6	-	-	-
	PRCM_per_domain_is_inactive	7	Indicates if the PER domain is active or not.	Domain is inactive	Domain is active
	Reserved	(19:8)	-	-	-
	PRCM_WKUP_L4_GICLK	20	Interface clock of WKUPL4 domain.	-	-
	Reserved	(31:21)	-	-	-

(1) Mode 5 in MUXMODE1 field CONTROL.CONTROL_PADCONF_DSS_DATA8[18:16]

6.4.7 Electromagnetic Interference Reduction for Clocking Generation (Spreading)

6.4.7.1 Overview

There are two major forms of digital signals: digital-data and digital-clock (CLK). Digital signals are the principal signal form in today's digital electronic products.

- A digital-data signal can be viewed as a sequence of pulses with different pulse widths
- A clock (CLK) signal is usually a string of rectangular pulses with the same pulse width

In the case of a clock, the generated signal has a predetermined frequency. Electromagnetic radiations such as radio frequency emissions are also generated from the constant frequency clock signals and their harmonics. Unfortunately, a problem exists with some devices such as communication devices for example in that the system clock generates unwanted spurious signals that interfere with the decoding of information from received signals by a receiver of the communication device. Note that data periodicity also causes interference.

This periodicity generates a significant power peak at the selected frequency, which in turn causes an EMI disturbance to the environment.

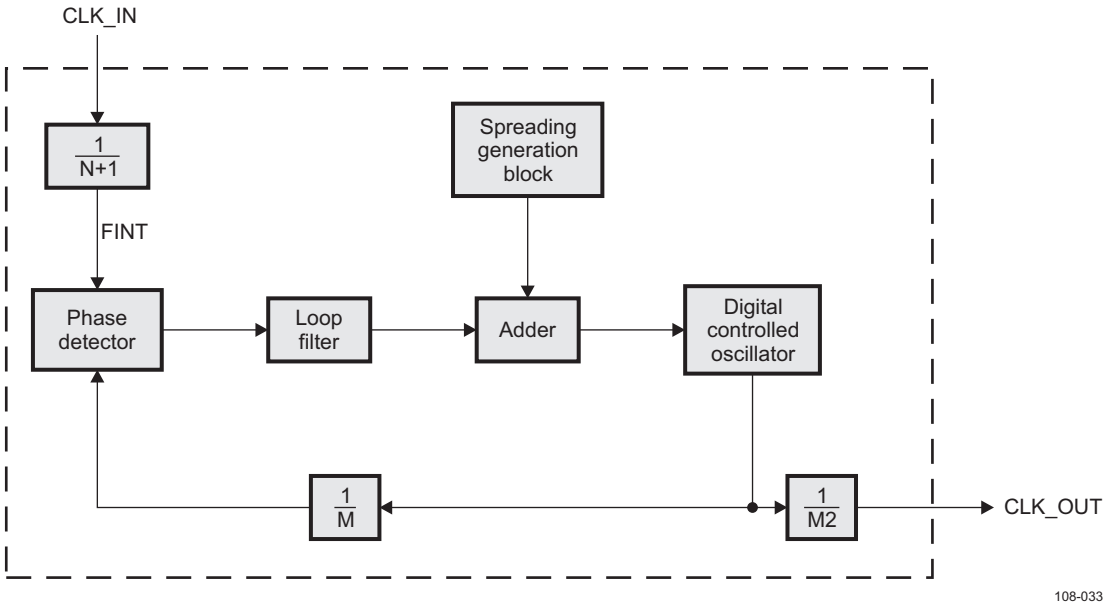
The harmonics of the generated signal system clock radiates in the other modules of the device, and the spurious energy can be enough to cause enough interference, which results in performance degradation (in the form of high bit error rates in case of a wireless communication interface for example).

In order to reduce the power peaks (and thus the energy of the electromagnetic noise generated for a specific frequency), an internal frequency modulation of the DPLL is used to distribute the energy to many different frequencies, thus reducing the power peaks.

This frequency modulation feature is directly integrated in some DPLLs of the device. The DPLLs that support this additional feature are called DPLL-D in the following.

Figure 6-10 shows a diagram of a DPLL that supports the EMI reduction feature.

Figure 6-10. DPLL with EMI Reduction Feature



The additional features of the DPLL compliant with EMI reduction are:

- EMI Reduction using Triangular frequency modulation, also called Spread Spectrum Clocking (SSC).

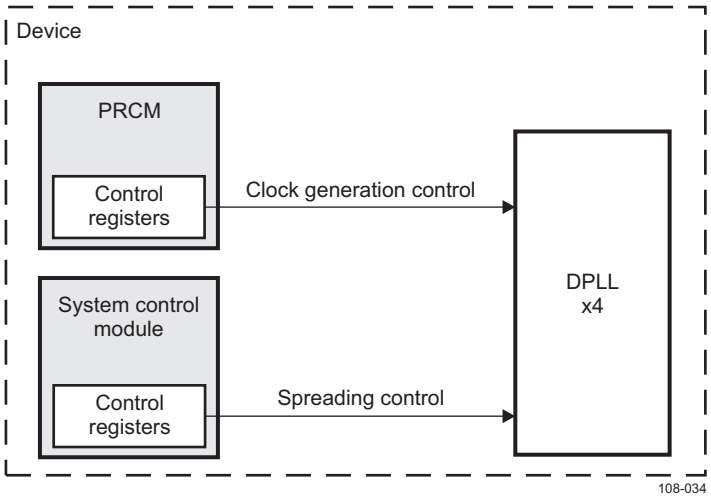
6.4.7.2 Integration

The four following DPLLs/Domains of the devices that support the EMI reduction feature are:

- DSS/DSI DPLL
- CORE domain DPLL
- PER domain DPLL
- USBHOST DPLL

Figure 6-11 highlights the four DPLL-D integration in the device.

Figure 6-11. DPLL-D Integration



The control of the DPLL is done both by the PRCM and the System Control Module:

- The PRCM controls the clocking generation of the DPLL. For more information, see the *Power Reset and Clock Management* chapter.
- The System Control Module contains all necessary bits that controls the EMI Reduction feature (Spread Spectrum Clocking).

6.4.7.2.1 Clocking, Reset, and Power Management Scheme

Please see the *Power Reset and Clock Management* chapter.

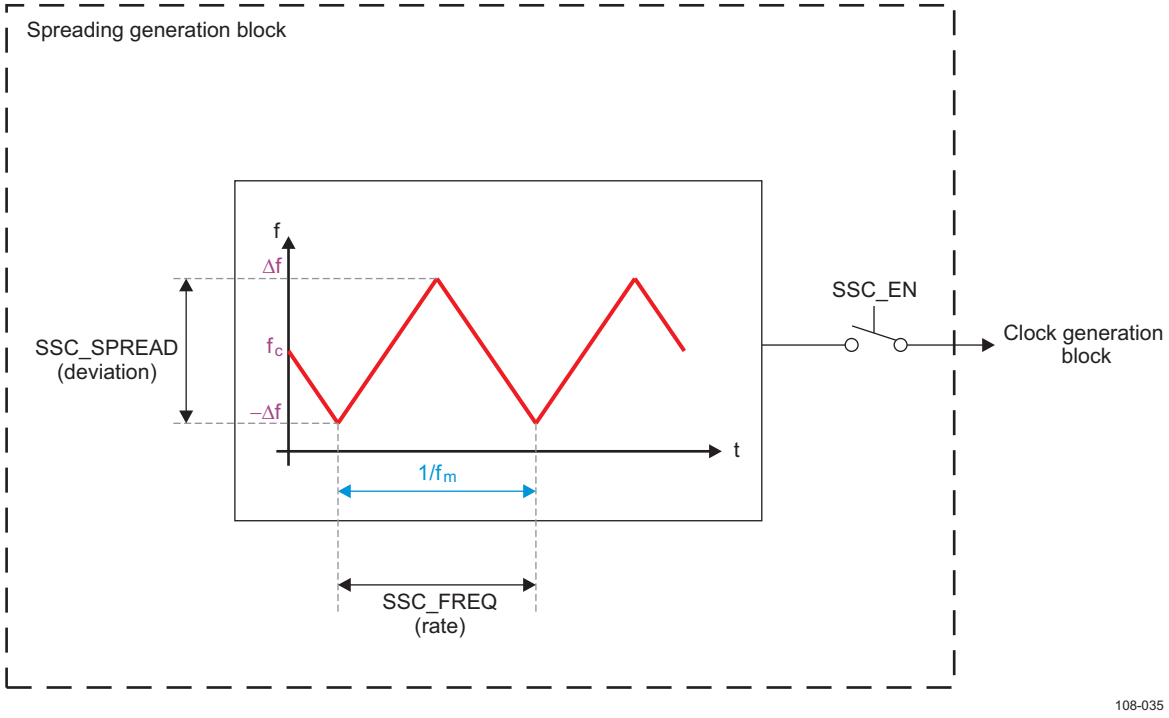
NOTE: DSI is not available on all devices. See Chapter 1, the *Device Family* section, to check availability of this module.

6.4.7.3 Functional Description

6.4.7.3.1 Spreading Generation Block

Figure 6-12 shows a diagram of the Spreading Generation block.

Figure 6-12. Spreading Generation Block Diagram



NOTE: Δf is the deviation from the center frequency. The total spreading deviation is equal to twice Δf.

f_c is the original clock frequency.

f_m is the spreading frequency.

This additional block generates the required waveform used to reduce EMI. This waveform is then modulated with the initial signal in order to add some controlled deviation in the clock signal frequency, which spreads the energy of the clock and its harmonics into a band of frequencies, and then reduces the Electomagnetic Interferences.

The **SSC_SPREAD** value handles the deviation (amplitude) of the generated signal from the original signal. It is controlled by the CONTROL.CONTROL_XXX_DPLL_SPREADING[3:2] XXX_SPREADING_AMPLITUDE bit field of the corresponding registers (where XXX is the name of the concerned DPLLs). Note that this register contains only a spreading ratio K, which is equal to $\Delta f / f_m$.

The **SSC_FREQ** value controls the rate of the generated signal. It is controlled by the CONTROL.CONTROL_XXX_DPLL_SPREADING[1:0] XXX_SPREADING_RATE bit field of the corresponding registers (where XXX is the name of the concerned DPLLs).

The **SSC_EN** signal enables/disables the frequency modulation feature of the DPLL. It is controlled by the CONTROL.CONTROL_XXX_DPLL_SPREADING[4] XXX_SPREADING_ENABLE bit field of the corresponding registers (where XXX is the name of the concerned DPLLs).

In any case the frequency modulation will be programmed in the System Control Module and it is not generally something that needs to be changed in real time.

6.4.7.3.2 Spread Spectrum Clocking (SSC)

6.4.7.3.2.1 Definition

The aim of the Spread Spectrum Clocking is to add a variation in the frequency of an original clock, which spreads the generated interferences over a larger band of frequency.

In theory, Spread Spectrum Clocking means that the clock signal is varied around the desired frequency. For example, for a 1 GHz clock, the frequency might be 999.5 Mhz at one moment in time and 1.0005 GHz at another. Doing this constantly causes the power of the tone to be "spread" out more over a broader band of tight frequencies (centered at the desired tone). To realize this constant variation on the original signal, a modulation with an additional signal (called spreading waveform) is realized.

Creating a spread-spectrum clock by spreading the initial clock frequency is done by defining the following parameters:

- The spreading frequency (deviation), which is the ratio of the range of spreading frequency over the original clock frequency.
- The modulation rate (f_m), which is used to determine the clock-frequency spreading-cycling rate, and is the time during which the generated clock frequency varies through Δf and returns to the original frequency.
- The modulation waveform, which describes the variation curve in term of time.

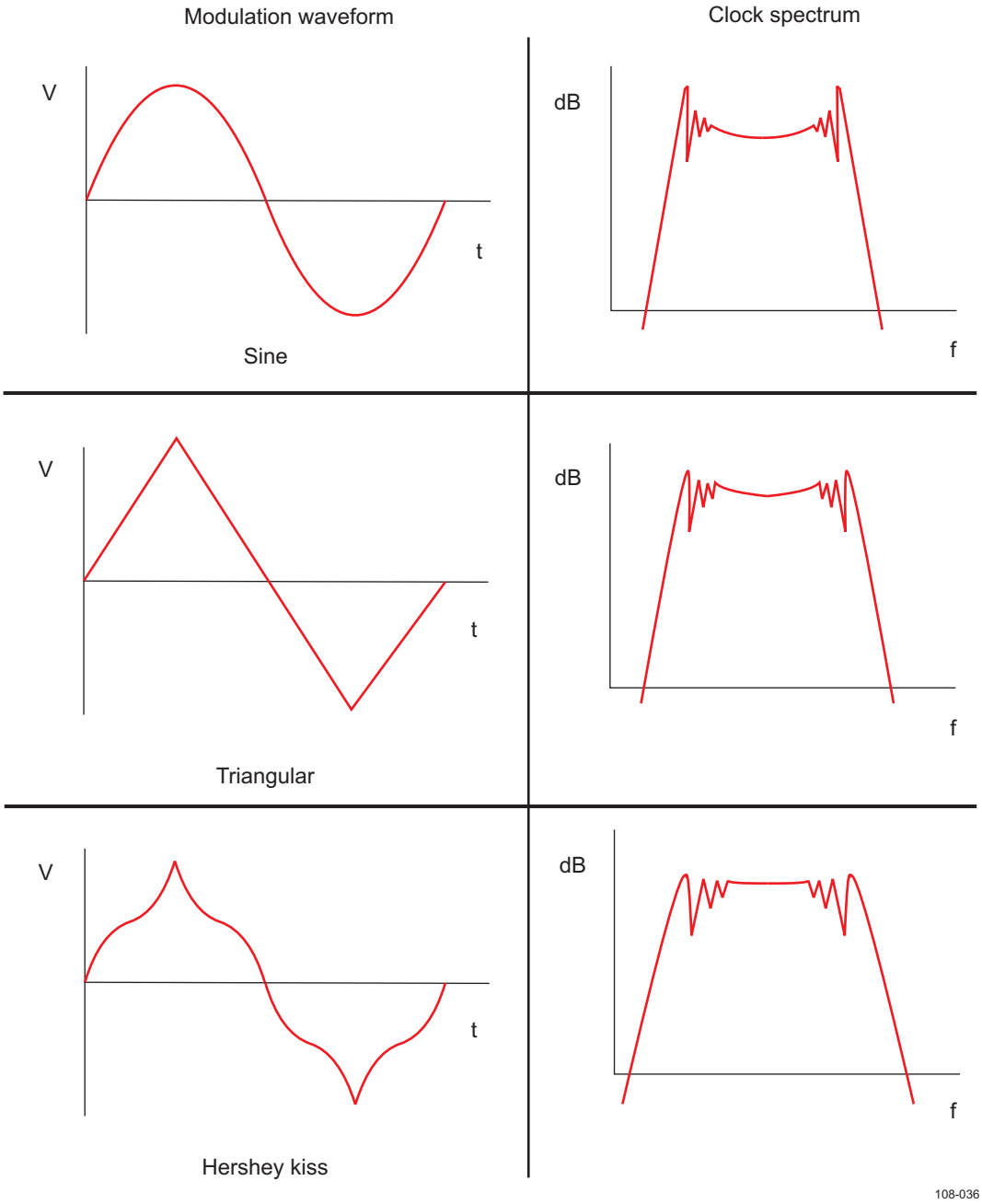
The spectral power reduction in the DPLL clocks is dependent on the Modulation Index (K) which is a ratio of spreading frequency, calculated from the frequency deviation (Δf) and the modulation rate (f_m).

6.4.7.3.2.2 The Modulation Waveforms

The shape (profile) of the generated clock signal depends on the modulation waveform that is used during the frequency modulation. Several profiles can be used, according to the desired shaping for the energy spreading.

Figure 6-13 shows three examples of modulation waveforms and the spectrum of the corresponding modulated clock signal.

Figure 6-13. Modulation Profiles



The triangular wave gives a relatively flat spectrum and is also easy to generate. The triangular waveform is used by the DPLL-D.

6.4.7.3.2.3 Effects on the Clock Signal

Figure 6-14 gives an example of the effect of a triangular spreading on a clock signal.

Figure 6-14. Effect of the SSC in Frequency

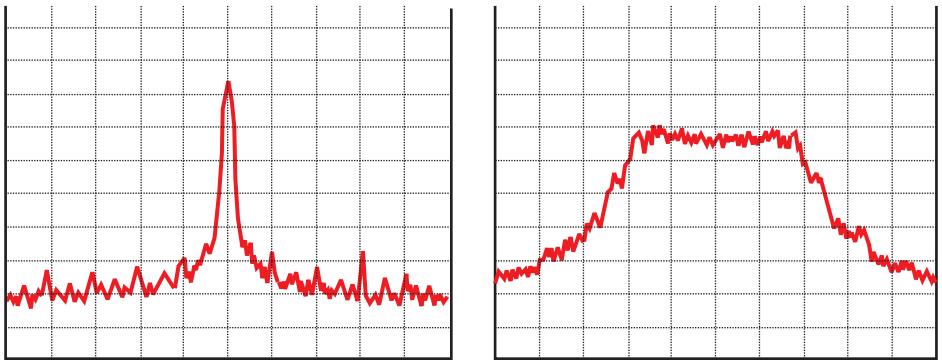
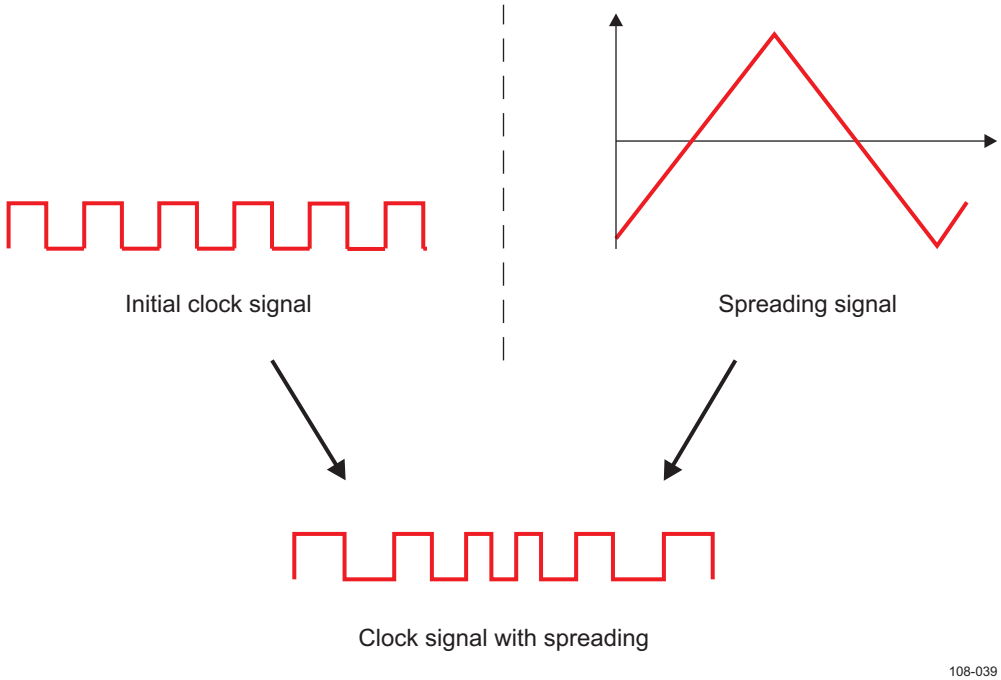


Figure 6-14 highlights the power reduction of the main peak, but also the flatter aspect of the modulated signal. One could notify that the minimum level of the second signal is higher than the first one. This effect is normal and is due to the "noise" added for the modulation.

NOTE: The spreading technique "scatters" the energy of the peaks on the other frequencies, which reduces the power of the peaks but increases the global "noise" of the signal.

Figure 6-15 shows the effect of a triangular spreading on a clock signal in the Time Domain.

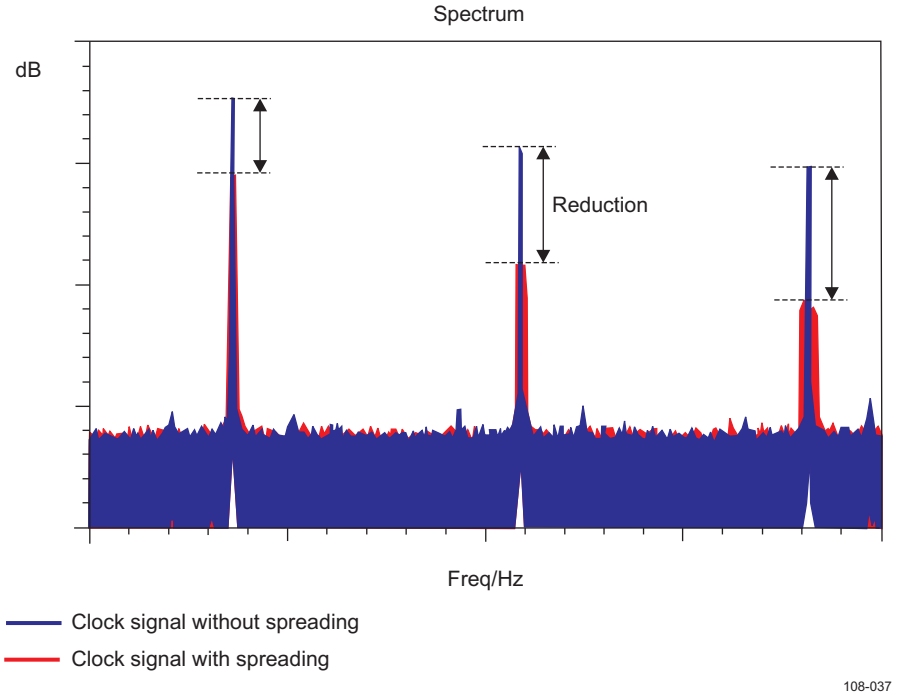
Figure 6-15. Effect of the SSC in the Time Domain



6.4.7.3.2.4 Estimation of the EMI Reduction Level

Figure 6-16 shows the effect of the spreading on a clock and its harmonics.

Figure 6-16. Peaks Reduction Due to Spreading



The electromagnetic interference reduction can be estimated with the following equation:

(1) Peak_power_reduction = 10 * log ((Deviation * f_c) / f_m)

With:

- Peak_power_reduction in dB
- Deviation in % of the initial clock frequency (f_c), equals to $\Delta f / f_c$
- f_c is the original clock frequency, in MHz
- f_m is the spreading frequency, in MHz

According to equation (1), it is also possible to compute the deviation, and then Δf, for a required peak power reduction:

(2) Deviation = (f_m / f_c) * $10^{(Peak\_power\_reduction\ /\ 10)}$

Example:

For f_c=400MHz, Deviation =1% peak from f_c(Δf = 4MHz) and f_m=400kHz, the estimated peak power reduction is 10dB.

6.4.7.3.2.5 Bandwidth Calculation (Carson Bandwidth Rule)

Carson's bandwidth rule defines the approximate bandwidth requirements of communications system components for a carrier signal that is frequency modulated by a continuous or broad spectrum of frequencies rather than a single frequency.

Carson's bandwidth rule is expressed by the relation CBR = 2 * (Δf + f_m) where CBR is the bandwidth requirement, Δf is the peak frequency deviation, and f_m is the highest frequency in the modulating signal.

For example, an FM signal with 5kHz peak deviation, and a maximum audio frequency of 3 kHz, would require an approximate bandwidth 2*(5+3) = 16 kHz.

Theoretically any FM signal will have an infinite number of sidebands and hence an infinite bandwidth but in practice all significant sideband energy (98% or more) is concentrated within the bandwidth defined by Carson's rule.

6.4.7.3.3 Frequency Limitations

There are some limitations on the use of the Spreading feature. Figure 6-17 shows the supported Spreading frequency and deviation.

Figure 6-17. Supported Spreading Frequency and Deviation

		SPREADING_AMPLITUDE (bits [3:2])			
		00	01	10	11
		K			
		4	6	8	10
SPREADING_RATE (bits [1:0])	Min to Max rate (kHz)	Frequency deviation +/- kHz			
00	62.5 to 125	250 to 500	375 to 750	500 to 1000	625 to 1250
01	125 to 250	500 to 1000	750 to 1500	1000 to 2000	1250 to 2500
10	250 to 500	1000 to 2000	1500 to 3000	2000 to 4000	2500 to 5000
11	500 to 1000	2000 to 4000	3000 to 6000	4000 to 8000	5000 to 10000

Not recommended. PLL jitter degradation and modulation amplitude not a linear function of FM.

108-040

Even if all the previous frequencies and deviations are supported by the DPLL-D, a range of "Safe operating regions" has been defined according to its impact on jitter.

Figure 6-17 shows the supported safe operating regions supported by the DPLL-D.

Figure 6-18. Supported Safe Operating Regions and Jitter Impact

CLKOUT FREQUENCY (25–900 MHz)	Recommended safe operating region	Max jitter impact in % due to spreading
400–900		+/– 1.25%
200–399		+/– 1.25%
100–199		+/– 1.25%
50–99		+/– 1.00%
25–49		+/– 2.00%

108-041

(1) Please see Figure 6-17 for color coding)

6.4.7.4 Basic Programming Model

6.4.7.4.1 Spread Spectrum Clocking Configuration

The configuration of the spreading feature is not mandatory when programming the DPLL. This feature is usually enabled when the DPLL clocks generate harmonics which can potentially interfere with the GSM carrier frequencies.

Once the "clock generation control" registers are configured, it is possible to configure the spreading on the clock signal.

The first thing to do is to calculate the *Deviation* and/or the *Spreading Rate* according to the desired peak power reduction. In addition to this, it is necessary to have K (modulation index), which value is equal to Δf / f_m.

Then, both XXX_SPREADING_AMPLITUDE and XXX_SPREADING_RATE bitfields can be configured with the calculated values.

Finally, the spreading has to be enabled using the XXX_SPREADING_ENABLE bit.

NOTE: It is necessary to carefully configure the spreading on a clock in order to avoid adding some "noise" on frequencies that are used by another module. For example, adding spreading on a clock to reduce noise on GSM frequencies can "move" the generated noise to the memory controller's frequency, and then degrade its performances.

Example:

For f_c=400MHz, Deviation =1% peak from f_c (Δf = 4MHz) and f_m=400kHz, the estimated peak power reduction is 10dB. K is then equal to 10. The DPLL can be configured as follow:

- XXX_SPREADING_RATE = 10 (250 to 500 KHz)
- XXX_SPREADING_AMPLITUDE = 11 (K = 10)
- XXX_SPREADING_ENABLE = 1

The state of the modulation feature can be monitored with the XXX_SPREADING_ENABLE_STATUS bit of the corresponding register.

NOTE:

- To support the spreading feature the DPLL Bandwidth is restricted to a max of 70KHz. This implies the Reference Clock frequency to be in range 0.75MHz to 2.1MHz, and then the XXX_DPLL_FREQSEL[3:0] bitfield to be in range "0011" to "0111".
- The required Δf (Frequency deviation) is targeted on the CLKOUT Frequency is achieved when M2 is programmed to 1. Otherwise the frequency deviation will be scaled down by the factor of M2 value. However, this does not affect the deviation, so frequency change of harmonics at GSM frequencies is the same.
- The spread of modulation frequency within each range is due to an internal auto-ranging function and specific frequencies cannot be selected.
- The lowest range (62.5 to 125 KHz) should only be selected when Reference Clock (PLL internal reference frequency) < 1.1 MHz is being used, to prevent the PLL feedback loop from canceling the modulation.

When deactivating the spreading (XXX_SPREADING_ENABLE = 0), the End-of-Spreading is synchronous to the internal spreading cycle. So there is no residual average frequency error. PER_DPLL_SPREADING_RATE option is not supported.

6.5 System Control Module Programming Model

6.5.1 Feature Settings

6.5.1.1 Video Driver

This section gives information about all modules and features in the high-tier device. See Chapter 1, the *Device Family* section, to check availability of modules and features. Unavailable module and feature pins are not functional.

- TVOUTBYPASS bit CONTROL.CONTROL_DEVCONF1[18]:
  0b0: Dual 10-bit video DAC TV out bypass disable
  0b1: Dual 10-bit video DAC TV out bypass enable
- TVACEN bit CONTROL.CONTROL_DEVCONF1[11]:
  0b0: Enables dc coupling for TV output
  0b1: Enables ac coupling for TV output

For more details on video DACs, see the *Display Subsystem* chapter.

6.5.1.2 McBSP1 Internal Clock

The McBSP1 internal clock gates the internal interconnect clock and selects the FSR, CLKR, and CLKS input for the McBSP1.

- MCBSP1_FSR bit CONTROL.CONTROL_DEVCONF0[4]:
  0: FSR is from the pin mcbsp1_fsr.
  1: FSR is from the pin mcbsp1_fsx.
- MCBSP1_CLKR bit CONTROL.CONTROL_DEVCONF0[3]:
  0: CLKR is from the pin mcbsp1_clkr.
  1: CLKR is from the pin mcbsp1_clkx.
- MCBSP1_CLKS bit CONTROL.CONTROL_DEVCONF0[2]:
  0: CLKS is from the PRCM functional clock.
  1: CLKS is from the pin mcbsp_clks.

For more details on McBSP, see the *Multichannel Serial Port Interface* chapter.

6.5.1.3 McBSP2 Internal Clock

The McBSP2 internal clock gates the internal interconnect clock and selects the FSR, CLKR, and CLKS input for the McBSP2. The McBSP2 does not have mcbsp2_clkr and mcbsp2_fsr external pins. Clock input is from the mcbsp2_clkx pin; FSR input is from the mcbsp2_fsx pin.

- MCBSP2_CLKS register bit CONTROL.CONTROL_DEVCONF0[6]:
  0: Clock is from the PRCM functional clock.
  1: Clock is from the external pin mcbsp_clks.

For more details on McBSP, see the *Multichannel Serial Port Interface* chapter.

6.5.1.4 McBSP3 Internal Clock

The McBSP3 internal clock gates the internal interconnect clock and selects the FSR, CLKR, and CLKS input for the McBSP3. The McBSP3 does not have mcbsp3_clkr and mcbsp3_fsr external pins. Clock input is from the mcbsp3_clkx pin; FSR input is from the mcbsp3_fsx pin.

- MCBSP3_CLKS register bit CONTROL.CONTROL_DEVCONF1[0]:
  0: Clock is from the PRCM functional clock.
  1: Clock is from the external pin mcbsp_clks.

For more details on McBSP, see the *Multichannel Serial Port Interface* chapter.

6.5.1.5 McBSP4 Internal Clock

The McBSP4 internal clock gates the internal interconnect clock and selects the FSR, CLKR, and CLKS input for the McBSP4. The McBSP4 does not have mcbsp4_clkr and mcbsp4_fsr external pins. Clock input is from the mcbsp4_clkx pin; FSR input is from the mcbsp4_fsx pin.

- MCBSP4_CLKS register bit CONTROL.CONTROL_DEVCONF1[2]:
  0: Clock is from the PRCM functional clock.
  1: Clock is from the external pin mcbsp_clks.

For more details on McBSP, see the *McBSP* chapter.

6.5.1.6 McBSP5 Internal Clock

The McBSP5 internal clock gates the internal interconnect clock and selects the FSR, CLKR, and CLKS input for the McBSP5. The McBSP5 does not have mcbsp5_clkr and mcbsp5_fsr external pins. Clock input is from the mcbsp5_clkx pin; FSR input is from the mcbsp5_fsx pin.

- MCBSP5_CLKS register bit CONTROL.CONTROL_DEVCONF1[4]:
  0: Clock is from the PRCM functional clock.
  1: Clock is from the external pin mcbsp_clks.

For more details on McBSP, see the *McBSP* chapter.

6.5.1.7 MMC/SD/SDIO2 Module Input Clock Selection

- MMCSDIO2ADPCLKISEL bit CONTROL.CONTROL_DEVCONF1[6]:
  0: Input clock is from the external pin.
  1: Internal loopback, module input clock is copied from the module output clock.

6.5.1.8 Setting Sensitivity on SYS_NDMAREQ[3:0] Input Pins

The seven SYS_NDMAREQ0 to SYS_NDMAREQ6 input pins can be either level or edge sensitive.

- SENSDMAREQ0 bit CONTROL.CONTROL_DEVCONF0[0]:
  0: Level sensitivity
  1: Edge sensitivity
- SENSDMAREQ1 bit CONTROL.CONTROL_DEVCONF0[1]:
  0: Level sensitivity
  1: Edge sensitivity
- SENSDMAREQ2 bit CONTROL.CONTROL_DEVCONF1[7]:
  0: Level sensitivity
  1: Edge sensitivity
- SENSDMAREQ3 bit CONTROL.CONTROL_DEVCONF1[8]:
  0: Level sensitivity
  1: Edge sensitivity

For more details on DMA, see the *DMA* chapter.

6.5.1.9 Force MPU Writes to Be Nonposted

MPUFORCEWRNP bit CONTROL.CONTROL_DEVCONF1[9] bit is for debugging only. When this bit is set to 1 (for debugging), all writes are forced to nonposted and the cache attributes are ignored. By default, this bit should be set to 0 (posted writes). For the best performance, keep this bit at 0.

6.5.2 Pad Configuration Programming Points

To configure the pad, ensure that the following are done:

- Identify signals required on the interface based on the target application.

Example: You want to configure UART1 interface on balls. The required signals are: uart1_tx, uart1_rts, uart1_cts, and uart1_rx.

- Choose the pads used for those signals. Some signals could be available on several pads and/or may be multiplexed with other signals that can be need for an other application. Refer to Section 6.4.4.3, *Pad Multiplexing Register Fields.*

  Example: Each UART1 interface signals are available on two pads. These signals are multiplexed with a McBSP signal.

  Assume that the McBSP interface is required in the system. Therefore, for the UART1 interface signals, pads must be used where those signals are not multiplexed with the McBSP signal.

- Identify the pad configuration registers associated to the pads you will use in your application. Refer to Section 6.4.4.3, *Pad Multiplexing Register Fields.*

  Example: With the previous hypothesis the pad configuration registers to program are:

  – uart1_cts: CONTROL.CONTROL_PADCONF_DSS_DATA0[15:0]
  – uart1_rts: CONTROL.CONTROL_PADCONF_DSS_DATA0[31:16]
  – uart1_tx: CONTROL.CONTROL_PADCONF_DSS_DATA6[15:0]
  – uart1_rx: CONTROL.CONTROL_PADCONF_DSS_DATA6[31:16]

  NOTE: In that configuration, dss_datan (where n = 0) signals will not be available on these balls.

- Configure the MUXMODE field of each pad configuration registers (CONTROL.CONTROL_PADCONF_X) associated to the pads used. Section 6.4.4.3, *Pad Multiplexing Register Fields* lists the entire mode available for each pin. Write the binary value of the mode used in the MUXMODE field of the pad configuration registers.

  Example: UART1 signals are available in mode 2. So you should write 0b010 in the corresponding MUXMODE field of pad configurations registers to program.

- Configure the pull of the pad when you use it as input. When the pad is used as output the pull is automatically disabled. Section 6.4.4.3, *Pad Multiplexing Register Fields* lists the pull available on each pad and it reset value. Set the appropriate value for PULLTYPESELECT bit of the pad configuration register to set the pull value (0b0 = Pull Down selected, 0b1 = Pull Up selected) and set the PULLUDENABLE bit of the pad configuration register to enable the pull on the pad (0b0 = pull disabled, 0b1 = pull enabled)

  Example: uart1_rts and uart1_tx are output signals, so the pull will be automatically disabled on the pad. uart1_cts and uart1_rx are input signals, so you should configure the pull for these pads:

  – uart1_cts: enable pull-up (write 0b1 in the PULLTYPESELECT bit and 0b1 in the PULLUDENABLE bit of the corresponding pad configuration register)
  – uart1_rx: enable pull-up (write 0b1 in the PULLTYPESELECT bit and 0b1 in the PULLUDENABLE bit of the corresponding pad configuration register)

- Set the INPUTENABLE bit of the pad configuration register if your pin is use as input.

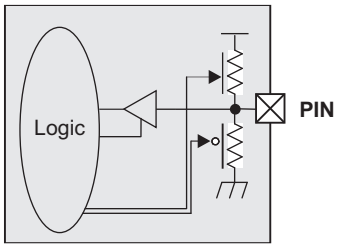
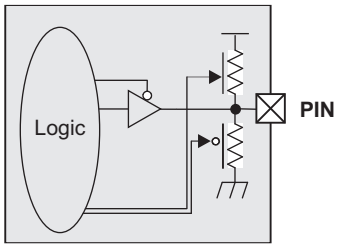
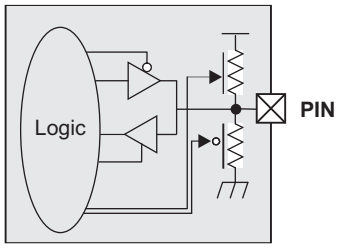
  Example: uart1_rts and uart1_tx are output signals, so clear the INPUTENABLE bit of the corresponding pad configuration register. uart1_cts and uart1_rx are input signals, so set the INPUTENABLE bit of the corresponding pad configuration register.

  NOTE: The order is unimportant for the previous setting of the pad configuration bits.

6.5.3 I/O Power Optimization Guidelines

In order to optimize I/O power, it is important to avoid unconnected or incorrectly-pulled pins. According to the type of the pins, the way to reduce power consumption can differ. Table 6-46 shows the 3 available pin (or ball) types.

Table 6-46. Existing Pin Types

Input	Output	Bidirectional
		

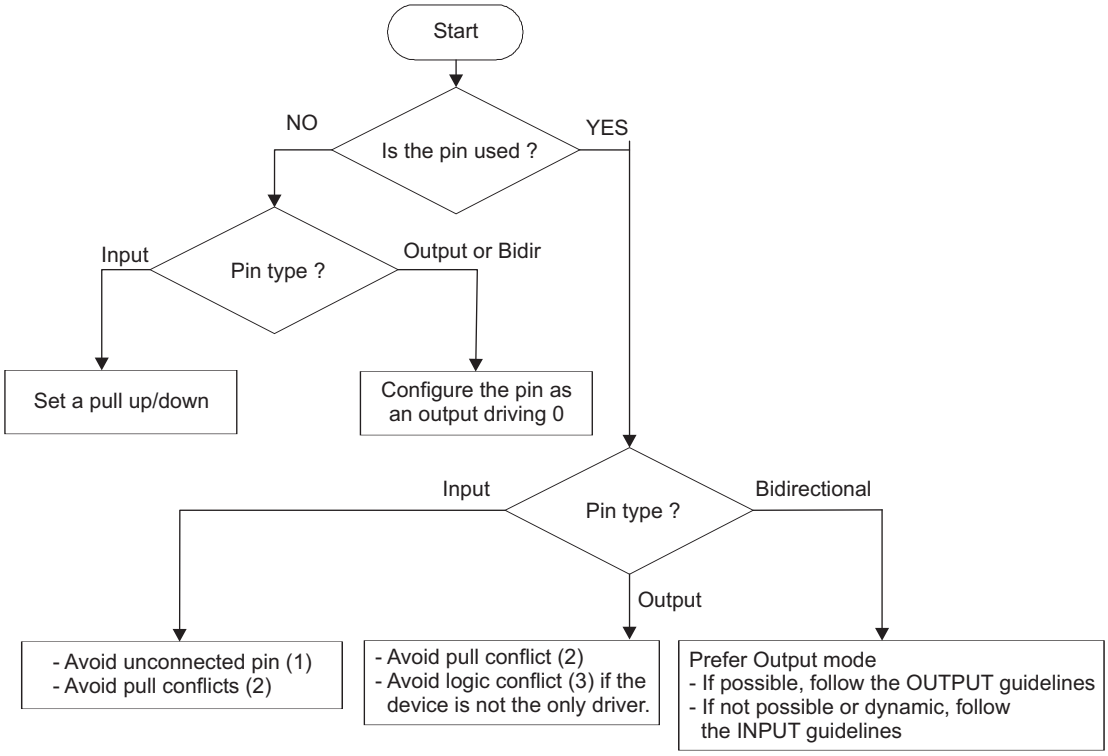
The configuration differs according to the I/O cell types. The following describes some pieces of advice which can be useful to avoid extra current leakage:

- For input pins, use a pull up/down when possible.
- For output pins, check existing pulls to avoid conflicts.
- For bidirectional pins, reconfigure the pin as an output driving 0 when possible.

Some I/O configurations involve modifications during the software setup of I/Os and sometimes require several hardware updates.

Figure 6-19 shows how to optimize the power consumption of pads.

Figure 6-19. I/O Power Optimization Flowchart



The following notes give additional explanations about the pin configuration.

1. In order to avoid unconnected pins, the configuration depends on its use:
   - If the pin is not driven externally, a pull up/down is required.
   - Otherwise, a pull up/down is not necessary.
2. Pull conflicts occur when there are different pulls on the same line. In order to correctly configure the pin, avoid external and internal pull together.
3. Logic conflicts consist in different electrical levels at the same time on one line. This can occur when several devices are connected to the same line. The two possible cases are:
   - If no external device drives the line, configure the pin to drive a '0'.
   - If another device drives the line, either the same value has to be driven or the pin has to be disconnected (HZ).

NOTE: It is advised to use high impedance logical state either on the device or the external component when the line is driven by both components.

The I/O pads are software-controlled by:

- Writing to the CONTROL_PADCONF_X registers in the Control Module for input/output and pull up/down configuration.
- Writing to the GPIOi.GPIO_OE registers in the GPIO module for input/output configuration.

For more information about how to configure the I/O pads, see Chapter 6, *System Control Module*.

For more information about the GPIO module, see the *General-Purpose Interface* chapter.

NOTE: For a correct configuration of each pin direction (input, output, bidirectional), both the CONTROL_PADCONF_X and the GPIOi.GPIO_OE registers must be written.

For more information about the I/O cell types and the corresponding optimized configuration, please contact your TI representative.

6.6 System Control Module Registers

This section gives information about all modules and features in the high-tier device. See Chapter 1, the *Device Family* section, to check availability of modules and features. Unavailable module and feature pins are not functional.

Table 6-47 lists the base address and address space for the SCM instances.

Table 6-47. Instance Summary

Module Name	Base Address	Size
INTERFACE	0x4800 2000	36 bytes
PADCONFS	0x4800 2030	564 bytes
GENERAL	0x4800 2270	767 bytes
MEM_WKUP	0x4800 2600	1K byte
PADCONFS_WKUP	0x4800 2A00	80 bytes
GENERAL_WKUP	0x4800 2A60	31 bytes

Some register description tables in this section are divided into different device types and ModeSP according to their access rights and reset value. For example, CONTROL.CONTROL_EXT_SEC_CONTROL is divided into seven tables named "TypeAModeB Bitfield Details" where:

- *A* can take different expressions depending on the device type:
  - E: EMULATOR device
  - G: General-purpose device
  - T: TEST device
  - S: SECURE device
  - B: BAD device
  - X: DON'T CARE device

  These access rights are enumerated in register descriptions as needed.
- *B* can take different expressions depending on the mpu mode:
  - SP: SECURE PRIVILEGE mode
  - NSP: NON SECURE PRIVILEGE mode
  - X: DON'T CARE mode

6.6.1 System Control Module Register Mapping Summary

NOTE: All module registers are 8-, 16-, or 32-bit accessible through the L4 interconnect (little endian encoding).

Table 6-48 lists the INTERFACE registers.

Table 6-48. INTERFACE Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_REVISION	R	32	0x0000 0000	0x4800 2000	Section 6.6.2.1
CONTROL_SYSCONFIG	RW	32	0x0000 0010	0x4800 2010	Section 6.6.2.2

Table 6-49 lists the PADCONFS registers.

Table 6-49. PADCONFS Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_PADCONF_SDRC_D0	RW	32	0x0000 0000	0x4800 2030	Section 6.6.3
CONTROL_PADCONF_SDRC_D2	RW	32	0x0000 0004	0x4800 2034	Section 6.6.3
CONTROL_PADCONF_SDRC_D4	RW	32	0x0000 0008	0x4800 2038	Section 6.6.3
CONTROL_PADCONF_SDRC_D6	RW	32	0x0000 000C	0x4800 203C	Section 6.6.3
CONTROL_PADCONF_SDRC_D8	RW	32	0x0000 0010	0x4800 2040	Section 6.6.3
CONTROL_PADCONF_SDRC_D10	RW	32	0x0000 0014	0x4800 2044	Section 6.6.3
CONTROL_PADCONF_SDRC_D12	RW	32	0x0000 0018	0x4800 2048	Section 6.6.3
CONTROL_PADCONF_SDRC_D14	RW	32	0x0000 001C	0x4800 204C	Section 6.6.3
CONTROL_PADCONF_SDRC_D16	RW	32	0x0000 0020	0x4800 2050	Section 6.6.3
CONTROL_PADCONF_SDRC_D18	RW	32	0x0000 0024	0x4800 2054	Section 6.6.3
CONTROL_PADCONF_SDRC_D20	RW	32	0x0000 0028	0x4800 2058	Section 6.6.3
CONTROL_PADCONF_SDRC_D22	RW	32	0x0000 002C	0x4800 205C	Section 6.6.3
CONTROL_PADCONF_SDRC_D24	RW	32	0x0000 0030	0x4800 2060	Section 6.6.3
CONTROL_PADCONF_SDRC_D26	RW	32	0x0000 0034	0x4800 2064	Section 6.6.3
CONTROL_PADCONF_SDRC_D28	RW	32	0x0000 0038	0x4800 2068	Section 6.6.3
CONTROL_PADCONF_SDRC_D30	RW	32	0x0000 003C	0x4800 206C	Section 6.6.3
CONTROL_PADCONF_SDRC_CLK	RW	32	0x0000 0040	0x4800 2070	Section 6.6.3
CONTROL_PADCONF_SDRC_CKE1	RW	32	0x0000 0234	0x4800 2264	Section 6.6.3
CONTROL_PADCONF_SDRC_DQS1	RW	32	0x0000 0044	0x4800 2074	Section 6.6.3
CONTROL_PADCONF_SDRC_DQS3	RW	32	0x0000 0048	0x4800 2078	Section 6.6.3
CONTROL_PADCONF_GPMC_A2	RW	32	0x0000 004C	0x4800 207C	Section 6.6.3
CONTROL_PADCONF_GPMC_A4	RW	32	0x0000 0050	0x4800 2080	Section 6.6.3
CONTROL_PADCONF_GPMC_A6	RW	32	0x0000 0054	0x4800 2084	Section 6.6.3
CONTROL_PADCONF_GPMC_A8	RW	32	0x0000 0058	0x4800 2088	Section 6.6.3
CONTROL_PADCONF_GPMC_A10	RW	32	0x0000 005C	0x4800 208C	Section 6.6.3
CONTROL_PADCONF_GPMC_D1	RW	32	0x0000 0060	0x4800 2090	Section 6.6.3
CONTROL_PADCONF_GPMC_D3	RW	32	0x0000 0064	0x4800 2094	Section 6.6.3
CONTROL_PADCONF_GPMC_D5	RW	32	0x0000 0068	0x4800 2098	Section 6.6.3
CONTROL_PADCONF_GPMC_D7	RW	32	0x0000 006C	0x4800 209C	Section 6.6.3
CONTROL_PADCONF_GPMC_D9	RW	32	0x0000 0070	0x4800 20A0	Section 6.6.3
CONTROL_PADCONF_GPMC_D11	RW	32	0x0000 0074	0x4800 20A4	Section 6.6.3
CONTROL_PADCONF_GPMC_D13	RW	32	0x0000 0078	0x4800 20A8	Section 6.6.3
CONTROL_PADCONF_GPMC_D15	RW	32	0x0000 007C	0x4800 20AC	Section 6.6.3
CONTROL_PADCONF_GPMC_NCS1	RW	32	0x0000 0080	0x4800 20B0	Section 6.6.3
CONTROL_PADCONF_GPMC_NCS3	RW	32	0x0000 0084	0x4800 20B4	Section 6.6.3
CONTROL_PADCONF_GPMC_NCS5	RW	32	0x0000 0088	0x4800 20B8	Section 6.6.3
CONTROL_PADCONF_GPMC_NCS7	RW	32	0x0000 008C	0x4800 20BC	Section 6.6.3
CONTROL_PADCONF_GPMC_NADV_ALE	RW	32	0x0000 0090	0x4800 20C0	Section 6.6.3
CONTROL_PADCONF_GPMC_NWE	RW	32	0x0000 0094	0x4800 20C4	Section 6.6.3
CONTROL_PADCONF_GPMC_NBE1	RW	32	0x0000 0098	0x4800 20C8	Section 6.6.3
CONTROL_PADCONF_GPMC_WAIT0	RW	32	0x0000 009C	0x4800 20CC	Section 6.6.3
CONTROL_PADCONF_GPMC_WAIT2	RW	32	0x0000 00A0	0x4800 20D0	Section 6.6.3
CONTROL_PADCONF_DSS_PCLK	RW	32	0x0000 00A4	0x4800 20D4	Section 6.6.3
CONTROL_PADCONF_DSS_VSYNC	RW	32	0x0000 00A8	0x4800 20D8	Section 6.6.3
CONTROL_PADCONF_DSS_DATA0	RW	32	0x0000 00AC	0x4800 20DC	Section 6.6.3
CONTROL_PADCONF_DSS_DATA2	RW	32	0x0000 00B0	0x4800 20E0	Section 6.6.3
CONTROL_PADCONF_DSS_DATA4	RW	32	0x0000 00B4	0x4800 20E4	Section 6.6.3

Table 6-49. PADCONFS Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_PADCONF_DSS_DATA6	RW	32	0x0000 00B8	0x4800 20E8	Section 6.6.3
CONTROL_PADCONF_DSS_DATA8	RW	32	0x0000 00BC	0x4800 20EC	Section 6.6.3
CONTROL_PADCONF_DSS_DATA10	RW	32	0x0000 00C0	0x4800 20F0	Section 6.6.3
CONTROL_PADCONF_DSS_DATA12	RW	32	0x0000 00C4	0x4800 20F4	Section 6.6.3
CONTROL_PADCONF_DSS_DATA14	RW	32	0x0000 00C8	0x4800 20F8	Section 6.6.3
CONTROL_PADCONF_DSS_DATA16	RW	32	0x0000 00CC	0x4800 20FC	Section 6.6.3
CONTROL_PADCONF_DSS_DATA18	RW	32	0x0000 00D0	0x4800 2100	Section 6.6.3
CONTROL_PADCONF_DSS_DATA20	RW	32	0x0000 00D4	0x4800 2104	Section 6.6.3
CONTROL_PADCONF_DSS_DATA22	RW	32	0x0000 00D8	0x4800 2108	Section 6.6.3
CONTROL_PADCONF_MCBSP2_FSX	RW	32	0x0000 010C	0x4800 213C	Section 6.6.3
CONTROL_PADCONF_MCBSP2_DR	RW	32	0x0000 0110	0x4800 2140	Section 6.6.3
CONTROL_PADCONF_MMC1_CLK	RW	32	0x0000 0114	0x4800 2144	Section 6.6.3
CONTROL_PADCONF_MMC1_DAT0	RW	32	0x0000 0118	0x4800 2148	Section 6.6.3
CONTROL_PADCONF_MMC1_DAT2	RW	32	0x0000 011C	0x4800 214C	Section 6.6.3
CONTROL_PADCONF_MMC1_DAT4	RW	32	0x0000 0120	0x4800 2150	Section 6.6.3
CONTROL_PADCONF_MMC1_DAT6	RW	32	0x0000 0124	0x4800 2154	Section 6.6.3
CONTROL_PADCONF_MMC2_CLK	RW	32	0x0000 0128	0x4800 2158	Section 6.6.3
CONTROL_PADCONF_MMC2_DAT0	RW	32	0x0000 012C	0x4800 215C	Section 6.6.3
CONTROL_PADCONF_MMC2_DAT2	RW	32	0x0000 0130	0x4800 2160	Section 6.6.3
CONTROL_PADCONF_MMC2_DAT4	RW	32	0x0000 0134	0x4800 2164	Section 6.6.3
CONTROL_PADCONF_MMC2_DAT6	RW	32	0x0000 0138	0x4800 2168	Section 6.6.3
CONTROL_PADCONF_MCBSP3_DX	RW	32	0x0000 013C	0x4800 216C	Section 6.6.3
CONTROL_PADCONF_MCBSP3_CLKX	RW	32	0x0000 0140	0x4800 2170	Section 6.6.3
CONTROL_PADCONF_UART2_CTS	RW	32	0x0000 0144	0x4800 2174	Section 6.6.3
CONTROL_PADCONF_UART2_TX	RW	32	0x0000 0148	0x4800 2178	Section 6.6.3
CONTROL_PADCONF_UART1_TX	RW	32	0x0000 014C	0x4800 217C	Section 6.6.3
CONTROL_PADCONF_UART1_CTS	RW	32	0x0000 0150	0x4800 2180	Section 6.6.3
CONTROL_PADCONF_MCBSP4_CLKX	RW	32	0x0000 0154	0x4800 2184	Section 6.6.3
CONTROL_PADCONF_MCBSP4_DX	RW	32	0x0000 0158	0x4800 2188	Section 6.6.3
CONTROL_PADCONF_MCBSP1_CLKR	RW	32	0x0000 015C	0x4800 218C	Section 6.6.3
CONTROL_PADCONF_MCBSP1_DX	RW	32	0x0000 0160	0x4800 2190	Section 6.6.3
CONTROL_PADCONF_MCBSP_CLKS	RW	32	0x0000 0164	0x4800 2194	Section 6.6.3
CONTROL_PADCONF_MCBSP1_CLKX	RW	32	0x0000 0168	0x4800 2198	Section 6.6.3
CONTROL_PADCONF_UART3_RTS_SD	RW	32	0x0000 016C	0x4800 219C	Section 6.6.3
CONTROL_PADCONF_UART3_TX_IRTX	RW	32	0x0000 0170	0x4800 21A0	Section 6.6.3
CONTROL_PADCONF_I2C1_SCL	RW	32	0x0000 0188	0x4800 21B8	Section 6.6.3
CONTROL_PADCONF_I2C1_SDA	RW	32	0x0000 018C	0x4800 21BC	Section 6.6.3
CONTROL_PADCONF_I2C2_SDA	RW	32	0x0000 0190	0x4800 21C0	Section 6.6.3
CONTROL_PADCONF_I2C3_SDA	RW	32	0x0000 0194	0x4800 21C4	Section 6.6.3
CONTROL_PADCONF_MCSPI1_CLK	RW	32	0x0000 0198	0x4800 21C8	Section 6.6.3
CONTROL_PADCONF_MCSPI1_SOMI	RW	32	0x0000 019C	0x4800 21CC	Section 6.6.3
CONTROL_PADCONF_MCSPI1_CS1	RW	32	0x0000 01A0	0x4800 21D0	Section 6.6.3
CONTROL_PADCONF_MCSPI1_CS3	RW	32	0x0000 01A4	0x4800 21D4	Section 6.6.3
CONTROL_PADCONF_MCSPI2_SIMO	RW	32	0x0000 01A8	0x4800 21D8	Section 6.6.3
CONTROL_PADCONF_MCSPI2_CS0	RW	32	0x0000 01AC	0x4800 21DC	Section 6.6.3
CONTROL_PADCONF_SYS_NIRQ	RW	32	0x0000 01B0	0x4800 21E0	Section 6.6.3
CONTROL_PADCONF_ETK_CLK	RW	32	0x0000 05D8	0x4800 25D8	Section 6.6.3

Table 6-49. PADCONFS Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_PADCONF_ETK_D0	RW	32	0x0000 05DC	0x4800 25DC	Section 6.6.3
CONTROL_PADCONF_ETK_D2	RW	32	0x0000 05E0	0x4800 25E0	Section 6.6.3
CONTROL_PADCONF_ETK_D4	RW	32	0x0000 05E4	0x4800 25E4	Section 6.6.3
CONTROL_PADCONF_ETK_D6	RW	32	0x0000 05E8	0x4800 25E8	Section 6.6.3
CONTROL_PADCONF_ETK_D8	RW	32	0x0000 05EC	0x4800 25EC	Section 6.6.3
CONTROL_PADCONF_ETK_D10	RW	32	0x0000 05F0	0x4800 25F0	Section 6.6.3
CONTROL_PADCONF_ETK_D12	RW	32	0x0000 05F4	0x4800 25F4	Section 6.6.3
CONTROL_PADCONF_ETK_D14	RW	32	0x0000 05F8	0x4800 25F8	Section 6.6.3
CONTROL_PADCONF_CCDC_PCLK	RW	32	0x0000 01B4	0x4800 21E4	Section 6.6.3
CONTROL_PADCONF_CCDC_HD	RW	32	0x0000 01B8	0x4800 21E8	Section 6.6.3
CONTROL_PADCONF_CCDC_WEN	RW	32	0x0000 01BC	0x4800 21EC	Section 6.6.3
CONTROL_PADCONF_CCDC_DATA1	RW	32	0x0000 01C0	0x4800 21F0	Section 6.6.3
CONTROL_PADCONF_CCDC_DATA3	RW	32	0x0000 01C4	0x4800 21F4	Section 6.6.3
CONTROL_PADCONF_CCDC_DATA5	RW	32	0x0000 01C8	0x4800 21F8	Section 6.6.3
CONTROL_PADCONF_CCDC_DATA7	RW	32	0x0000 01CC	0x4800 21FC	Section 6.6.3
CONTROL_PADCONF_RMII_MDIO_CLK	RW	32	0x0000 01D0	0x4800 2200	Section 6.6.3
CONTROL_PADCONF_RMII_RXD1	RW	32	0x0000 01D4	0x4800 2204	Section 6.6.3
CONTROL_PADCONF_RMII_RXER	RW	32	0x0000 01D8	0x4800 2208	Section 6.6.3
CONTROL_PADCONF_RMII_TXD1	RW	32	0x0000 01DC	0x4800 220C	Section 6.6.3
CONTROL_PADCONF_RMII_50MHZ_CLK	RW	32	0x0000 01E0	0x4800 2210	Section 6.6.3
CONTROL_PADCONF_HECC1_TXD	RW	32	0x0000 01E4	0x4800 2214	Section 6.6.3
CONTROL_PADCONF_SYS_BOOT7	RW	32	0x0000 01E8	0x4800 2218	Section 6.6.3
CONTROL_PADCONF_SDRC_DQS1N	RW	32	0x0000 01EC	0x4800 221C	Section 6.6.3
CONTROL_PADCONF_SDRC_DQS3N	RW	32	0x0000 01F0	0x4800 2220	Section 6.6.3
CONTROL_PADCONF_SDRC_STRBEN_DLY1	RW	32	0x0000 01F4	0x4800 2224	Section 6.6.3
CONTROL_PADCONF_SDRC_CKE	RW	32	0x0000 0234	0x4800 2264	Section 6.6.3

Table 6-50 lists the GENERAL registers.

Table 6-50. GENERAL Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_PADCONF_OFF	RW	32	0x0000 0000	0x4800 2270	Section 6.6.4.1
CONTROL_DEVCONF0	RW	32	0x0000 0004	0x4800 2274	Section 6.6.4.2
CONTROL_MEM_DFTRW0	RW	32	0x0000 0008	0x4800 2278	Section 6.6.4.3
CONTROL_MEM_DFTRW1	RW	32	0x0000 000C	0x4800 227C	Section 6.6.4.4
CONTROL_MSUSPENDMUX_0	RW	32	0x0000 0020	0x4800 2290	Section 6.6.4.5
CONTROL_MSUSPENDMUX_1	RW	32	0x0000 0024	0x4800 2294	Section 6.6.4.6
CONTROL_MSUSPENDMUX_2	RW	32	0x0000 0028	0x4800 2298	Section 6.6.4.7
CONTROL_MSUSPENDMUX_4	RW	32	0x0000 0030	0x4800 22A0	Section 6.6.4.8
CONTROL_MSUSPENDMUX_5	RW	32	0x0000 0034	0x4800 22A4	Section 6.6.4.9
CONTROL_MSUSPENDMUX_6	RW	32	0x0000 0038	0x4800 22A8	Section 6.6.4.10
CONTROL_DEVCONF1	RW	32	0x0000 0068	0x4800 22D8	Section 6.6.4.11
CONTROL_SEC_STATUS	RW	32	0x0000 0070	0x4800 22E0	Section 6.6.4.12
CONTROL_SEC_ERR_STATUS	RW	32	0x0000 0074	0x4800 22E4	Section 6.6.4.13
CONTROL_SEC_ERR_STATUS_DEBUG	RW	32	0x0000 0078	0x4800 22E8	Section 6.6.4.14
CONTROL_STATUS	R	32	0x0000 0080	0x4800 22F0	Section 6.6.4.15

Table 6-50. GENERAL Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_RPUB_KEY_H_0	R	32	0x0000 0090	0x4800 2300	Section 6.6.4.16
CONTROL_RPUB_KEY_H_1	R	32	0x0000 0094	0x4800 2304	Section 6.6.4.17
CONTROL_RPUB_KEY_H_2	R	32	0x0000 0098	0x4800 2308	Section 6.6.4.18
CONTROL_RPUB_KEY_H_3	R	32	0x0000 009C	0x4800 230C	Section 6.6.4.19
CONTROL_RPUB_KEY_H_4	R	32	0x0000 00A0	0x4800 2310	Section 6.6.4.20
CONTROL_USB_CONF_0	R	32	0x0000 0100	0x4800 2370	Section 6.6.4.21
CONTROL_USB_CONF_1	R	32	0x0000 0104	0x4800 2374	Section 6.6.4.22
CONTROL_FUSE_EMAC_LSB	R	32	0x0000 0110	0x4800 2380	Section 6.6.4.23
CONTROL_FUSE_EMAC_MSB	R	32	0x0000 0114	0x4800 2384	Section 6.6.4.24
CONTROL_FUSE_SR	RW	32	0x0000 0130	0x4800 23A0	Section 6.6.4.25
CONTROL_CEK_0	RW	32	0x0000 0134	0x4800 23A4	Section 6.6.4.26
CONTROL_CEK_1	RW	32	0x0000 0138	0x4800 23A8	Section 6.6.4.27
CONTROL_CEK_2	RW	32	0x0000 013C	0x4800 23AC	Section 6.6.4.28
CONTROL_CEK_3	RW	32	0x0000 0140	0x4800 23B0	Section 6.6.4.29
CONTROL_MSV_0	RW	32	0x0000 0144	0x4800 23B4	Section 6.6.4.30
CONTROL_CEK_BCH_0	RW	32	0x0000 0148	0x4800 23B8	Section 6.6.4.31
CONTROL_CEK_BCH_1	RW	32	0x0000 014C	0x4800 23BC	Section 6.6.4.32
CONTROL_CEK_BCH_2	RW	32	0x0000 0150	0x4800 23C0	Section 6.6.4.33
CONTROL_CEK_BCH_3	RW	32	0x0000 0154	0x4800 23C4	Section 6.6.4.34
CONTROL_CEK_BCH_4	RW	32	0x0000 0158	0x4800 23C8	Section 6.6.4.35
CONTROL_MSV_BCH_0	RW	32	0x0000 015C	0x4800 23CC	Section 6.6.4.36
CONTROL_MSV_BCH_1	RW	32	0x0000 0160	0x4800 23D0	Section 6.6.4.37
CONTROL_SWRV_0	RW	32	0x0000 0164	0x4800 23D4	Section 6.6.4.38
CONTROL_SWRV_1	RW	32	0x0000 0168	0x4800 23D8	Section 6.6.4.39
CONTROL_SWRV_2	RW	32	0x0000 016C	0x4800 23DC	Section 6.6.4.40
CONTROL_SWRV_3	RW	32	0x0000 0170	0x4800 23E0	Table 24-6
CONTROL_SWRV_4	RW	32	0x0000 0174	0x4800 23E4	Section 6.6.4.42
CONTROL_DEBOBS_0	RW	32	0x0000 01B0	0x4800 2420	Section 6.6.4.43
CONTROL_DEBOBS_1	RW	32	0x0000 01B4	0x4800 2424	Section 6.6.4.44
CONTROL_DEBOBS_2	RW	32	0x0000 01B8	0x4800 2428	Section 6.6.4.45
CONTROL_DEBOBS_3	RW	32	0x0000 01BC	0x4800 242C	Section 6.6.4.46
CONTROL_DEBOBS_4	RW	32	0x0000 01C0	0x4800 2430	Section 6.6.4.47
CONTROL_DEBOBS_5	RW	32	0x0000 01C4	0x4800 2434	Section 6.6.4.48
CONTROL_DEBOBS_6	RW	32	0x0000 01C8	0x4800 2438	Section 6.6.4.49
CONTROL_DEBOBS_7	RW	32	0x0000 01CC	0x4800 243C	Section 6.6.4.50
CONTROL_DEBOBS_8	RW	32	0x0000 01D0	0x4800 2440	Section 6.6.4.51
CONTROL_WKUP_CTRL	RW	32	0x0000 0A5C	0x4800 2A5C	Section 6.6.4.52
CONTROL_DSS_DPLL_SPREADING	RW	32	0x0000 01E0	0x4800 2450	Section 6.6.4.53
CONTROL_CORE_DPLL_SPREADING	RW	32	0x0000 01E4	0x4800 2454	Section 6.6.4.54
CONTROL_PER_DPLL_SPREADING	RW	32	0x0000 01E8	0x4800 2458	Section 6.6.4.55
CONTROL_USBHOST_DPLL_SPREADING	RW	32	0x0000 01EC	0x4800 245C	Section 6.6.4.56
CONTROL_DPF_OCM_RAM_FW_ADDR_MATCH	RW	32	0x0000 0228	0x4800 2498	Section 9.1.7.2.26
CONTROL_DPF_OCM_RAM_FW_REQINFO	RW	32	0x0000 022C	0x4800 249C	Section 6.6.4.58
CONTROL_DPF_OCM_RAM_FW_WR	RW	32	0x0000 0230	0x4800 24A0	Section 6.6.4.59
CONTROL_DPF_REGION4_GPMC_FW_ADDR_MATCH	RW	32	0x0000 0234	0x4800 24A4	Section 6.6.4.60
CONTROL_DPF_REGION4_GPMC_FW_REQINFO	RW	32	0x0000 0238	0x4800 24A8	Section 6.6.4.61

Table 6-50. GENERAL Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_DPF_REGION4_GPMC_FW_WR	RW	32	0x0000 023C	0x4800 24AC	Section 6.6.4.62
CONTROL_APE_FW_DEFAULT_SECURE_LOCK	RW	32	0x0000 024C	0x4800 24BC	Section 6.6.4.63
CONTROL_OCMROM_SECURE_DEBUG	RW	32	0x0000 0250	0x4800 24C0	Section 6.6.4.64
CONTROL_EXT_SEC_CONTROL	RW	32	0x0000 0264	0x4800 24D4	Section 6.6.4.65
CONTROL_DEVCONF2	RW	32	0x0000 0310	0x4800 2580	Section 6.6.4.66
CONTROL_DEVCONF3	RW	32	0x0000 0314	0x4800 2584	Section 6.6.4.67
CONTROL_CBA_PRIORITY	RW	32	0x0000 0320	0x4800 2590	Section 6.6.4.68
CONTROL_LVL_INTR_CLEAR	RW	32	0x0000 0324	0x4800 2594	Section 6.6.4.69
CONTROL_IP_SW_RESET	RW	32	0x0000 0328	0x4800 2598	Section 6.6.4.70
CONTROL_IPSS_CLK_CTRL	RW	32	0x0000 032C	0x4800 259C	Section 6.6.4.71
CONTROL_IDCODE	RW	32	0x0030 7F94	0x4830 A204	Section 6.6.4.72

Table 6-51 lists the MEM_WKUP registers.

Table 6-51. MEM_WKUP Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_SAVE_RESTORE_MEM	RW	32	0x0600 - 0x09FC	0x4800 2600 - 0x4800 29FC	Section 6.6.5

Table 6-52 lists the PADCONFS_WKUP registers.

Table 6-52. PADCONFS_WKUP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_PADCONF_I2C4_SCL	RW	32	0x0000 0000	0x4800 2A00	Section 6.6.6
CONTROL_PADCONF_SYS_32K	RW	32	0x0000 0004	0x4800 2A04	Section 6.6.6
CONTROL_PADCONF_SYS_NRESWARM	RW	32	0x0000 0008	0x4800 2A08	Section 6.6.6
CONTROL_PADCONF_SYS_BOOT1	RW	32	0x0000 000C	0x4800 2A0C	Section 6.6.6
CONTROL_PADCONF_SYS_BOOT3	RW	32	0x0000 0010	0x4800 2A10	Section 6.6.6
CONTROL_PADCONF_SYS_BOOT5	RW	32	0x0000 0014	0x4800 2A14	Section 6.6.6
CONTROL_PADCONF_SYS_OFF_MODE	RW	32	0x0000 0018	0x4800 2A18	Section 6.6.6
CONTROL_PADCONF_JTAG_NTRST	RW	32	0x0000 001C	0x4800 2A1C	Section 6.6.6
CONTROL_PADCONF_JTAG_TMS_TMSC	RW	32	0x0000 0020	0x4800 2A20	Section 6.6.6
CONTROL_PADCONF_JTAG_EMU0	RW	32	0x0000 0024	0x4800 2A24	Section 6.6.6
CONTROL_PADCONF_JTAG_RTCK	RW	32	0x0000 004C	0x4800 2A4C	Section 6.6.6
CONTROL_PADCONF_JTAG_TDO	RW	32	0x0000 0050	0x4800 2A50	Section 6.6.6

Table 6-53 lists the GENERAL_WKUP registers.

Table 6-53. GENERAL_WKUP Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_SEC_TAP	RW	32	0x0000 0000	0x4800 2A60	Section 6.6.7.1
CONTROL_SEC_EMU	RW	32	0x0000 0004	0x4800 2A64	Section 6.6.7.2
CONTROL_WKUP_DEBOBS_0	RW	32	0x0000 0008	0x4800 2A68	Section 6.6.7.3
CONTROL_WKUP_DEBOBS_1	RW	32	0x0000 000C	0x4800 2A6C	Section 6.6.7.4
CONTROL_WKUP_DEBOBS_2	RW	32	0x0000 0010	0x4800 2A70	Section 6.6.7.5

Table 6-53. GENERAL_WKUP Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
CONTROL_WKUP_DEBOBS_3	RW	32	0x0000 0014	0x4800 2A74	Section 6.6.7.6
CONTROL_WKUP_DEBOBS_4	RW	32	0x0000 0018	0x4800 2A78	Section 6.6.7.7
CONTROL_SEC_DAP	RW	32	0x0000 001C	0x4800 2A7C	Section 6.6.7.8

6.6.2 INTERFACE Register Descriptions

Table 6-150 through Table 6-55 describe the interface register bits.

6.6.2.1 CONTROL_REVISION

Table 6-54. CONTROL_REVISION

Address Offset	0x00		
Physical address	0x4800 2000	**Instance**	INTERFACE
Description	Control module Revision number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REVISION							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Read returns reset value.	R	0x000000
7:0	REVISION	Revision number	R	-

6.6.2.2 Control System Configuration Register (CONTROL_SYSCONFIG)

Table 6-55. Control System Configuration Register (CONTROL_SYSCONFIG)

Address Offset	0x10		
Physical address	0x4800 2010	**Instance**	INTERFACE
Description	Set various parameters relative to the Idle mode of the Control module		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											IDLEMODE			ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Read returns reset value.	R	0x0000000
4:3	IDLEMODE	Power Management, req/ack control	R/W	0x0
		0x0: Force-idle. An idle request is acknowledged unconditionally		
		0x1: Reserved		
		0x2: Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module		
		0x3: Reserved		
2	ENAWAKEUP	Wake-up enable. Not used in the module	R	0x0
1	SOFTRESET	Software reset. Not used in the module	R	0x0
0	AUTOIDLE	Internal interface clock gating strategy	R/W	0x0
		0x0: Interface clock is free-running		
		0x1: Automatic interface clock gating strategy is applied, based on the interconnect interface activity		

6.6.3 PADCONFS Register Description

Each 32-bit PADCONF register gathers the configuration of two pads. For example, CONTROL.CONTROL_PADCONF_GPMC_7 is used to configure gpmc_7 pad (bits [15:0]) and gpmc_8 pad (bits [31:16]). See Figure 6-8 for more information about PADCONF registers.

According to the pad type, some features are configurable or not. Table 6-56 gives the description of a fully configurable pad.

Table 6-56. CONTROL_PADCONF_X

Address Offset	0x0000 0000 - 0x0000 05C8		
Physical address	See Table 6-49	**Instance**	SYSC_PADCONFS
Description	Pad configuration register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							INPUTENABLE1	RESERVED			PULLTYPESELECT1	PULLUDENABLE1	MUXMODE1			RESERVED							INPUTENABLE0	RESERVED			PULLTYPESELECT0	PULLUDENABLE0	MUXMODE0		

Bits	Field Name	Description	Type	Reset
31:25	RESERVED	Reserved	R	0
24	INPUTENABLE1	Input enable value for pad_x	R/W	1
23:21	RESERVED	Reserved	R	0
20	PULLTYPESELECT1	Pull-Up/Down selection for pad_x: 0: Pull-Down selected 1: Pull-Up selected	R/W	Pad dependent
19	PULLUDENABLE1	Pull-Up/Down enable for pad_x: 0: Pull-Up/Down disabled 1: Pull-Up/Down enabled	R/W	Pad dependent
18:16	MUXMODE1	Functional multiplexing selection for pad_x	R/W	Pad dependent
15:9	RESERVED	Reserved	R	0
8	INPUTENABLE0	Input enable value for pad_y	R/W	1
7:5	RESERVED	Reserved	R	0
4	PULLTYPESELECT0	Pull-Up/Down selection for pad_y: 0: Pull-Down selected 1: Pull-Up selected	R/W	Pad dependent
3	PULLUDENABLE0	Pull-Up/Down enable for pad_y: 0: Pull-Up/Down disabled 1: Pull-Up/Down enabled	R/W	Pad dependent
2:0	MUXMODE0	Functional multiplexing selection for pad_y	R/W	Pad dependent

NOTE: The Bits field gives the field number for the pairs of pads gathered in each register.

Table 6-57 describes the reset values and the corresponding register for each pad.

NOTE:

- All '-' assume that the corresponding bit is not available. These bits are considered as Reserved.
- The reset value for Reserved bits is 0.
- In the MUX Reset States column and the PU/PD Reset States column of Table 6-57 , a dash (-) indicates that the field is hardwired to logic 0. No corresponding control block ports are implemented for these bits.

Table 6-57. CONTROL_PADCONF_CAPABILITIES

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_SDRC_D0[15:0]	sdrc_d0	0x4800 2030	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D0[31:16]	sdrc_d1	0x4800 2030	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D2[15:0]	sdrc_d2	0x4800 2034	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D2[31:16]	sdrc_d3	0x4800 2034	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D4[15:0]	sdrc_d4	0x4800 2038	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D4[31:16]	sdrc_d5	0x4800 2038	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D6[15:0]	sdrc_d6	0x4800 203C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D6[31:16]	sdrc_d7	0x4800 203C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D8[15:0]	sdrc_d8	0x4800 2040	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D8[31:16]	sdrc_d9	0x4800 2040	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D10[15:0]	sdrc_d10	0x4800 2044	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D10[31:16]	sdrc_d11	0x4800 2044	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D12[15:0]	sdrc_d12	0x4800 2048	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D12[31:16]	sdrc_d13	0x4800 2048	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D14[15:0]	sdrc_d14	0x4800 204C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D14[31:16]	sdrc_d15	0x4800 204C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D16[15:0]	sdrc_d16	0x4800 2050	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D16[31:16]	sdrc_d17	0x4800 2050	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D18[15:0]	sdrc_d18	0x4800 2054	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D18[31:16]	sdrc_d19	0x4800 2054	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D20[15:0]	sdrc_d20	0x4800 2058	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D20[31:16]	sdrc_d21	0x4800 2058	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D22[15:0]	sdrc_d22	0x4800 205C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D22[31:16]	sdrc_d23	0x4800 205C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D24[15:0]	sdrc_d24	0x4800 2060	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D24[31:16]	sdrc_d25	0x4800 2060	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D26[15:0]	sdrc_d26	0x4800 2064	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D26[31:16]	sdrc_d27	0x4800 2064	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D28[15:0]	sdrc_d28	0x4800 2068	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D28[31:16]	sdrc_d29	0x4800 2068	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D30[15:0]	sdrc_d30	0x4800 206C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_D30[31:16]	sdrc_d31	0x4800 206C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_CKE0[31:16]	sdrc_cke0	0x4800 2264	0b1	0b000	0b11	---
CONTROL_PADCONF_SDRC_CLK[15:0]	sdrc_clk	0x4800 2070	0b1	0b000	0b00	---

Table 6-57. CONTROL_PADCONF_CAPABILITIES (continued)

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_SDRC_CLK[31:16]	sdrc_dqs0p	0x4800 2070	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS1[15:0]	sdrc_dqs1p	0x4800 2074	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS1[31:16]	sdrc_dqs2p	0x4800 2074	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS3[15:0]	sdrc_dqs3p	0x4800 2078	0b1	0b000	0b00	---
CONTROL_PADCONF_SYS_BOOT7[15:0]	sdrc_dqs0n	0x4800 2218	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS1N[15:0]	sdrc_dqs1n	0x4800 221C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS1N[31:16]	sdrc_dqs2n	0x4800 221C	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS3N[15:0]	sdrc_dqs3n	0x4800 2220	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS3N[31:16]	sdrc_strben_dly0	0x4800 2220	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_STRBEN_DLY1[15:0]	sdrc_strben_dly1	0x4800 2224	0b1	0b000	0b00	---
CONTROL_PADCONF_SDRC_DQS3[31:16]	gpmc_a1	0x4800 2078	0b1	0b000	0b01	0b111
CONTROL_PADCONF_GPMC_A2[15:0]	gpmc_a2	0x4800 207C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_GPMC_A2[31:16]	gpmc_a3	0x4800 207C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_GPMC_A4[15:0]	gpmc_a4	0x4800 2080	0b1	0b000	0b01	0b111
CONTROL_PADCONF_GPMC_A4[31:16]	gpmc_a5	0x4800 2080	0b1	0b000	0b01	0b111
CONTROL_PADCONF_GPMC_A6[15:0]	gpmc_a6	0x4800 2084	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_A6[31:16]	gpmc_a7	0x4800 2084	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_A8[15:0]	gpmc_a8	0x4800 2088	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_A8[31:16]	gpmc_a9	0x4800 2088	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_A10[15:0]	gpmc_a10	0x4800 208C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_A10[31:16]	gpmc_d0	0x4800 208C	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D1[15:0]	gpmc_d1	0x4800 2090	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D1[31:16]	gpmc_d2	0x4800 2090	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D3[15:0]	gpmc_d3	0x4800 2094	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D3[31:16]	gpmc_d4	0x4800 2094	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D5[15:0]	gpmc_d5	0x4800 2098	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D5[31:16]	gpmc_d6	0x4800 2098	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D7[15:0]	gpmc_d7	0x4800 209C	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_D7[31:16]	gpmc_d8	0x4800 209C	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D9[15:0]	gpmc_d9	0x4800 20A0	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D9[31:16]	gpmc_d10	0x4800 20A0	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D11[15:0]	gpmc_d11	0x4800 20A4	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D11[31:16]	gpmc_d12	0x4800 20A4	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D13[15:0]	gpmc_d13	0x4800 20A8	0b1	0b000	0b11	0b000

Table 6-57. CONTROL_PADCONF_CAPABILITIES (continued)

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_GPMC_D13[31:16]	gpmc_d14	0x4800 20A8	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D15[15:0]	gpmc_d15	0x4800 20AC	0b1	0b000	0b11	0b000
CONTROL_PADCONF_GPMC_D15[31:16]	gpmc_ncs0	0x4800 20AC	-	0b000	--	---
CONTROL_PADCONF_GPMC_NCS1[15:0]	gpmc_ncs1	0x4800 20B0	0b1	0b000	0b00	0b000
CONTROL_PADCONF_GPMC_NCS1[31:16]	gpmc_ncs2	0x4800 20B0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_NCS3[15:0]	gpmc_ncs3	0x4800 20B4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_NCS3[31:16]	gpmc_ncs4	0x4800 20B4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_NCS5[15:0]	gpmc_ncs5	0x4800 20B8	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_NCS5[31:16]	gpmc_ncs6	0x4800 20B8	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_NCS7[15:0]	gpmc_ncs7	0x4800 20BC	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_NCS7[31:16]	gpmc_clk	0x4800 20BC	0b1	0b000	0b00	0b000
CONTROL_PADCONF_GPMC_NADV_ALE[15:0]	gpmc_nadv_ale	0x4800 20C0	-	0b000	--	---
CONTROL_PADCONF_GPMC_NADV_ALE[31:16]	gpmc_noe	0x4800 20C0	-	0b000	--	---
CONTROL_PADCONF_GPMC_NWE[15:0]	gpmc_nwe	0x4800 20C4	-	0b000	--	---
CONTROL_PADCONF_GPMC_NWE[31:16]	gpmc_nbe0_cle	0x4800 20C4	0b1	0b000	0b00	0b000
CONTROL_PADCONF_GPMC_NBE1[15:0]	gpmc_nbe1	0x4800 20C8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_GPMC_NBE1[31:16]	gpmc_nwp	0x4800 20C8	0b1	0b000	0b00	0b000
CONTROL_PADCONF_GPMC_WAIT0[15:0]	gpmc_wait0	0x4800 20CC	0b1	0b000	0b11	---
CONTROL_PADCONF_GPMC_WAIT0[31:16]	gpmc_wait1	0x4800 20CC	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_WAIT2[15:0]	gpmc_wait2	0x4800 20D0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_GPMC_WAIT2[31:16]	gpmc_wait3	0x4800 20D0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_DSS_PCLK[15:0]	dss_pclk	0x4800 20D4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_DSS_PCLK[31:16]	dss_hsync	0x4800 20D4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_DSS_VSYNC[15:0]	dss_vsync	0x4800 20D8	0b1	0b000	0b11	0b111
CONTROL_PADCONF_DSS_VSYNC[31:16]	dss_acbias	0x4800 20D8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA0[15:0]	dss_data0	0x4800 20DC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA0[31:16]	dss_data1	0x4800 20DC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA2[15:0]	dss_data2	0x4800 20E0	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA2[31:16]	dss_data3	0x4800 20E0	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA4[15:0]	dss_data4	0x4800 20E4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA4[31:16]	dss_data5	0x4800 20E4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA6[15:0]	dss_data6	0x4800 20E8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA6[31:16]	dss_data7	0x4800 20E8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA8[15:0]	dss_data8	0x4800 20EC	0b1	0b000	0b01	0b111

Table 6-57. CONTROL_PADCONF_CAPABILITIES (continued)

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_DSS_DATA8[31:16]	dss_data9	0x4800 20EC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA10[15:0]	dss_data10	0x4800 20F0	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA10[31:16]	dss_data11	0x4800 20F0	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA12[15:0]	dss_data12	0x4800 20F4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA12[31:16]	dss_data13	0x4800 20F4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA14[15:0]	dss_data14	0x4800 20F8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA14[31:16]	dss_data15	0x4800 20F8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA16[15:0]	dss_data16	0x4800 20FC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA16[31:16]	dss_data17	0x4800 20FC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA18[15:0]	dss_data18	0x4800 2100	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA18[31:16]	dss_data19	0x4800 2100	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA20[15:0]	dss_data20	0x4800 2104	0b1	0b000	0b11	0b111
CONTROL_PADCONF_DSS_DATA20[31:16]	dss_data21	0x4800 2104	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA22[15:0]	dss_data22	0x4800 2108	0b1	0b000	0b01	0b111
CONTROL_PADCONF_DSS_DATA22[31:16]	dss_data23	0x4800 2108	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_PCLK[15:0]	ccdc_pclk	0x4800 21E4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_PCLK[31:16]	ccdc_field	0x4800 21E4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_HD[15:0]	ccdc_hd	0x4800 21E8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_HD[31:16]	ccdc_vd	0x4800 21E8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_WEN[15:0]	ccdc_wen	0x4800 21EC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_WEN[31:16]	ccdc_data0	0x4800 21EC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA1[15:0]	ccdc_data1	0x4800 21F0	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA1[31:16]	ccdc_data2	0x4800 21F0	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA3[15:0]	ccdc_data3	0x4800 21F4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA3[31:16]	ccdc_data4	0x4800 21F4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA5[15:0]	ccdc_data5	0x4800 21F8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA5[31:16]	ccdc_data6	0x4800 21F8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA7[15:0]	ccdc_data7	0x4800 21FC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_CCDC_DATA7[31:16]	rmii_mdio_data	0x4800 21FC	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_MDIO_CLK[15:0]	rmii_mdio_clk	0x4800 2200	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_MDIO_CLK[31:16]	rmii_rxd0	0x4800 2200	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_RXD1[15:0]	rmii_rxd1	0x4800 2204	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_RXD1[31:16]	rmii_crs_dv	0x4800 2204	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_RXER[15:0]	rmii_rxer	0x4800 2208	0b1	0b000	0b11	0b111

Table 6-57. CONTROL_PADCONF_CAPABILITIES (continued)

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_RMII_RXER[31:16]	rmii_txd0	0x4800 2208	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_TXD1[15:0]	rmii_txd1	0x4800 220C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_TXD1[31:16]	rmii_txen	0x4800 220C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_50MHZ_CLK[15:0]	rmii_50mhz_clk	0x4800 2210	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MCBSP2_FSX[31:16]	mcbsp2_clkx	0x4800 213C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP2_DR[15:0]	mcbsp2_dr	0x4800 2140	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP2_DR[31:16]	mcbsp2_dx	0x4800 2140	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_CLK[15:0]	mmc1_clk	0x4800 2144	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_CLK[31:16]	mmc1_cmd	0x4800 2144	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT0[15:0]	mmc1_dat0	0x4800 2148	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT0[31:16]	mmc1_dat1	0x4800 2148	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT2[15:0]	mmc1_dat2	0x4800 214C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT2[31:16]	mmc1_dat3	0x4800 214C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT4[15:0]	mmc1_dat4	0x4800 2150	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT4[31:16]	mmc1_dat5	0x4800 2150	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT6[15:0]	mmc1_dat6	0x4800 2154	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC1_DAT6[31:16]	mmc1_dat7	0x4800 2154	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC2_CLK[15:0]	mmc2_clk	0x4800 2158	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC2_CLK[31:16]	mmc2_cmd	0x4800 2158	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MMC2_DAT0[15:0]	mmc2_dat0	0x4800 215C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MMC2_DAT0[31:16]	mmc2_dat1	0x4800 215C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MMC2_DAT2[15:0]	mmc2_dat2	0x4800 2160	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MMC2_DAT2[31:16]	mmc2_dat3	0x4800 2160	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MMC2_DAT4[15:0]	mmc2_dat4	0x4800 2164	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC2_DAT4[31:16]	mmc2_dat5	0x4800 2164	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC2_DAT6[15:0]	mmc2_dat6	0x4800 2168	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MMC2_DAT6[31:16]	mmc2_dat7	0x4800 2168	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP3_DX[15:0]	mcbsp3_dx	0x4800 216C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP3_DX[31:16]	mcbsp3_dr	0x4800 216C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP3_CLKX[15:0]	mcbsp3_clkx	0x4800 2170	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP3_CLKX[31:16]	mcbsp3_fsx	0x4800 2170	0b1	0b000	0b01	0b111
CONTROL_PADCONF_UART2_CTS[15:0]	uart2_cts	0x4800 2174	0b1	0b000	0b11	0b111
CONTROL_PADCONF_UART2_CTS[31:16]	uart2_rts	0x4800 2174	0b1	0b000	0b11	0b111
CONTROL_PADCONF_UART2_TX[15:0]	uart2_tx	0x4800 2178	0b1	0b000	0b11	0b111

Table 6-57. CONTROL_PADCONF_CAPABILITIES (continued)

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_UART2_TX[31:16]	uart2_rx	0x4800 2178	0b1	0b000	0b11	0b111
CONTROL_PADCONF_UART1_TX[15:0]	uart1_tx	0x4800 217C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_UART1_TX[31:16]	uart1_rts	0x4800 217C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_UART1_CTS[15:0]	uart1_cts	0x4800 2180	0b1	0b000	0b01	0b111
CONTROL_PADCONF_UART1_CTS[31:16]	uart1_rx	0x4800 2180	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP4_CLKX[15:0]	mcbsp4_clkx	0x4800 2184	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP4_CLKX[31:16]	mcbsp4_dr	0x4800 2184	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP4_DX[15:0]	mcbsp4_dx	0x4800 2188	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP4_DX[31:16]	mcbsp4_fsx	0x4800 2188	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP1_CLKR[15:0]	mcbsp1_clkr	0x4800 218C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP1_CLKR[31:16]	mcbsp1_fsr	0x4800 218C	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP1_DX[15:0]	mcbsp1_dx	0x4800 2190	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP1_DX[31:16]	mcbsp1_dr	0x4800 2190	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP_CLKS[15:0]	mcbsp_clks	0x4800 2194	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP_CLKS[31:16]	mcbsp1_fsx	0x4800 2194	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP1_CLKX[15:0]	mcbsp1_clkx	0x4800 2198	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCBSP1_CLKX[31:16]	uart3_cts_rctx	0x4800 2198	0b1	0b000	0b11	0b111
CONTROL_PADCONF_UART3_RTS_SD[15:0]	uart3_rts_sd	0x4800 219C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_UART3_RTS_SD[31:16]	uart3_rx_irrx	0x4800 219C	0b1	0b000	0b11	0b111
CONTROL_PADCONF_UART3_TX_IRTX[15:0]	uart3_tx_irtx	0x4800 21A0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_RMII_50MHZ_CLK[31:16]	usb0_drvvbus	0x4800 2210	0b1	0b000	0b01	0b111
CONTROL_PADCONF_HECC1_TXD[15:0]	hecc1_txd	0x4800 2214	0b1	0b000	0b11	0b111
CONTROL_PADCONF_HECC1_TXD[31:16]	hecc1_rxd	0x4800 2214	0b1	0b000	0b01	0b111
CONTROL_PADCONF_I2C1_SCL[31:16]	i2c1_scl	0x4800 21B8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_I2C1_SDA[15:0]	i2c1_sda	0x4800 21BC	0b1	0b000	0b11	---
CONTROL_PADCONF_I2C1_SDA[31:16]	i2c2_scl	0x4800 21BC	0b1	0b000	0b11	0b111
CONTROL_PADCONF_I2C2_SDA[15:0]	i2c2_sda	0x4800 21C0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_I2C2_SDA[31:16]	i2c3_scl	0x4800 21C0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_I2C3_SDA[15:0]	i2c3_sda	0x4800 21C4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_I2C3_SDA[31:16]	hdq_sio	0x4800 21C4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MCSPI1_CLK[15:0]	mcspi1_clk	0x4800 21C8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCSPI1_CLK[31:16	mcspi1_simo	0x4800 21C8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCSPI1_SOMI[15:0]	mcspi1_somi	0x4800 21CC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCSPI1_SOMI[31:16]	mcspi1_cs0	0x4800 21CC	0b1	0b000	0b11	0b111

Table 6-57. CONTROL_PADCONF_CAPABILITIES (continued)

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MuxMode
CONTROL_PADCONF_MCSPI1_CS1[15:0]	mcspi1_cs1	0x4800 21D0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MCSPI1_CS1[31:16]	mcspi1_cs2	0x4800 21D0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MCSPI1_CS3[15:0]	mcspi1_cs3	0x4800 21D4	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MCSPI1_CS3[31:16]	mcspi2_clk	0x4800 21D4	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCSPI2_SIMO[15:0]	mcspi2_simo	0x4800 21D8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCSPI2_SIMO[31:16]	mcspi2_somi	0x4800 21D8	0b1	0b000	0b01	0b111
CONTROL_PADCONF_MCSPI2_CS0[15:0]	mcspi2_cs0	0x4800 21DC	0b1	0b000	0b11	0b111
CONTROL_PADCONF_MCSPI2_CS0[31:16]	mcspi2_cs1	0x4800 21DC	0b1	0b000	0b01	0b111
CONTROL_PADCONF_SYS_NIRQ[15:0]	sys_nirq	0x4800 21E0	0b1	0b000	0b11	0b111
CONTROL_PADCONF_ETK_CLK[15:0]	etk_clk	0x4800 25D8	0b1	0b000	0b11	0b100
CONTROL_PADCONF_ETK_CLK[31:16]	etk_ctl	0x4800 25D8	0b1	0b000	0b11	0b100
CONTROL_PADCONF_ETK_D0[15:0]	etk_d0	0x4800 25DC	0b1	0b000	0b11	0b100
CONTROL_PADCONF_ETK_D0[31:16]	etk_d1	0x4800 25DC	0b1	0b000	0b11	0b100
CONTROL_PADCONF_ETK_D2[15:0]	etk_d2	0x4800 25E0	0b1	0b000	0b11	0b100
CONTROL_PADCONF_ETK_D2[31:16]	etk_d3	0x4800 25E0	0b1	0b000	0b11	0b100
CONTROL_PADCONF_ETK_D4[15:0]	etk_d4	0x4800 25E4	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D4[31:16]	etk_d5	0x4800 25E4	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D6[15:0]	etk_d6	0x4800 25E8	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D6[31:16]	etk_d7	0x4800 25E8	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D8[15:0]	etk_d8	0x4800 25EC	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D8[31:16]	etk_d9	0x4800 25EC	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D10[15:0]	etk_d10	0x4800 25F0	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D10[31:16]	etk_d11	0x4800 25F0	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D12[15:0]	etk_d12	0x4800 25F4	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D12[31:16]	etk_d13	0x4800 25F4	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D14[15:0]	etk_d14	0x4800 25F8	0b1	0b000	0b01	0b100
CONTROL_PADCONF_ETK_D14[31:16]	etk_d15	0x4800 25F8	0b1	0b000	0b01	0b100

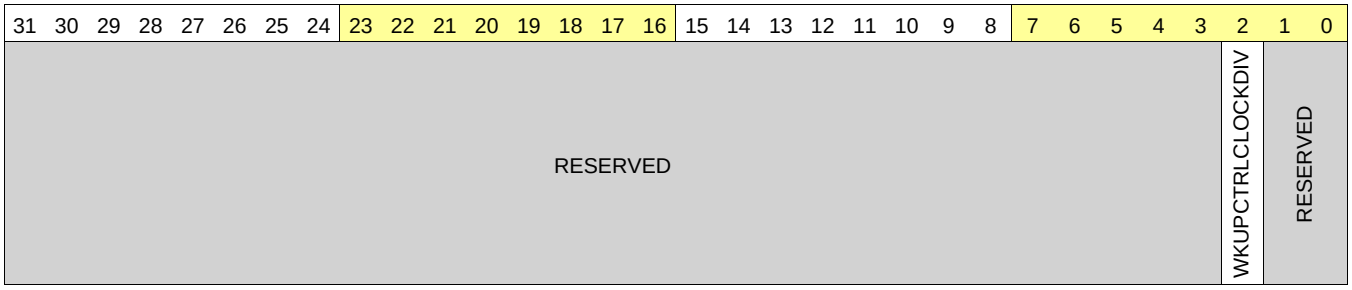
6.6.4 GENERAL Register Descriptions

Table 6-58 through Table 6-151 describe the GENERAL registers bits.

6.6.4.1 CONTROL_PADCONF_OFF

Table 6-58. CONTROL_PADCONF_OFF

Address Offset	0x0000 0000		
Physical address	0x4800 2270	**Instance**	GENERAL
Description	Off mode pad configuration register		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Read returns reset value.	R	0x0000000
2	WKUPCTRLCLOCKDIV	Wkup_ctrl module clock divider 0x0: Clock is divided by 4 0x1: Clock is divided by 2	R/W	0x0
1:0	RESERVED	Reserved	R	0x0

6.6.4.2 CONTROL_DEVCONF0

Table 6-59. CONTROL_DEVCONF0

Address Offset	0x0000 0004		
Physical address	0x4800 2274	**Instance**	GENERAL
Description	Static device configuration register-0. Module dedicated functions		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RESERVED																	MCBSP2_CLKS	RESERVED	MCBSP1_FSR	MCBSP1_CLKR	MCBSP1_CLKS	SENSDMAREQ1	SENSDMAREQ0

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Reserved bits	R/W	0x00
26	RESERVED	Reserved bit	R/W	0x1
25	RESERVED	Reserved bit	R/W	0x0
24:7	RESERVED	Read returns reset value	R	0x000
6	MCBSP2_CLKS	Select the CLKS input for the module McBSP2 Note : There are no external pins McBSP2_CLKR and McBSP2_FSR for the module McBSP2. For this module, CLKR input is from the pin McBSP2_CLKX and FSR input is from the pin McBSP2_FSX 0x0: CLKS is from the PRCM functional clock 0x1: CLKS is from the external pin McBSP_CLKS	R/W	0x0
5	RESERVED	Read returns reset value.	R	0
4	MCBSP1_FSR	Select the FSR input for the module McBSP1 0x0: FSR is from the pin McBSP1_FSR 0x1: FSR is from the pin McBSP1_FSX	R/W	0x0
3	MCBSP1_CLKR	Select the CLKR input for the module McBSP1 0x0: CLKR is from the pin McBSP1_CLKR 0x1: CLKR is from the pin McBSP1_CLKX	R/W	0x0
2	MCBSP1_CLKS	Select the CLKS input for the module McBSP1 0x0: CLKS is from the PRCM functional clock 0x1: CLKS is from the external pin McBSP_CLKS	R/W	0x0
1	SENSDMAREQ1	Set sensitivity on SYS_NDMAREQ1 input pin 0x0: Level sensitivity 0x1: Edge sensitivity	R/W	0x0
0	SENSDMAREQ0	Set sensitivity on SYS_NDMAREQ0 input pin 0x0: Level sensitivity 0x1: Edge sensitivity	R/W	0x0

6.6.4.3 CONTROL_MEM_DFTRW0

Table 6-60. CONTROL_MEM_DFTRW0

Address Offset	0x0000 0008		
Physical address	0x4800 2278	**Instance**	GENERAL
Description	DFT Read and Write Controls for memory blocks		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																MEMORY4DFTGLXCTRL	MEMORY3DFTGLXCTRL	MEMORY2DFTWRITECTRL		MEMORY2DFTREADCTRL		RESERVED		MEMORY1DFTWRITECTRL		MEMORY1DFTREADCTRL		MEMORY0DFTGLXCTRL		MEMORY0DFTWRITECTRL		MEMORY0DFTREADCTRL	

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns reset value	R	0X0
15	MEMORY4DFTGLXCTRL	ETB memory DFT GLX ctrl	Refer to Table 6-61	0X0
14	MEMORY3DFTGLXCTRL	WKUP memory DFT GLX ctrl	Refer to Table 6-61	0X0
13:12	MEMORY2DFTWRITECTRL	McBSP2 memory DFT write ctrl	Refer to Table 6-61	0X0
11:10	MEMORY2DFTREADCTRL	McBSP2 memory DFT read ctrl	Refer to Table 6-61	0X0
9	RESERVED	Read returns reset value	R	0X0
8:7	MEMORY1DFTWRITECTRL	EMAC, USBOTG ,HECC, and VPFE memory DFT write ctrl	Refer to Table 6-61	0X0
6:5	MEMORY1DFTREADCTRL	EMAC, USBOTG ,HECC, and VPFE memory DFT read ctrl	Refer to Table 6-61	0X0
4	MEMORY0DFTGLXCTRL	SGX_ss memory DFT GLX ctrl	Refer to Table 6-61	0X0
3:2	MEMORY0DFTWRITECTRL	SGX_ss memory DFT write ctrl	Refer to Table 6-61	0X0
1:0	MEMORY0DFTREADCTRL	SGX_ss memory DFT read ctrl	Refer to Table 6-61	0X0

Table 6-61. Type Value For CONTROL_MEM_DFTRW0 Register

Field Name	MPU Mode	
	NSP	SP
MEMORY4DFTGLXCTRL	Read returns 0s	R/W
MEMORY3DFTGLXCTRL	Read returns 0s	R/W
MEMORY2DFTWRITECTRL	Read returns 0s	R/W
MEMORY2DFTREADCTRL	Read returns 0s	R/W
MEMORY1DFTWRITECTRL	Read returns 0s	R/W
MEMORY1DFTREADCTRL	Read returns 0s	R/W
MEMORY0DFTGLXCTRL	Read returns 0s	R/W
MEMORY0DFTWRITECTRL	Read returns 0s	R/W

Table 6-61. Type Value For CONTROL_MEM_DFTRW0 Register (continued)

Field Name	MPU Mode	
	NSP	SP
MEMORY0DFTREADCTRL	Read returns 0s	R/W

6.6.4.4 CONTROL_MEM_DFTRW1

Table 6-62. CONTROL_MEM_DFTRW1

Address Offset	0x0000 000C		
Physical address	0x4800 227C	**Instance**	GENERAL
Description	DFT Read and Write Controls for memory blocks		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFTREADWRITEENABLE	RESERVED							MEMORY10DFTWRITECTRL		MEMORY10DFTREADCTRL		MEMORY9DFTWRITECTRL		MEMORY9DFTREADCTRL		RESERVED				MEMORY7DFTWRITECTRL		MEMORY7DFTREADCTRL		MEMORY6DFTGLXCTRL	RESERVED								

Bits	Field Name	Description	Type	Reset
31	DFTREADWRITEENABLE	Control use of CONTROL_MEM_DFTRWx registers: 0x0: DFT Read/write setting is coming from the Test sub-system 0x1: DFT Read/write setting is coming from the CONTROL_MEM_DFTRWx registers	Refer to Table 6-63	0
30:26	RESERVED	Read returns reset value.	R	0x0
25:24	MEMORY10DFTWRITECTRL	DSI memory DFT write ctrl	Refer to Table 6-63	0x0
23:22	MEMORY10DFTREADCTRL	DSI memory DFT read ctrl	Refer to Table 6-63	0x0
21:20	MEMORY9DFTWRITECTRL	DISP_ss memory DFT write ctrl	Refer to Table 6-63	0x0
19:18	MEMORY9DFTREADCTRL	DISP_ss memory DFT read ctrl	Refer to Table 6-63	0x0
17:13	RESERVED	Reserved	R	0x0
12:11	MEMORY7DFTWRITECTRL	MPU_ss memory DFT write ctrl	Refer to Table 6-63	0x0
10:9	MEMORY7DFTREADCTRL	MPU_ss memory DFT read ctrl	Refer to Table 6-63	0x0
8	MEMORY6DFTGLXCTRL	OCMRAM memory DFT GLX ctrl	Refer to Table 6-63	0x0

Bits	Field Name	Description	Type	Reset
7:0	RESERVED	Read returns reset value	R	0x0

Table 6-63. Type Value For CONTROL_MEM_DFTRW1 Register

Field Name	MPU Mode	
	NSP	SP
DFTREADWRITEENABLE	Read returns 0s	R/OCO
MEMORY10DFTREADCTRL	Read returns 0s	R/W
MEMORY10DFTWRITECTRL	Read returns 0s	R/W
MEMORY9DFTREADCTRL	Read returns 0s	R/W
MEMORY9DFTWRITECTRL	Read returns 0s	R/W
MEMORY7DFTREADCTRL	Read returns 0s	R/W
MEMORY7DFTWRITECTRL	Read returns 0s	R/W
MEMORY6DFTGLXCTRL	Read returns 0s	R/W

NOTE: DSI is not available on all devices. See Chapter 1, the *Device Family* section, to check availability of this module.

6.6.4.5 CONTROL_MSUSPENDMUX_0

Table 6-64. CONTROL_MSUSPENDMUX_0

Address Offset	0x0000 0020		
Physical address	0x4800 2290	**Instance**	GENERAL
Description	MSuspend Control register: control the use of MSuspend signals at module level		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								MCBSP2MSCTRL			MCBSP1MSCTRL			I2C2MSCTRL		I2C1MSCTRL				RESERVED											

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Read returns reset value.	R	0x00
23:21	MCBSP2MSCTRL	Control McBSP_2 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
20:18	MCBSP1MSCTRL	Control McBSP_1 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
17:15	I2C2MSCTRL	Control I2C_2 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		

Bits	Field Name	Description	Type	Reset
		0x6: Reserved		
		0x7: Reserved		
14:12	I2C1MSCTRL	Control I2C_1 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
11:0	RESERVED	Read returns reset value.	R	0x00

6.6.4.6 CONTROL_MSUSPENDMUX_1

Table 6-65. CONTROL_MSUSPENDMUX_1

Address Offset	0x0000 0024		
Physical address	0x4800 2294	**Instance**	GENERAL
Description	MSuspend Control register : control the use of MSuspend signals at module level		
Type	R/W		

31 30	29 28 27	26 25 24	23 22 21	20 19 18	17 16	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
RESERVED	GPTM7MSCTRL	GPTM6MSCTRL	GPTM5MSCTRL	GPTM4MSCTRL	GPTM3MSCTRL	GPTM2MSCTRL	GPTM1MSCTRL	RESERVED

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Read returns reset value.	R	0x0
29:27	GPTM7MSCTRL	Control General Purpose Timer 7 sensitivity to MCU 0x0: No sensitivity: no MSuspend signal reaches the module 0x1: Sensitivity to MCU MSuspend signals 0x2: Reserved 0x3: Reserved 0x4: Reserved 0x5: Reserved 0x6: Reserved 0x7: Reserved	R/W	0x0
26:24	GPTM6MSCTRL	Control General Purpose Timer 6 sensitivity to MCU 0x0: No sensitivity: no MSuspend signal reaches the module 0x1: Sensitivity to MCU MSuspend signals 0x2: Reserved 0x3: Reserved 0x4: Reserved 0x5: Reserved 0x6: Reserved 0x7: Reserved	R/W	0x0
23:21	GPTM5MSCTRL	Control General Purpose Timer 5 sensitivity to MCU 0x0: No sensitivity: no MSuspend signal reaches the module 0x1: Sensitivity to MCU MSuspend signals 0x2: Reserved 0x3: Reserved 0x4: Reserved 0x5: Reserved	R/W	0x0

Bits	Field Name	Description	Type	Reset
		0x6: Reserved		
		0x7: Reserved		
20:18	GPTM4MSCTRL	Control General Purpose Timer 4 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
17:15	GPTM3MSCTRL	Control General Purpose Timer 3 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
14:12	GPTM2MSCTRL	Control General Purpose Timer 2 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
11:9	GPTM1MSCTRL	Control General Purpose Timer 1 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
8:0	RESERVED	Read returns reset value.	R	0x00

6.6.4.7 CONTROL_MSUSPENDMUX_2

Table 6-66. CONTROL_MSUSPENDMUX_2

Address Offset	0x0000 0028		
Physical Address	0x4800 2298	**Instance**	GENERAL
Description	MSuspend Control register : control the use of MSuspend signals at module level		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SYNCTMMSCTRL			RESERVED					WD3MSCTRL				WD2MSCTRL				WD1MSCTRL				GPTM12MSCTRL			
GPTM11MSCTRL				GPTM10MSCTRL				GPTM9MSCTRL				GPTM8MSCTRL																			

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Read returns reset value.	R	0x0
29:27	SYNCTMMSCTRL	Control Sync Timer32K sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
26:24	RESERVED	Read returns reset value.	R	0x0
23:21	WD3MSCTRL	Control Watch Dog 4 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
20:18	WD2MSCTRL	Control Watch Dog 2 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		

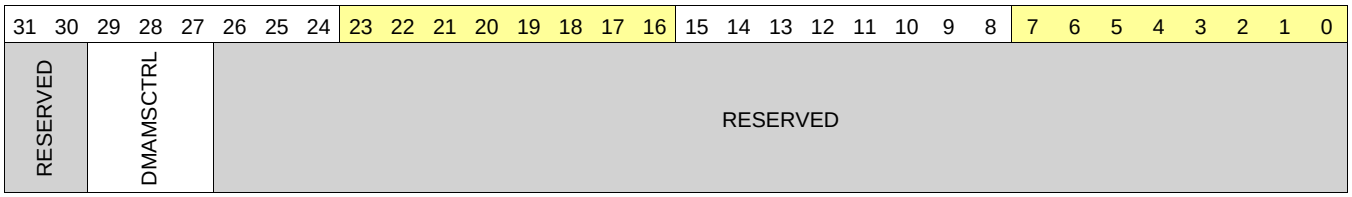
Bits	Field Name	Description	Type	Reset
		0x7: Reserved		
17:15	WD1MSCTRL	Control Watch Dog 1 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
14:12	GPTM12MSCTRL	Control General Purpose Timer 12 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
11:9	GPTM11MSCTRL	Control General Purpose Timer 11 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
8:6	GPTM10MSCTRL	Control General Purpose Timer 10 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
5:3	GPTM9MSCTRL	Control General Purpose Timer 9 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
2:0	GPTM8MSCTRL	Control General Purpose Timer 8 sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		

Bits	Field Name	Description	Type	Reset
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		

6.6.4.8 CONTROL_MSUSPENDMUX_4

Table 6-67. CONTROL_MSUSPENDMUX_4

Address Offset	0x0000 0030		
Physical Address	0x4800 22A0	**Instance**	GENERAL
Description	MSuspend Control register: control the use of MSuspend signals at module level		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Read returns reset value.	R	0x0
29:27	DMAMSCTRL	Control DMA sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
26:0	RESERVED	Read returns reset value.	R	0x0000000

6.6.4.9 CONTROL_MSUSPENDMUX_5

Table 6-68. CONTROL_MSUSPENDMUX_5

Address Offset	0x0000 0034		
Physical Address	0x4800 22A4	**Instance**	GENERAL
Description	MSuspend Control register: control the use of MSuspend signals at module level Not used		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								I2C3MSCTRL			RESERVED													MCBSP5MSCTRL			MCBSP4MSCTRL			MCBSP3MSCTRL		

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Read returns reset value.	R	0x0
23:21	I2C3MSCTRL	Control I2C-3 Sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
20:9	RESERVED	Read returns reset value.	R	0x0
8:6	MCBSP5MSCTRL	Control McBSP-5 Sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
5:3	MCBSP4MSCTRL	Control McBSP-4 Sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		

Bits	Field Name	Description	Type	Reset
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
2:0	MCBSP3MSCTRL	Control McBSP-3 Sensitivity to MCU	R/W	0x0
		0x0: No sensitivity: no MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		

6.6.4.10 CONTROL_MSUSPENDMUX_6

Table 6-69. CONTROL_MSUSPENDMUX_6

Address Offset	0x0000 0038		
Physical Address	0x4800 22A8	**Instance**	GENERAL
Description	MSuspend Control register: control the use of MSuspend signals at module level Not used		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								USB20OTGMSCTRL		CPGMACMSCTRL			HECCMSCTRL		

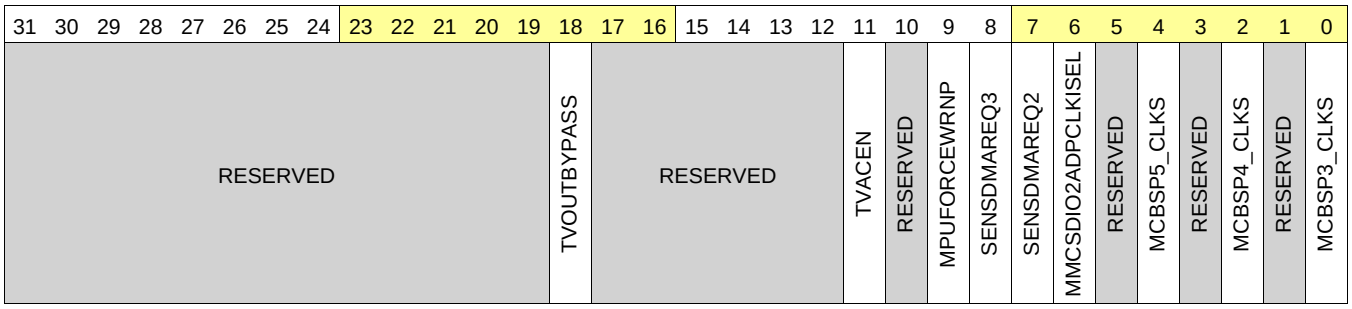
Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Read returns reset value.	R	0x0
8:6	USB20OTGMSCTRL	Control sensitivity USB20OTGMSCTRL to MCU MSuspend signals	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
5:3	CPGMACMSCTRL	Control sensitivity CPGMACMSCTRL to MCU MSuspend signals	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals 0thers: No sensitivity: No MSuspend signal reaches the module		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		
		0x7: Reserved		
2:0	HECCMSCTRL	Control sensitivity HECCMSCTRL to MCU MSuspend signals	R/W	0x0
		0x0: No sensitivity: No MSuspend signal reaches the module		
		0x1: Sensitivity to MCU MSuspend signals		
		0x2: Reserved		
		0x3: Reserved		
		0x4: Reserved		
		0x5: Reserved		
		0x6: Reserved		

Bits	Field Name	Description	Type	Reset
		0x7: Reserved		

6.6.4.11 CONTROL_DEVCONF1

Table 6-70. CONTROL_DEVCONF1

Address Offset	0x0000 0068		
Physical Address	0x4800 22D8	**Instance**	GENERAL
Description	Static device configuration register-1. Module dedicated functions		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:19	RESERVED	Read returns reset value	R	0x0
18	TVOUTBYPASS	Active high enable Dual 10-bit video DAC TV out bypass	R/W	0x0
17:12	RESERVED	Read returns reset value.	R	0x0
11	TVACEN	TV AC coupled load enable for TV output	R/W	0x0
10	RESERVED	Read returns reset value.	R	0x0
9	MPUFORCEWRNP	Force MPU writes to others to be non posted 0x0: Posted writes 0x1: Non posted writes	R/W	0x0
8	SENSDMAREQ3	Set sensitivity on SYS_NDMAREQ3 input pin 0x0: Level sensitivity 0x1: Edge sensitivity	R/W	0x0
7	SENSDMAREQ2	Set sensitivity on SYS_NDMAREQ2 input pin 0x0: Level sensitivity 0x1: Edge sensitivity	R/W	0x0
6	MMCSDIO2ADPCLKISEL	MMC/SDIO2 Module Input Clock selection 0x0: Input clock is from the external pin 0x1: Internal loop-back; module input clock is copied from the module output clock	R/W	0x0
5	RESERVED	Read returns reset value.	R	0x0
4	MCBSP5_CLKS	Select the CLKS input for the module McBSP5 0x0: CLKS is from the PRCM functional clock 0x1: CLKS is from the external pin McBSP_CLKS	R/W	0x0
3	RESERVED	Read returns reset value.	R	0x0
2	MCBSP4_CLKS	Select the CLKS input for the module McBSP4 0x0: CLKS is from the PRCM functional clock 0x1: CLKS is from the external pin McBSP_CLKS	R/W	0x0

Bits	Field Name	Description	Type	Reset
1	RESERVED	Read returns reset value.	R	0x0
0	MCBSP3_CLKS	Select the CLKS input for the module McBSP3 0x0: CLKS is from the PRCM functional clock 0x1: CLKS is from the external pin McBSP_CLKS	R/W	0x0

6.6.4.12 CONTROL_SEC_STATUS

Table 6-71. CONTROL_SEC_STATUS

Address Offset	0x0000 0070		
Physical Address	0x4800 22E0	**Instance**	GENERAL
Description	Security status register		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	MPUL2ISNOTACCESSIBLE	RESERVED	MPUL1ISNOTACCESSIBLE	RESERVED		COREBANK2ISNOTACCESSIBLE	COREBANK1ISNOTACCESSIBLE	RESERVED	MPUL2ISDESTROYED	RESERVED	MPUL1ISDESTROYED	RESERVED		COREBANK2ISDESTROYED	COREBANK1ISDESTROYED	USBHOSTWKUPRST	NEONWKUPRST	RESERVED		SGXWKUPRST	DISPWKUPRST	RESERVED	PERWKUPRST	EMUWKUPRST	COREWKUPRST	MPUWKUPRST	RAMBISTSTARTED	RESERVED		GLOBALWARMRESET	POWERONRESET

Bits	Field Name	Description	Type	Reset
31	RESERVED	Read returns reset value.	R	0
30	MPUL2ISNOTACCESSIBLE	L2 I,D$ accessible status 0x0: L2 I,D$ is accessible 0x1: L2 I,D$ is not accessible	Refer to Table 6-72	0x0
29	RESERVED	Read returns reset value.	R	0x0
28	MPUL1ISNOTACCESSIBLE	L1 I$ accessible status 0x0: L1 I$ is accessible 0x1: L1 I$ is not accessible	Refer to Table 6-72	0x0
27:26	RESERVED	Read returns reset value.	R	0x0
25	COREBANK2ISNOTACCESSIBLE	RAM Bank2 accessible status 0x0: RAM Bank2 is accessible 0x1: RAM Bank2 is not accessible	Refer to Table 6-72	0x0
24	COREBANK1ISNOTACCESSIBLE	RAM Bank1 accessible status 0x0: RAM Bank1 is accessible 0x1: RAM Bank1 is not accessible	Refer to Table 6-72	0x0
23	RESERVED	Read returns reset value.	R	0x0
22	MPUL2ISDESTROYED	L2 I,D$ damage status 0x0: L2 I,D$ is safe 0x1: L2 I,D$ is destroyed	Refer to Table 6-72	0x0
21	RESERVED	Read returns reset value.	R	0x0
20	MPUL1ISDESTROYED	L1 I$ damage status 0x0: L1 I$ is safe 0x1: L1 I$ is destroyed	Refer to Table 6-72	0x0

Bits	Field Name	Description	Type	Reset
19:18	RESERVED	Read returns reset value.	R	0x0
17	COREBANK2ISDESTROYED	RAM Bank2 damage status 0x0: RAM Bank2 is safe 0x1: RAM Bank2 is destroyed	Refer to Table 6-72	0x0
16	COREBANK1ISDESTROYED	RAM Bank1 damage status 0x0: RAM Bank1 is safe 0x1: RAM Bank1 is destroyed	Refer to Table 6-72	0x0
15	USBHOSTWKUPRST	USB Host Domain Reset Status 0x0: No USB Host domain reset (any source of reset) 0x0: USB Host domain has been reset (any source of reset)	Refer to Table 6-72	0x0
14	NEONWKUPRST	Neon Domain Reset Status 0x0: No Neon domain reset 0x1: Neon has been reset	Refer to Table 6-72	0x0
13:12	RESERVED	Read returns reset value.	R	0x0
11	SGXWKUPRST	SGX Domain Reset Status 0x0: No SGX domain reset 0x1: SGX domain has been reset	Refer to Table 6-72	0x0
10	DISPWKUPRST	Display Domain Reset Status 0x0: No Display domain reset 0x1: Display domain has been reset	Refer to Table 6-72	0x0
9	RESERVED	Read returns reset value.	R	0x0
8	PERWKUPRST	Peripheral Domain Reset Status 0x0: No Peripheral domain reset 0x1: Peripheral domain has been reset	Refer to Table 6-72	0x0
7	EMUWKUPRST	Emulation Domain Reset Status 0x0: No Emulation domain reset 0x1: Emulation domain has been reset	Refer to Table 6-72	0x0
6	COREWKUPRST	Core Domain Reset Status 0x0: No Core domain reset 0x1: Core domain has been reset	Refer to Table 6-72	0x0
5	MPUWKUPRST	MPU domain Reset Status 0x0: No MPU domain reset 0x1: MPU domain has been reset	Refer to Table 6-72	0x0
4	RAMBISTSTARTED	RAM BIST Started status 0x0: Memory BIST on on-chip RAM not started 0x1: Memory BIST on on-chip RAM started	Refer to Table 6-72	0x0
3:2	RESERVED	Read returns reset value.	R	0x0
1	GLOBALWARMRESET	Global Warm Reset status 0x0: previous reset was not a Global Software Warm reset 0x1: previous reset was a Global Software warm reset	Refer to Table 6-72	0x0
0	POWERONRESET	Power On Reset status 0x0: Previous reset was not a PowerOn Reset 0x1: Previous reset was a PowerOn Reset	Refer to Table 6-72	0x1

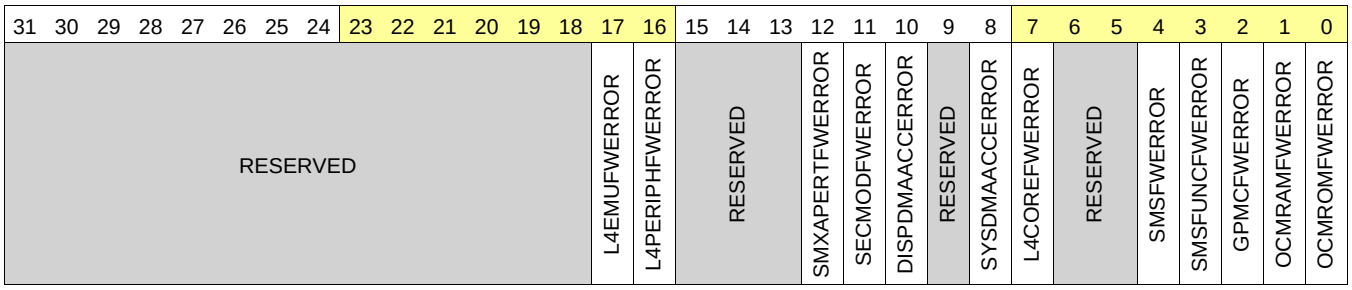
Table 6-72. Type Value For CONTROL_SEC_STATUS Register

MPU Mode	NSP	SP
MPUL2ISNOTACCESSIBLE	R	R/W1toClr
MPUL1ISNOTACCESSIBLE	R	R/W1toClr
COREBANK2ISNOTACCESSIBLE	R	R/W1toClr
COREBANK1ISNOTACCESSIBLE	R	R/W1toClr
MPUL2ISDESTROYED	R	R/W1toClr
MPUL1ISDESTROYED	R	R/W1toClr
COREBANK2ISDESTROYED	R	R/W1toClr
COREBANK1ISDESTROYED	R	R/W1toClr
USBHOSTWKUPRST	R	R/W1toClr
NEONWKUPRST	R	R/W1toClr
SGXWKUPRST	R	R/W1toClr
DISPWKUPRST	R	R/W1toClr
PERWKUPRST	R	R/W1toClr
EMUWKUPRST	R	R/W1toClr
COREWKUPRST	R	R/W1toClr
MPUWKUPRST	R	R/W1toClr
RAMBISTSTARTED	R	R/W1toClr
GLOBALWARMRESET	R	R/W1toClr
POWERONRESET	R	R/W1toClr

6.6.4.13 CONTROL_SEC_ERR_STATUS

Table 6-73. CONTROL_SEC_ERR_STATUS

Address Offset	0x0000 0074		
Physical Address	0x4800 22E4	**Instance**	GENERAL
Description	Security error status register		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Read returns reset value.	R	0x0
17	L4EMUFWERROR	L4 Emulation Firewall Error 0x0: No L4 Emulation interconnect firewall error 0x1: L4 Emulation interconnect firewall error	Refer to Table 6-74	0x0
16	L4PERIPHFWERROR	L4 Peripheral Firewall Error 0x0: No L4 Peripheral interconnect firewall error 0x1: L4 Peripheral interconnect firewall error	Refer to Table 6-74	0x0
15:13	RESERVED	Read returns reset value.	R	0x0
12	SMXAPERTFWERROR	L3 Register target Firewall error 0x0: No L3 Register Target firewall error 0x1: L3 Register Target firewall error	Refer to Table 6-74	0x0
11	SECMODFWERROR	Secure State Machine Firewall Error 0x0: No Secure State Machine firewall error 0x1 Secure State Machine firewall error	Refer to Table 6-74	0x0
10	DISPDMAACCERROR	Disp Dma Access Error 0x0: No access error to DISP DMA secure channels 0x1: Unauthorized access to a DISP DMA secure channel	Refer to Table 6-74	0x0
9	RESERVED	Read returns reset value.	R	0x0
8	SYSDMAACCERROR	sDma Access Error 0x0: No access error to sDMA secure channels 0x1: Unauthorized access to a sDMA secure channel	Refer to Table 6-74	0x0
7	L4COREFWERROR	L4 Security Firewall Error 0x0: No error from L4 Core security firewall 0x1: Error from L4 Core security firewall	Refer to Table 6-74	0x0
6:5	RESERVED	Read returns reset value.	R	0x0

Bits	Field Name	Description	Type	Reset
4	SMSFWERROR	SMS Firewall Error 0x0: No SMS firewall error 0x1: SMS firewall error	Refer to Table 6-74	0x0
3	SMSFUNCFWERROR	SMS Functional Firewall Error 0x0: No SMS functional firewall error 0x1: SMS functional firewall error	Refer to Table 6-74	0x0
2	GPMCFWERROR	GPMC Firewall Error 0x0: No error from GPMC security firewall 0x1: Error from GPMC security firewall	Refer to Table 6-74	0x0
1	OCMRAMFWERROR	On Chip Ram Firewall Error 0x0: No error from On Chip RAM security firewall 0x1: Error from On Chip RAM security firewall	Refer to Table 6-74	0x0
0	OCMROMFWERROR	On Chip Rom Firewall Error 0x0: No error from On Chip ROM security firewall 0x1: Error from On Chip ROM security firewall	Refer to Table 6-74	0x0

Table 6-74. Type Value For CONTROL_SEC_ERR_STATUS Register

MPU Mode	NSP	SP
L4EMUFWERROR	R	R/W1toClr
L4PERIPHFWERROR	R	R/W1toClr
SMXAPERTFWERROR	R	R/W1toClr
SECMODEFWERROR	R	R/W1toClr
DISPDMAACCERROR	R	R/W1toClr
SYSDMAACCERROR	R	R/W1toClr
L4COREFWERROR	R	R/W1toClr
SMSFWERROR	R	R/W1toClr
SMSFUNCFWERROR	R	R/W1toClr
GPMCFWERROR	R	R/W1toClr
OCMRAMFWERROR	R	R/W1toClr
OCMROMFWERROR	R	R/W1toClr

6.6.4.14 CONTROL_SEC_ERR_STATUS_DEBUG

Table 6-75. CONTROL_SEC_ERR_STATUS_DEBUG

Address Offset	0x0000 0078		
Physical Address	0x4800 22E8	**Instance**	GENERAL
Description	Security Error Status Debug Register		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														L4EMUDBGFWERROR	L4PERIPHERALDBGFWERROR	RESERVED			SMXAPERTDBGFWERROR	RESERVED				L4COREDBGFWERROR	RESERVED			SMSDBGFWERROR	GPMCDBGFWERROR	OCMRAMDBGFWERROR	OCMROMDBGFWERROR

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Read returns reset value	R	0x0
17	L4EMUDBGFWERROR	L4 Emulation Debug Firewall Error 0x0: No firewall error in L4 Emulation Debug 0x1: Firewall error in L4 Emulation Debug	Refer to Table 6-76	0x0
16	L4PERIPHERALDBGFWERROR	L4 Peripheral Debug Firewall Error 0x0: No firewall error in L4 Peripheral Debug 0x1: Firewall error in L4 Peripheral Debug	Refer to Table 6-76	0x0
15:13	RESERVED	Read returns reset value	R	0x0
12	SMXAPERTDBGFWERROR	L3 Register target Debug Firewall error 0x0: No firewall error in L3 Register Target Debug 0x1: Firewall error in L3 Register Target Debug	Refer to Table 6-76	0x0
11:8	RESERVED	Read returns reset value.	R	0x0
7	L4COREDBGFWERROR	L4 Core Debug Firewall Error 0x0: No firewall error in L4 Core Debug 0x1: Firewall error in L4 Core Debug	Refer to Table 6-76	0x0
6:4	RESERVED	Read returns reset value.	R	0x0
3	SMSDBGFWERROR	SMS Debug Firewall Error 0x0: No Firewall debug error in SMS 0x1: Firewall debug error in SMS	Refer to Table 6-76	0x0
2	GPMCDBGFWERROR	GPMC Debug Firewall Error 0x0: No error from GPMC debug security firewall 0x1: Error from GPMC debug security firewall	Refer to Table 6-76	0x0

Bits	Field Name	Description	Type	Reset
1	OCMRAMDBGFWERROR	On Chip Ram Debug Firewall Error 0x0: No error from On Chip RAM debug security firewall 0x1: Error from On Chip RAM debug security firewall	Refer to Table 6-76	0x0
0	OCMROMDBGFWERROR	On Chip Rom Debug Firewall Error 0x0: No error from On Chip ROM debug security firewall 0x1: Error from On Chip ROM debug security firewall	Refer to Table 6-76	0x0

Table 6-76. Type Value For CONTROL_SEC_ERR_STATUS_DEBUG Register

MPU Mode	NSP	SP
L4EMUDBGFWERROR	R	R/W1toClr
L4PERIPHDBGFWERROR	R	R/W1toClr
SMXAPERTDBGFWERROR	R	R/W1toClr
L4COREDBGFWERROR	R	R/W1toClr
SMSDBGFWERROR	R	R/W1toClr
GPMCDBGFWERROR	R	R/W1toClr
OCMRAMDBGFWERROR	R	R/W1toClr
OCMROMDBGFWERROR	R	R/W1toClr

6.6.4.15 CONTROL_STATUS

Table 6-77. CONTROL_STATUS

Address Offset	0x0000 0080		
Physical Address	0x4800 22F0	**Instance**	GENERAL
Description	Control Module Status register: latches system information at reset time		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					DEVICETYPE			RESERVED						SYS_BOOT						

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Read returns reset value.	R	0x-
10:8	DEVICETYPE	Device type captured at reset time 0x1: Reserved 0x2: Reserved 0x3: GP device Other values: Reserved	R	0x-
7:6	RESERVED	Read returns reset value.	R	0x-
5:0	SYS_BOOT	sys_boot[5:0] pin values sampled at power-on reset	R	0x-

6.6.4.16 CONTROL_RPUB_KEY_H_0

Table 6-78. CONTROL_RPUB_KEY_H_0

Address Offset	0x0000 0090		
Physical Address	0x4800 2300	**Instance**	SYSC_GENERAL1
Description	Root_public_key_hash; B-field		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPKH_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	RPKH_31_0	Root Public Key Hash bits [31:0] Fuse Keys [31:0]	R	0x00000000

6.6.4.17 CONTROL_RPUB_KEY_H_1

Table 6-79. CONTROL_RPUB_KEY_H_1

Address Offset	0x0000 0094		
Physical Address	0x4800 2304	**Instance**	SYSC_GENERAL1
Description	Root_public_key_hash; B-field		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPKH_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	RPKH_63_32	Root Public Key Hash bits [63:32] Fuse Keys [63:32]	R	0x00000000

6.6.4.18 CONTROL_RPUB_KEY_H_2

Table 6-80. CONTROL_RPUB_KEY_H_2

Address Offset	0x0000 0098		
Physical Address	0x4800 2308	**Instance**	SYSC_GENERAL1
Description	Root_public_key_hash; B-field		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPKH_95_64																															

Bits	Field Name	Description	Type	Reset
31:0	RPKH_95_64	Root Public Key Hash bits [95:64] Fuse Keys [95:64]	R	0x00000000

6.6.4.19 CONTROL_RPUB_KEY_H_3

Table 6-81. CONTROL_RPUB_KEY_H_3

Address Offset	0x0000 009C		
Physical Address	0x4800 230C	**Instance**	SYSC_GENERAL1
Description	Root_public_key_hash; B-field		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPKH_127_96																															

Bits	Field Name	Description	Type	Reset
31:0	RPKH_127_96	Root Public Key Hash bits [127:96] Fuse Keys [127:96]	R	0x00000000

6.6.4.20 CONTROL_RPUB_KEY_H_4

Table 6-82. CONTROL_RPUB_KEY_H_4

Address Offset	0x0000 00A0		
Physical Address	0x4800 2310	**Instance**	SYSC_GENERAL1
Description	Root_public_key_hash; B-field		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPKH_159_128																															

Bits	Field Name	Description	Type	Reset
31:0	RPKH_159_128	Root Public Key Hash bits [159:128] Fuse Keys [159:128]	R	0x00000000

6.6.4.21 CONTROL_USB_CONF_0

Table 6-83. CONTROL_USB_CONF_0

Address Offset	0x0000 0100		
Physical Address	0x4800 2370	**Instance**	SYSC_GENERAL1
Description	USB Fuse conf [31:0],USB Product ID [31:16] Vendor ID [15:0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB_CONF_31_0																															

Bits	Field Name	Description	Type	Reset
31:0	USB_CONF_31_0	USB Fuse conf [31:0],USB Product ID [31:16] Vendor ID [15:0]	R	0x00000000

6.6.4.22 CONTROL_USB_CONF_1

Table 6-84. CONTROL_USB_CONF_1

Address Offset	0x0000 0104		
Physical Address	0x4800 2374	**Instance**	SYSC_GENERAL1
Description	USB Fuse conf [63:32], SEQ_DISADAPTCLK[1], USB PHY detection mode [0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBCONF_63_32																															

Bits	Field Name	Description	Type	Reset
31:0	USBCONF_63_32	USB Fuse conf [63:32], SEQ_DISADAPTCLK[1], USB PHY detection mode [0]	R	0x00000000

6.6.4.23 CONTROL_FUSE_EMAC_LSB

Table 6-85. CONTROL_FUSE_EMAC_LSB

Address Offset	0x0000 0110		
Physical Address	0x4800 2380	**Instance**	SYSC_GENERAL1
Description	Fuse OPP [95:72]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FUSE_OPP_95_72																							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Read returns reset value.	R	0x00000000
23:0	FUSE_OPP_95_72	Fuse OPP [95:72]	R	0x00000000

6.6.4.24 CONTROL_FUSE_EMAC_MSB

Table 6-86. CONTROL_FUSE_EMAC_MSB

Address Offset	0x0000 0114		
Physical Address	0x4800 2384	**Instance**	SYSC_GENERAL1
Description	Fuse OPP [119:96]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FUSE_OPP_119_96																							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Read returns reset value.	R	0x00000000
24:0	FUSE_OPP_119_96	Fuse OPP [119:96]	R	0x00000000

6.6.4.25 CONTROL_FUSE_SR

Table 6-87. CONTROL_FUSE_SR

Address Offset	0x0000 0130		
Physical Address	0x4800 23A0	**Instance**	SYSC_GENERAL1
Description	Fuse SR1 and SR2		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																FUSE_SR2								FUSE_SR1							

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns reset value.	R	0x00000000
15:8	FUSE_SR2	Fuse SR 2	R	0x00000000
7:0	FUSE_SR1	Fuse SR 1	R	0x00000000

6.6.4.26 CONTROL_CEK_0

Table 6-88. CONTROL_CEK_0

Address Offset	0x0000 0134		
Physical Address	0x4800 23A4	**Instance**	SYSC_GENERAL1
Description	Customer Key [31:0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_0																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_0	Cpefuse CEK [31:0]	Refer to Table 6-89	0x00000000

Table 6-89. Type Value For CONTROL_CEK_0 Register

MPU Mode	SP	NSP	-
Device Type	**S/B/E/T**	**E/T/S/B**	**G**
CEK_0	R/W1toClr	R	R

6.6.4.27 CONTROL_CEK_1

Table 6-90. CONTROL_CEK_1

Address Offset	0x0000 0138		
Physical Address	0x4800 23A8	**Instance**	SYSC_GENERAL1
Description	Customer Key [63:32]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_1																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_1	Cpefuse CEK [63:32]	Refer to Table 6-91	0x00000000

Table 6-91. Type Value For CONTROL_CEK_1 Register

MPU Mode	SP	NSP	-
Device Type	**S/B/E/T**	**E/T/S/B**	**G**
CEK_1	R/W1toClr	R	R

6.6.4.28 CONTROL_CEK_2

Table 6-92. CONTROL_CEK_2

Address Offset	0x0000 013C		
Physical Address	0x4800 23AC	**Instance**	SYSC_GENERAL1
Description	Customer Key [95:64]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CEK_2																							

Bits	Field Name	Description	Type	Reset
31:0	CEK_2	Cpefuse CEK [95:64]	Refer to Table 6-93	0x00000000

Table 6-93. Type Value For CONTROL_CEK_2 Register

MPU Mode	SP	NSP	-
Device Type	**S/B/E/T**	**E/T/S/B**	**G**
CEK_2	R/W1toClr	R	R

6.6.4.29 CONTROL_CEK_3

Table 6-94. CONTROL_CEK_3

Address Offset	0x0000 0140		
Physical Address	0x4800 23B0	**Instance**	SYSC_GENERAL1
Description	Customer Key [127:96]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_3																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_3	Cpefuse CEK [127:96]	Refer to Table 6-95	0x00000000

Table 6-95. Type Value For CONTROL_CEK_3 Register

MPU Mode	SP	NSP	-
Device Type	**S/B/E/T**	**E/T/S/B**	**G**
CEK_3	R/W1toClr	R	R

6.6.4.30 CONTROL_MSV_0

Table 6-96. CONTROL_MSV_0

Address Offset	0x0000 0144		
Physical Address	0x4800 23B4	**Instance**	SYSC_GENERAL1
Description	Model specific value [31:0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSV_0																															

Bits	Field Name	Description	Type	Reset
31:0	MSV_0	Model specific value [31:0]	Refer to Table 6-97	0x00000000

Table 6-97. Type Value For CONTROL_MSV_0 Register

MPU Mode	SP	NSP	-
Device Type	**S/B/E/T**	**E/T/S/B**	**G**
MSV_0	R/W1toClr	R	R

6.6.4.31 CONTROL_CEK_BCH_0

Table 6-98. CONTROL_CEK_BCH_0

Address Offset	0x0000 0148		
Physical Address	0x4800 23B8	**Instance**	SYSC_GENERAL1
Description	Cpefuse CEK (BCH Decoded) [31:0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_BCH_0																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_BCH_0	Cpefuse CEK (BCH Decoded) [31:0]	R	0x00000000

6.6.4.32 CONTROL_CEK_BCH_1

Table 6-99. CONTROL_CEK_BCH_1

Address Offset	0x0000 014C		
Physical Address	0x4800 23BC	**Instance**	SYSC_GENERAL1
Description	Cpefuse CEK (BCH Decoded) [63:32]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_BCH_1																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_BCH_1	Cpefuse CEK (BCH Decoded) [63:32]	R	0x00000000

6.6.4.33 CONTROL_CEK_BCH_2

Table 6-100. CONTROL_CEK_BCH_2

Address Offset	0x0000 0150		
Physical Address	0x4800 23C0	**Instance**	SYSC_GENERAL1
Description	Cpefuse CEK (BCH Decoded) [95:64]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_BCH_2																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_BCH_2	Cpefuse CEK (BCH Decoded) [95:64]	R	0x00000000

6.6.4.34 CONTROL_CEK_BCH_3

Table 6-101. CONTROL_CEK_BCH_3

Address Offset	0x0000 0154		
Physical Address	0x4800 23C4	**Instance**	SYSC_GENERAL1
Description	Cpefuse CEK (BCH Decoded) [127:96]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_BCH_3																															

Bits	Field Name	Description	Type	Reset
31:24	CEK_BCH_3	Cpefuse CEK (BCH Decoded) [127:96]	R	0x00000000

6.6.4.35 CONTROL_CEK_BCH_4

Table 6-102. CONTROL_CEK_BCH_4

Address Offset	0x0000 0158		
Physical Address	0x4800 23C8	**Instance**	SYSC_GENERAL1
Description	Cpefuse CEK (BCH Decoded) [143:128]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_BCH_4																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_BCH_4	Cpefuse CEK (BCH Decoded) [159:128]	R	0x00000000

6.6.4.36 CONTROL_MSV_BCH_0

Table 6-103. CONTROL_MSV_BCH_0

Address Offset	0x0000 015C		
Physical Address	0x4800 23CC	**Instance**	SYSC_GENERAL1
Description	Cpefuse MSV (BCH Decoded) [31:0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_MSV_BCH_0																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_MSV_BCH_0	Cpefuse MSV (BCH Decoded) [31:0]	R	0x00000000

6.6.4.37 CONTROL_MSV_BCH_1

Table 6-104. CONTROL_MSV_BCH_1

Address Offset	0x0000 0160		
Physical Address	0x4800 23D0	**Instance**	SYSC_GENERAL1
Description	Cpefuse MSV (BCH Decoded) [63:32]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_MSV_BCH_1																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_MSV_BCH_1	Cpefuse MSV (BCH Decoded) [63:32]	R	0x00000000

6.6.4.38 CONTROL_SWRV_0

Table 6-105. CONTROL_SWRV_0

Address Offset	0x0000 0164		
Physical Address	0x4800 23D4	**Instance**	SYSC_GENERAL1
Description	Software revision value [31:0]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_SWRV_0																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_SWRV_0	Software revision value [31:0]	R	0x00000000

6.6.4.39 CONTROL_SWRV_1

Table 6-106. CONTROL_SWRV_1

Address Offset	0x0000 0168		
Physical Address	0x4800 23D8	**Instance**	SYSC_GENERAL1
Description	Software revision value [63:32]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_SWRV_1																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_SWRV_1	Software revision value [63:32]	R	0x00000000

6.6.4.40 CONTROL_SWRV_2

Table 6-107. CONTROL_SWRV_2

Address Offset	0x0000 016C		
Physical Address	0x4800 23DC	**Instance**	SYSC_GENERAL1
Description	Software revision value [95:64]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_SWRV_2																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_SWRV_2	Software revision value [95:64]	R	0x00000000

6.6.4.41 CONTROL_SWRV_3

Table 6-108. CONTROL_SWRV_3

Address Offset	0x0000 0170		
Physical Address	0x4800 23E0	**Instance**	SYSC_GENERAL1
Description	Software revision value [127:96]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_SWRV_3																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_SWRV_3	Software revision value [127:96]	R	0x00000000

6.6.4.42 CONTROL_SWRV_4

Table 6-109. CONTROL_SWRV_4

Address Offset	0x0000 0174		
Physical Address	0x4800 23E4	**Instance**	SYSC_GENERAL1
Description	Software revision value [:]		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEK_SWRV_4																															

Bits	Field Name	Description	Type	Reset
31:0	CEK_SWRV_4	Software revision value [159:128]	R	0x00000000

6.6.4.43 CONTROL_DEBOBS_0

Table 6-110. CONTROL_DEBOBS_0

Address Offset	0x0000 01B0		
Physical Address	0x4800 2420	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg0, hw_dbg1		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX0								RESERVED								OBSMUX1							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX0	Select the set of signals to be exported for WKUPOBSMUX0	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX1	Select the set of signals to be exported for WKUPOBSMUX1	R/W	0x000

6.6.4.44 CONTROL_DEBOBS_1

Table 6-111. CONTROL_DEBOBS_1

Address Offset	0x0000 01B4		
Physical Address	0x4800 2424	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg2, hw_dbg3		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX2								RESERVED								OBSMUX3							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX2	Select the set of signals to be exported for WKUPOBSMUX2	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX3	Select the set of signals to be exported for WKUPOBSMUX3	R/W	0x000

6.6.4.45 CONTROL_DEBOBS_2

Table 6-112. CONTROL_DEBOBS_2

Address Offset	0x0000 01B8		
Physical Address	0x4800 2428	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg4, hw_dbg5		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX4								RESERVED								OBSMUX5							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX4	Select the set of signals to be exported for WKUPOBSMUX4	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX5	Select the set of signals to be exported for WKUPOBSMUX5	R/W	0x000

6.6.4.46 CONTROL_DEBOBS_3

Table 6-113. CONTROL_DEBOBS_3

Address Offset	0x0000 01BC		
Physical Address	0x4800 242C	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg6, hw_dbg7		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX6								RESERVED								OBSMUX7							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX6	Select the set of signals to be exported for WKUPOBSMUX6	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX7	Select the set of signals to be exported for WKUPOBSMUX7	R/W	0x000

6.6.4.47 CONTROL_DEBOBS_4

Table 6-114. CONTROL_DEBOBS_4

Address Offset	0x0000 01C0		
Physical Address	0x4800 2430	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg8, hw_dbg9		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX8								RESERVED								OBSMUX9							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX8	Select the set of signals to be exported for WKUPOBSMUX8	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX9	Select the set of signals to be exported for WKUPOBSMUX9	R/W	0x000

6.6.4.48 CONTROL_DEBOBS_5

Table 6-115. CONTROL_DEBOBS_5

Address Offset	0x0000 01C4		
Physical Address	0x4800 2434	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg10, hw_dbg11		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX10								RESERVED								OBSMUX11							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX10	Select the set of signals to be exported for WKUPOBSMUX10	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX11	Select the set of signals to be exported for WKUPOBSMUX11	R/W	0x000

6.6.4.49 CONTROL_DEBOBS_6

Table 6-116. CONTROL_DEBOBS_6

Address Offset	0x0000 01C8		
Physical Address	0x4800 2438	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg12, hw_dbg13		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX12								RESERVED								OBSMUX13							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX12	Select the set of signals to be exported for WKUPOBSMUX12	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX13	Select the set of signals to be exported for WKUPOBSMUX13	R/W	0x000

6.6.4.50 CONTROL_DEBOBS_7

Table 6-117. CONTROL_DEBOBS_7

Address Offset	0x0000 01CC		
Physical Address	0x4800 243C	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg14, hw_dbg15		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX14								RESERVED								OBSMUX15							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX14	Select the set of signals to be exported for WKUPOBSMUX14	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX15	Select the set of signals to be exported for WKUPOBSMUX15	R/W	0x000

6.6.4.51 CONTROL_DEBOBS_8

Table 6-118. CONTROL_DEBOBS_8

Address Offset	0x0000 01D0		
Physical Address	0x4800 2440	**Instance**	GENERAL
Description	Select the set of signals to be observed for hw_dbg16, hw_dbg17		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								OBSMUX16								RESERVED								OBSMUX17							

Bits	Field Name	Description	Type	Reset
31:23	RESERVED	Read returns reset value.	R	0x00
22:16	OBSMUX16	Select the set of signals to be exported for WKUPOBSMUX16	R/W	0x000
15:7	RESERVED	Read returns reset value.	R	0x00
6:0	OBSMUX17	Select the set of signals to be exported for WKUPOBSMUX17	R/W	0x000

6.6.4.52 CONTROL_WKUP_CTRL

Table 6-119. CONTROL_WKUP_CTRL

Address Offset	0x0000 0A5C		
Physical Address	0x4800 2A5C	**Instance**	GENERAL
Description	USB TXEN polarity control and log modem warm reset source mux sel		
Type	R/W		

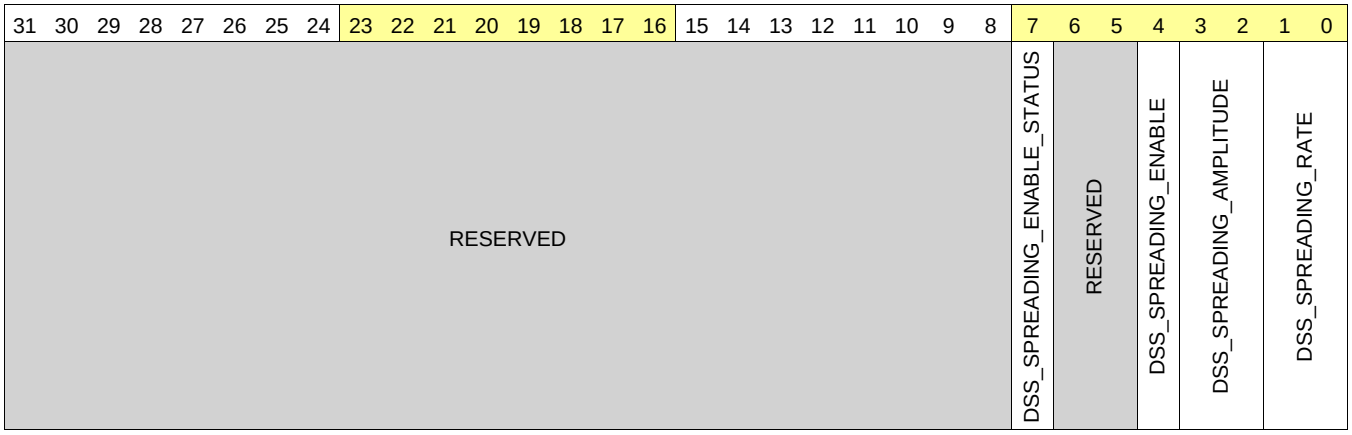
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													MM_FSUSB3_TXEN_N_	MM_FSUSB2_TXEN_N_	MM_FSUSB1_TXEN_N_

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Reserved for future use.	R	0
2	MM_FSUSB3_TXEN_N_ OUT_POLARITY_CTRL	Polarity control for TXEN signal of serial USB interface, port3. 0: Active low 1: Active high	R/W	0
1	MM_FSUSB2_TXEN_N_ OUT_POLARITY_CTRL	Polarity control for TXEN signal of serial USB interface, port2. 0: Active low 1: Active high	R/W	0
0	MM_FSUSB1_TXEN_N_ OUT_POLARITY_CTRL	Polarity control for TXEN signal of serial USB interface, port1. 0: Active low 1: Active high	R/W	0

6.6.4.53 CONTROL_DSS_DPLL_SPREADING

Table 6-120. CONTROL_DSS_DPLL_SPREADING

Address Offset	0x0000 01E0		
Physical Address	0x4800 2450	**Instance**	GENERAL
Description	This register controls the EMI Reduction feature for Display_SS/DSI DPLL		
Type	R/W		



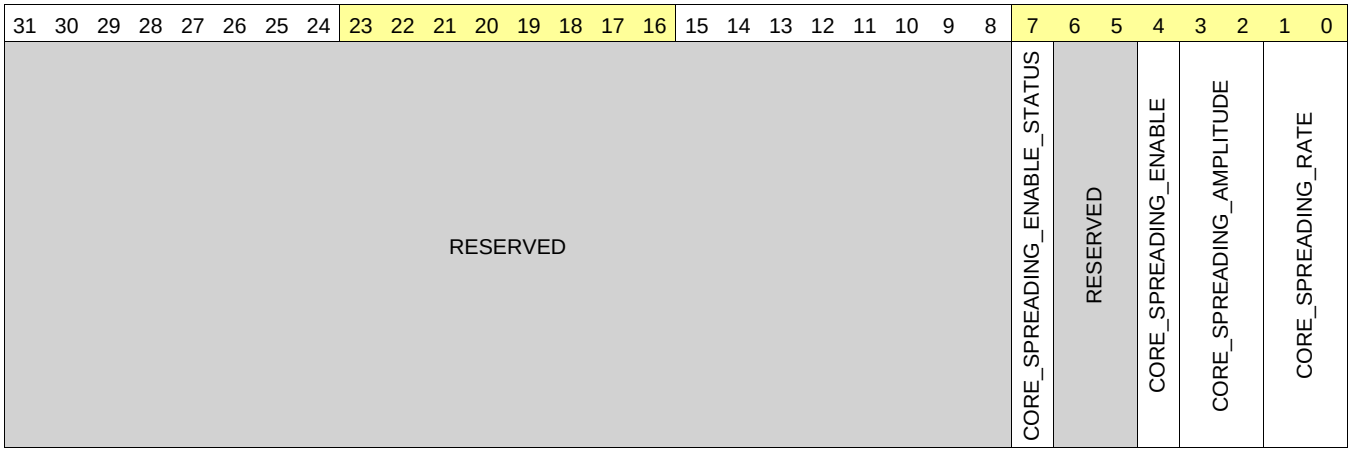
Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reads return zero	R	0x000000
7	DSS_SPREADING_ENABLE_STATUS	Indicates the status of the SSC feature	R	-
6:5	RESERVED	Reads return zero	R	0x0
4	DSS_SPREADING_ENABLE	Enables/disables EMI Reduction feature (Spreading): 0: Modulation stops at the end of the current modulation cycle. 1: Modulation cycle is started.	R/W	0x0
3:2	DSS_SPREADING_AMPLITUDE	Controls the modulation index. This index (K) is a ratio of Frequency spread (Δf) and spreading rate (f_m) . 00: K = 4 01: K = 6 10: K = 8 11: K = 10	R/W	0x0
1:0	DSS_SPREADING_RATE	Controls the rate of frequency modulation: 00: 62.5 to 125 KHz 01: 125 to 250 KHz 10: 250 to 500 KHz 11: 500 to 1000 KHz	R/W	0x0

NOTE: DSI is not available on all devices. See Chapter 1, the *Device Family* section, to check availability of this module.

6.6.4.54 CONTROL_CORE_DPLL_SPREADING

Table 6-121. CONTROL_CORE_DPLL_SPREADING

Address Offset	0x0000 0454		
Physical Address	0x4800 2454	**Instance**	GENERAL
Description	This register controls the EMI Reduction feature for CORE domain DPLL		
Type	R/W		

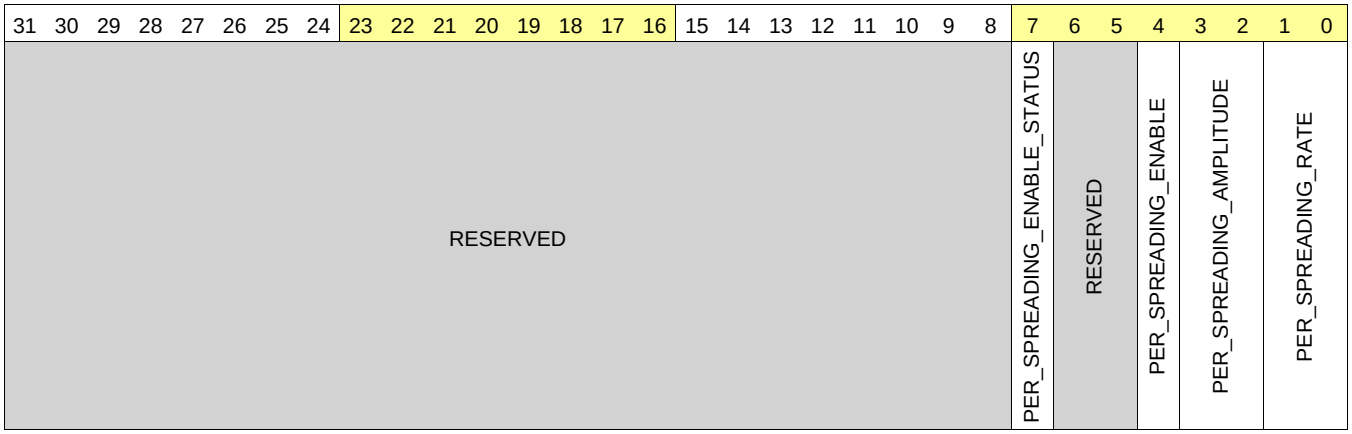


Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reads return zero	R	0x000000
7	CORE_SPREADING_ENABLE_STATUS	Indicates the status of the SSC feature	R	-
6:5	RESERVED	Reads return zero	R	0x0
4	CORE_SPREADING_ENABLE	Enables/disables EMI Reduction feature (Spreading): 0: Modulation stops at the end of the current modulation cycle. 1: Modulation cycle is started.	R/W	0x0
3:2	CORE_SPREADING_AMPLITUDE	Controls the modulation index. This index (K) is a ratio of Frequency spread (Δf) and spreading rate (f_m) . 00: K = 4 01: K = 6 10: K = 8 11: K = 10	R/W	0x0
1:0	CORE_SPREADING_RATE	Controls the rate of frequency modulation: 00: 62.5 to 125 KHz 01: 125 to 250 KHz 10: 250 to 500 KHz 11: 500 to 1000 KHz	R/W	0x0

6.6.4.55 CONTROL_PER_DPLL_SPREADING

Table 6-122. CONTROL_PER_DPLL_SPREADING

Address Offset	0x0000 0458		
Physical Address	0x4800 2458	**Instance**	GENERAL
Description	This register controls the EMI Reduction feature for PER domain DPLL		
Type	R		



Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reads return zero	R	0x000000
7	PER_SPREADING_ENABLE_STATUS	Indicates the status of the Spreading feature	R	-
6:5	RESERVED	Reads return zero	R	0x0
4	PER_SPREADING_ENABLE	Enables/disables EMI Reduction feature (Spreading): 0: Modulation stops at the end of the current modulation cycle. 1: Modulation cycle is started.	R/W	0x0
3:2	PER_SPREADING_AMPLITUDE	Controls the modulation index. This index (K) is a ratio of the spreading frequency (Δf) and spreading rate (f_m) . 00: K = 4 01: K = 6 10: K = 8 11: K = 10	R/W	0x0
1:0	PER_SPREADING_RATE	Controls the rate of frequency modulation: 00: Reserved 01: 125 to 250 KHz 10: 250 to 500 KHz 11: 500 to 1000 KHz	R/W	0x0

6.6.4.56 CONTROL_USBHOST_DPLL_SPREADING

Table 6-123. CONTROL_USBHOST_DPLL_SPREADING

Address Offset	0x0000 045C		
Physical Address	0x4800 245C	**Instance**	GENERAL
Description	This register controls the EMI Reduction feature for USBHOST DPLL		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								USBHOST_SPREADING_ENABLE_STATUS	RESERVED		USBHOST_SPREADING_ENABLE	USBHOST_SPREADING_AMPLITUDE		USBHOST_SPREADING_RATE	

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reads return zero	R	0x000000
7	USBHOST_SPREADING_ENABLE_STATUS	Indicates the status of the Spreading feature	R	-
6:5	RESERVED	Reads return zero	R	0x0
4	USBHOST_SPREADING_ENABLE	Enables/disables EMI Reduction feature (Spreading): 0: Modulation stops at the end of the current modulation cycle. 1: Modulation cycle is started.	R/W	0x0
3:2	USBHOST_SPREADING_AMPLITUDE	Controls the modulation index. This index (K) is a ratio of Frequency spread (Δf) and spreading rate (f_m) . 00: K = 4 01: K = 6 10: K = 8 11: K = 10	R/W	0x0
1:0	USBHOST_SPREADING_RATE	Controls the rate of frequency modulation: 00: 62.5 to 125 KHz 01: 125 to 250 KHz 10: 250 to 500 KHz 11: 500 to 1000 KHz	R/W	0x0

6.6.4.57 CONTROL_DPF_OCM_RAM_FW_ADDR_MATCH

Table 6-124. CONTROL_DPF_OCM_RAM_FW_ADDR_MATCH

Address Offset	0x0000 0228		
Physical Address	0x4800 2498	**Instance**	SYSC_GENERAL1
Description	OCM RAM Dynamic Power Framework Handing		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								REGIONOCMRAMFWADDRMATCH																									

Bits	Field Name	Description	Type	Reset
31:20	RESERVED	Read returns reset value	R	0x00
21:0	REGIONOCMRAMFWADDRMATCH	Refer to SMX FW addr_match field	Refer to Table 6-125	0x000000

Table 6-125. Type Value For CONTROL_DPF_OCM_RAM_FW_ADDR_MATCH Register

MPU Mode	NSP	SP
REGIONOCMRAMFWADDRMATCH	R	R/W

6.6.4.58 CONTROL_DPF_OCM_RAM_FW_REQINFO

Table 6-126. CONTROL_DPF_OCM_RAM_FW_REQINFO

Address Offset	0x0000 022C		
Physical Address	0x4800 249C	**Instance**	SYSC_GENERAL1
Description	OCM RAM Dynamic Power Framework Handing		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																REGIONOCMRAMFWREQINFO															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns reset value	R	0x00
15:0	REGIONOCMRAMFWREQINFO	Refer to SMX FW REQINFO permission field	Refer to Table 6-127	0xFFFFFFFF

Table 6-127. Type Value For CONTROL_DPF_OCM_RAM_FW_REQINFO Register

MPU Mode	NSP	SP
REGIONOCMRAMFWREQINFO	R	R/W

6.6.4.59 CONTROL_DPF_OCM_RAM_FW_WR

Table 6-128. CONTROL_DPF_OCM_RAM_FW_WR

Address Offset	0x0000 0230		
Physical Address	0x4800 24A0	**Instance**	SYSC_GENERAL1
Description	OCM RAM Dynamic Power Framework Handing		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																REGIONOCMRAMFWWR															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns reset value	R	0x00000
15:0	REGIONOCMRAMFWWR	Refer to SMX FW WR permission field	Refer to Table 6-129	0xFFFF

Table 6-129. Type Value For CONTROL_DPF_OCM_RAM_FW_WR Register

MPU Mode	NSP	SP
REGIONOCMRAMFWWR	R	R/W

6.6.4.60 CONTROL_DPF_REGION4_GPMC_FW_ADDR_MATCH

Table 6-130. CONTROL_DPF_REGION4_GPMC_FW_ADDR_MATCH

Address Offset	0x0000 0235		
Physical Address	0x4800 24A4	**Instance**	GENERAL
Description	GPMC Dynamic Power Framework Handing		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								REGION4GPMCFWADDRMATCH																									

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Read returns reset value	R	0x00000000
29:0	REGION4GPMCFWADDRMATCH	Exported value to SMX FW region 4 GPMC ADDR_MATCH4 field	Refer to Table 6-131	0x00000000

Table 6-131. Type Value For CONTROL_DPF_REGION4_GPMC_FW_ADDR_MATCH Register

MPU Mode	NSP	SP
REGION4GPMCFWADDRMATCH	R	RW

6.6.4.61 CONTROL_DPF_REGION4_GPMC_FW_REQINFO

Table 6-132. CONTROL_DPF_REGION4_GPMC_FW_REQINFO

Address Offset	0x0000 0238		
Physical Address	0x4800 24A8	**Instance**	GENERAL
Description	GPMC Dynamic Power Framework Handing		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REGION4GPMCFWREQINFO																															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns reset value.	R	0x0000
15:0	REGION4GPMCFWREQINFO	Exported value to SMX FW region 4 GPMC REQINFO_PERMISSION_4 field	Refer to Table 6-133	0xFFFF

Table 6-133. Type Value For CONTROL_DPF_REGION4_GPMC_FW_REQINFO Register

MPU Mode	NSP	SP
REGION4GPMCFWREQINFO	R	R/W

6.6.4.62 CONTROL_DPF_REGION4_GPMC_FW_WR

Table 6-134. CONTROL_DPF_REGION4_GPMC_FW_WR

Address Offset	0x0000 023C		
Physical Address	0x4800 24AC	**Instance**	GENERAL
Description	GPMC Dynamic Power Framework Handing		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																REGION4GPMCFWWR															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns reset value.	R	0x0000
15:0	REGION4GPMCFWWR	Exported value to SMX FW region 4 GPMC WRITE_PERMISSION_4 field	Refer to Table 6-135	0xFFFF

Table 6-135. Type Value For CONTROL_DPF_REGION4_GPMC_FW_WR Register

MPU Mode	NSP	SP
REGION4GPMCFWWR	R	R/W

6.6.4.63 CONTROL_APE_FW_DEFAULT_SECURE_LOCK

Table 6-136. CONTROL_APE_FW_DEFAULT_SECURE_LOCK

Address Offset	0x0000 024C		
Physical Address	0x4800 24BC	**Instance**	GENERAL
Description	This register controls the Firewall security lock features.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			L4TIMERDEFAULTDEBUGLOCK	RESERVED	L4DISPDEFAULTDEBUGLOCK	L4CRYPTODEFAULTDEBUGLOCK	L4COREAPDEFAULTDEBUGLOCK	RESERVED			L4TIMERDEFAULTSECURELOCK	RESERVED	L4DISPDEFAULTSECURELOCK	L4CRYPTODEFAULTSECURELOCK	L4COREAPDEFAULTSECURELOCK	RESERVED				SMSDEFAULTDEBUGLOCK	OCMRAMDEFAULTDEBUGLOCK	IPSSDEFAULTDEBUGLOCK	GPMCDEFAULTDEBUGLOCK	RESERVED				SMSDEFAULTSECURELOCK	OCMRAMDEFAULTSECURELOCK	IPSSDEFAULTSECURELOCK	GPMCDEFAULTSECURELOCK

Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Read returns reset value.	R	0x0
28	L4TIMERDEFAULTDEBUGLOCK	When set to high the TIMER12 default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
27	RESERVED	Read returns reset value.	R	0x0
26	L4DISPDEFAULTDEBUGLOCK	When set to high the DISPLAY default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
25	L4CRYPTODEFAULTDEBUGLOCK	When set to high the CRYPTO default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
24	L4COREAPDEFAULTDEBUGLOCK	When set to high the L4 CORE AP default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
23:21	RESERVED	Read returns reset value.	R	0x0
20	L4TIMERDEFAULTSECURELOCK	When set to high the TIMER12 default region is set as secure	Refer to Table 6-137	0x0
19	RESERVED	Read returns reset value.	R	0x0
18	L4DISPDEFAULTSECURELOCK	When set to high the DISPLAY default region is set as secure	Refer to Table 6-137	0x0
17	L4CRYPTODEFAULTSECURELOCK	When set to high the CRYPTO default region is set as secure	Refer to Table 6-137	Refer to Table 6-138
16	L4COREAPDEFAULTSECURELOCK	When set to high the L4 CORE AP default region is set as secure	Refer to Table 6-137	Refer to Table 6-138
15:12	RESERVED	Read returns reset value.	R	0x0
11	SMSDEFAULTDEBUGLOCK	When set to high the SMS default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
10	OCMRAMDEFAULTDEBUGLOCK	When set to high the OCM-RAM default region is set as debug	Refer to Table 6-137	Refer to Table 6-138

Bits	Field Name	Description	Type	Reset
9	IPSSDEFAULTDEBUGLOCK	When set to high the EMAC, USBOTG, HECC, and VPFE default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
8	GPMCDEFAULTDEBUGLOCK	When set to high the GPMC default region is set as debug	Refer to Table 6-137	Refer to Table 6-138
7:4	RESERVED	Read returns reset value.	R	0x0
3	SMSDEFAULTSECURELOCK	When set to low the SMS default region is set as public	Refer to Table 6-137	Refer to Table 6-138
2	OCMRAMDEFAULTSECURELOCK	When set to low the OCM-RAM default region is set as public	Refer to Table 6-137	Refer to Table 6-138
1	IPSSDEFAULTSECURELOCK	When set to low the EMAC, USBOTG, HECC, and VPFE default region is set as public	Refer to Table 6-137	Refer to Table 6-138
0	GPMCDEFAULTSECURELOCK	When set to low the GPMC default region is set as public	Refer to Table 6-137	Refer to Table 6-138

Table 6-137. Type Value For CONTROL_APE_FW_DEFAULT_SECURE_LOCK Register

MPU Mode	SP	NSP
L4TIMERDEFAULTDEBUGLOCK	R/W	R
L4DISPDEFAULTDEBUGLOCK	R/W	R
L4CRYPTODEFAULTDEBUGLOCK	R/W	R
L4COREAPDEFAULTDEBUGLOCK	R/W	R
L4TIMERDEFAULTSECURELOCK	R/W	R
L4DISPDEFAULTSECURELOCK	R/W	R
L4CRYPTODEFAULTSECURELOCK	R/W	R
L4COREAPDEFAULTSECURELOCK	R/W	R
SMSDEFAULTDEBUGLOCK	R/W	R
OCMRAMDEFAULTDEBUGLOCK	R/W	R
IPSS DEFAULTDEBUGLOCK	R/W	R
GPMCDEFAULTDEBUGLOCK	R/W	R
SMSDEFAULTSECURELOCK	R/W	R
OCMRAMDEFAULTSECURELOCK	R/W	R
IPSS DEFAULTSECURELOCK	R/W	R
GPMCDEFAULTSECURELOCK	R/W	R

Table 6-138. Reset Value For CONTROL_APE_FW_DEFAULT_SECURE_LOCK Register

MPU Mode	SP			NSP
Device Type	G	S/B	E/T	E/T/S/B/G
L4TIMERDEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
L4DISPDEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
L4CRYPTODEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
L4COREAPDEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
L4CRYPTODEFAULTSECURELOCK	0x0	0x1	0x1	0x0
L4COREAPDEFAULTSECURELOCK	0x0	0x1	0x1	0x0
SMSDEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
OCMRAMDEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
GPMCDEFAULTDEBUGLOCK	0x1	0x0	0x1	0x0
SMSDEFAULTSECURELOCK	0x0	0x1	0x1	0x0

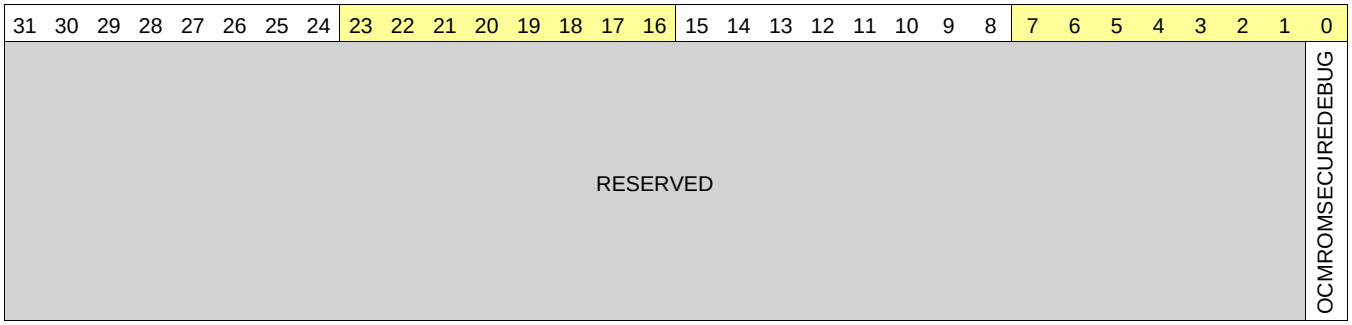
Table 6-138. Reset Value For CONTROL_APE_FW_DEFAULT_SECURE_LOCK Register (continued)

MPU Mode	SP			NSP
Device Type	**G**	**S/B**	**E/T**	**E/T/S/B/G**
OCMRAMDEFAULTSECURELOCK	0x0	0x1	0x1	0x0
GPMCDEFAULTSECURELOCK	0x0	0x1	0x1	0x0

6.6.4.64 CONTROL_OCMROM_SECURE_DEBUG

Table 6-139. CONTROL_OCMROM_SECURE_DEBUG

Address Offset	0x0000 0250		
Physical Address	0x4800 24C0	**Instance**	GENERAL
Description	Secure ROM Code Debug Configuration register		
Type	R/W		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Read returns reset value	R	0x0
0	OCMROMSECUREDEBUG	When set to HIGH the Secure ROM access is granted for debug access 0x0: When cleared SECURE ROM access is not granted for debug purposes 0x1: When set SECURE ROM access is granted for debug purposes	Refer to Table 6-140	Refer to Table 6-141

Table 6-140. Type Value For CONTROL_OCMROM_SECURE_DEBUG Register

MPU Mode	SP	NSP
OCMROMSECUREDEBUG	R/W	R

Table 6-141. Reset Value For CONTROL_OCMROM_SECURE_DEBUG Register

MPU Mode	SP		NSP
Device Type	**E/T**	**S/B/G**	**E/T/S/B/G**
OCMROMSECUREDEBUG	0x1	0x0	0x0

6.6.4.65 CONTROL_EXT_SEC_CONTROL

Table 6-142. CONTROL_EXT_SEC_CONTROL

Address Offset	0x0000 0264		
Physical Address	0x4800 24D4	**Instance**	GENERAL
Description	External security control register		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													I2CSENABLE	CCSECURITYDISABLE	SECUREEXECINDICATOR

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Read returns reset value.	R	0x0
2	I2CSENABLE	I2C module access control 0x0: I2C module access is unrestricted 0x1: I2C module access is allowed in Secure Mode Only	Refer to Table 6-143	0x0
1	CCSECURITYDISABLE	Companion Chip Security Control 0x0: Access to Companion Chip protected registers is not allowed 0x1: Access to Companion Chip protected registers is allowed	Refer to Table 6-143	Refer to Table 6-144
0	SECUREEXECINDICATOR	Secure Execution Indicator 0x0: Secure Execution Indicator Reset 0x1: Secure Execution Indicator Set	Refer to Table 6-143	0x0

Table 6-143. Type Value For CONTROL_EXT_SEC_CONTROL Register

MPU Mode	SP		NSP
Device Type	**E/T**	**G/S/B**	**E/T/S/B/G**
I2CSENABLE	R	R/W	R
CCSECURITYDISABLE	R	R/W	R
SECUREEXECINDICATOR	R	R/W	R

Table 6-144. Reset Value For CONTROL_EXT_SEC_CONTROL Register

MPU Mode	SP		NSP
Device Type	**E/T**	**G/S/B**	**E/T/S/B/G**
CCSECURITYDISABLE	0x1	0x0	0x0

6.6.4.66 CONTROL_DEVCONF2

Table 6-145. CONTROL_DEVCONF2

Address Offset	0x0000 0310		
Physical Address	0x4800 2580	**Instance**	GENERAL
Description	Static device configuration register - 2. Module dedicated functions.		
Type	R/W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					VDD_HVMODEOUT	FUSE_SECIP_DISABLE_GRP2	FUSE_SECIP_DISABLE_GRP1	USBPHY_GPIO_MODE	RESERVED			FUNC_SYS32K_SEL	FUNC_MODE_SEL	RESERVED	FUSE_AVDAC_DISABLE	USBOTG_OTGMODE		USBOTG_SESSENDEN	USBOTG_VBUSDETECTEN	USBOTG_SELINPUTCLKFREQ				USBOTG_PWRCLKGOOD	USBOTG_VBUSSENSE	USBOTG_PHY_PLLON	USBOTG_PHY_RESET	USBOTG_PHY_PD	USBOTG_POWERDOWNOTG	USBOTG_DATAPOLARITY	VPFE_PCLK_INVERT_EN

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved	R	0
26	VDD_HVMODEOUT	IO Voltage detect cell (CQdetect) output: 0 : 1.8V IO operation 1 : 3.3V IO operation	R	
25	FUSE_SECIP_DISABLE_GRP2	Gate the clocks of Group2 Security IPs: 2xAES, 2xDES/3DES, RNG 0: Enabled 1: Disabled	R	-
24	FUSE_SECIP_DISABLE_GRP1	Gate the clocks of Group1 Security IPs: PKA, SHA1/MD5, SHA2/MD5 0: Enabled 1: Disabled	R	-
23	USBPHY_GPIO_MODE	Selects the GPIO mode of USB PHY 0 : USB PHY normal mode operation 1 : UART3 Tx/Rx are coming on DP and DM pins of USBPHY	R/W	0
22:20	Reserved	Reserved	R	0
19	FUNC_SYS32K_SEL	Sys_boot7 value 0 : External 32K Oscillator is selected 1 : Internal 32K clock is selected which is derived from sys clock divided by 800	R	
18	FUNC_MODE_SEL	Sys_boot8 value 0 : Normal mode DDR 32 Bit 1 : Reduced package mode DDR 16 Bit	R	
17	Reserved	Reserved	R	0
16	FUSE_AVDAC_DISABLE	Goes high when AVDAC is disabled using efuse. 0 : AVDAC Disabled 1 : AVDAC Enabled	R	

Bits	Field Name	Description	Type	Reset
15:14	USBOTG_OTGMODE	These two bits will control whether the USB2.0 PHY comparators for VBUS and ID pins are used to drive the VBUSVALID, SESSVALID, SESSEND [3 different levels on VBUS in] and the IDDIG [ID pin value] to the USB controller. In cases where OTG is not needed, it may be more convenient to configure VBUS and ID from MMRs than actually having to drive the VBUS and ID pins to specific levels. Encoding is: OTGMODE = 00 : Do not override phy values. Let PHY drive signals to controller based on its comparators for the VBUS and ID pins. OTGMODE = 01 : Override phy values to force USB Host Operation OTGMODE = 10 : Override phy values to force USB Device Operation OTGMODE = 11 : Override phy values to force USB Host Operation with VBUS low	R/W	3
13	USBOTG_SESSENDEN	1 : Will turn on the Session end comparator	R/W	0
12	USBOTG_VBUSDETECTEN	1 : Will turn on all comparators (except session end comparator) on the vbus line	R/W	0
11:8	USBOTG_SELINPUTCLKFREQ	USB PHY PLL reference clock frequency 0001 : Reserved 0010 : Reserved 0011 : Reserved 0100 : Reserved 0101 : Reserved 0110 : Reserved 0111 : sys_clk 1000 : Reserved 1001 : Reserved	R/W	7
7	USBOTG_PWRCLKGOOD	Goes high when Clock is present from oscillator, all power sources are good and PLL is locked.	R	-
6	USBOTG_VBUSSENSE	Detects the presence of VBUS irrespective of the VDDA supplies. Goes high when VBUS goes above a threshold.	R	-
5	USBOTG_PHY_PLLON	0 : Turn off USB PHY PLL 1 : Turn on the USB PHY PLL	R/W	0
4	USBOTG_PHY_RESET	0 : USB PHY is out of Reset 1 : USB PHY is in Reset	R/W	1
3	USBOTG_PHY_PD	0 : USB PHY is Power Up 1 : USB PHY Power down	R/W	1
2	USBOTG_POWERDOWNOTG	0 : Power up the OTG module of USB PHY 1 : Power down the OTG module of USB PHY	R/W	1
1	USBOTG_DATAPOLARITY	0 : Inverted polarity 1 : Normal polarity	R/W	1
0	VPFE_PCLK_INVERT_EN	0 : Do not Invert the pclk clock coming from IO buffer and going to VPFE module 1 : Invert the pclk clock coming from IO buffer and going to VPFE module	R/W	0

NOTE: DSI is not available on all devices. See Chapter 1, the *Device Family* section, to check availability of this module.

6.6.4.67 CONTROL_DEVCONF3

Table 6-146. CONTROL_DEVCONF3

Address Offset	0x0000 0314		
Physical Address	0x4800 2584	**Instance**	GENERAL
Description	Static device configuration register - 3. Module dedicated functions.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DDRPHY_CLK_EN	EMIF4A_FCLKEN	VTP_PWRSAVE	DDR_CMOSEN	VTP_DRVSTRENGTH			DDR_VREF_EN	DDR_VREF_TAP		VTP_READY		DDR_CONFIG_TERMOFF		DDR_CONFIG_TERMON	

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Reserved	R	0
15	DDRPHY_CLK_EN	This signal is connected to DDR datamacro (CONFIG_CLKEN) signal. Once EMIF is in IDLE mode (Idle/ack protocol is satisfied), this bit can be set to '0' to save the DLL and EMIF4A power 0: DDR PHY DLL clocks are OFF, clock to emif4a from DDR PHY is also gated. 1: DDR PHY clocks are enabled	RW	1
14	EMIF4A_FCLKEN	0 : EMIF clocks will be gated after power-idle protocol is complete 1: EMIF clocks will not be gated after power-idle protocol is complete	RW	0
13	VTP_PWRSAVE	0 : Disbale the power save mode of VTP controller 1: Enable the power save mode of VTP controller	RW	0
12	DDR_CMOSEN	0: Power down the CMOS receive buffer of DDR IO buffers when in DDR2 mode. 1 : Enable the CMOS receive buffer of DDR IO buffers. To be set when mDDR is used.	RW	0
11:9	VTP_DRVSTRENGTH	Drive Strength Control for VTP controller 000 : 0.69*Ext Resistor 001 : 0.79*Ext Resistor 010 : 0.89*Ext Resistor 011 : 0.99*Ext Resistor 100 : 1.09*Ext Resistor 101 : 1.19*Ext Resistor 110 : 1.29*Ext Resistor 111 : 1.39*Ext Resistor	RW	3
8	DDR_VREF_EN	0 : External Reference for VREF cell for DDR 1: Internal Reference for VREF cell for DDR	RW	0

Bits	Field Name	Description	Type	Reset
7:6	DDR_VREF_TAP	Valid only when Internal reference is selected. 00 : Vref = 50 % of VDDS 01 : Vref = 52.5 % of VDDS 10 : Vref = 47.5 % of VDDS 11 : Vref = 50 % of VDDS	RW	0
5	VTP_READY	Status from VTP controller 0 : Not ready 1 : Ready	R	-
4	Reserved	Reserved	R	0
3:2	DDR_CONFIG_TERMOFF	2-bit input signal configuration to disable termination. The macro uses this state to provide a disable termination signal to the IO's while not reading 00 : No termination 01 : Thevenin termination – half termination 10 : Thevenin termination – full termination 11 : Thevenin termination – full termination	RW	0
1:0	DDR_CONFIG_TERMON	2-bit input signal configuration to enable termination. The macro uses this state to provide a enable termination signal to the IO's while in read mode. 00 : No termination 01 : Thevenin termination – half termination 10 : Thevenin termination – full termination 11 : Thevenin termination – full termination	RW	0

6.6.4.68 CONTROL_CBA_PRIORITY

Table 6-147. CONTROL_CBA_PRIORITY

Address Offset	0x0000 0320		
Physical Address	0x4800 2590	**Instance**	GENERAL
Description	CBA Priority register for SCR. Software can change the VBUSP priority of CBA masters by programming this register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CBA_USBCDMA_PRIORITY				RESERVED		CBA_USBM_PRIORITY		RESERVED		CBA_EMAC_PRIORITY		RESERVED		CBA_VPFE_PRIORITY	

Bits	Field Name	Description	Type	Reset
31:15	Reserved	Reserved.	R	0
14:12	CBA_USBCDMA_PRIORITY	VBUSP Priority of USB-CDMA master interface of USBOTG20SS. 0 is the Highest Priority and 7 is the Lowest Priority.	RW	0
11	Reserved	Reserved.	R	0
10:8	CBA_USBM_PRIORITY	VBUSP Priority of USBM master interface of USBOTG20SS. 0 is the Highest Priority and 7 is the Lowest Priority.	RW	0
7	Reserved	Reserved.	R	0
6:4	CBA_EMAC_PRIORITY	VBUSP Priority of CPMAC master interface 0 is the Highest Priority and 7 is the Lowest Priority.	RW	0
3	Reserved	Reserved.	R	0
2:0	CBA_VPFE_PRIORITY	VBUSP Priority of VPFE master interface 0 is the Highest Priority and 7 is the Lowest Priority.	RW	0

6.6.4.69 CONTROL_LVL_INTR_CLEAR

Table 6-148. CONTROL_LVL_INTR_CLEAR

Address Offset	0x0000 0324		
Physical Address	0x4800 2594	**Instance**	GENERAL
Description	Interrupt clear register for VPFE, CPGMAC, and USBOTG.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								VPFE_CCDC_VD2_INT_CLR	VPFE_CCDC_VD1_INT_CLR	VPFE_CCDC_VD0_INT_CLR	USB20OTGSS_USB_INT_CLR	CPGMAC_C0_TX_PULSE_CLR	CPGMAC_C0_RX_THRESH_PULSE_CLR	CPGMAC_C0_RX_PULSE_CLR	CPGMAC_C0_MISC_PULSE_CLR

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Reserved	R	0
15:8	Reserved	Reserved, This field is self clearing	RW	0
7	VPFE_CCDC_VD2_INT_CLR	Writing 1 will clear this interrupt.	RW	0
6	VPFE_CCDC_VD1_INT_CLR	Writing 1 will clear this interrupt.	RW	0
5	VPFE_CCDC_VD0_INT_CLR	Writing 1 will clear this interrupt.	RW	0
4	USB20OTGSS_USB_INT_CLR	Writing 1 will clear this interrupt.	RW	0
3	CPGMAC_C0_TX_PULSE_CLR	Writing 1 will clear this interrupt.	RW	0
2	CPGMAC_C0_RX_THRESH_PULSE_CLR	Writing 1 will clear this interrupt.	RW	0
1	CPGMAC_C0_RX_PULSE_CLR	Writing 1 will clear this interrupt.	RW	0
0	CPGMAC_C0_MISC_PULSE_CLR	Writing 1 will clear this interrupt.	RW	0

6.6.4.70 CONTROL_IP_SW_RESET

Table 6-149. CONTROL_IP_SW_RESET

Address Offset	0x0000 0328		
Physical Address	0x4800 2598	**Instance**	GENERAL
Description	Software reset register for VPFE, HECC, CPGMAC, and USBOTG.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											VPFE_PCLK_SW_RST	HECC_SW_RST	VPFE_VBUSP_SW_RST	CPGMACSS_SW_RST	USB20OTGSS_SW_RST

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reserved	R	0
4	VPFE_PCLK_SW_RST	SW reset for VPFE (PCLK domain) Writing 1 to this register will generate reset to corresponding module. This register is not self clearing. Write 0 before start using that module.	RW	0
3	HECC_SW_RST	SW reset for HECC Writing 1 to this register will generate reset to corresponding module. This register is not self clearing. Write 0 before start using that module.	RW	0
2	VPFE_VBUSP_SW_RST	SW reset for VPFE (VBUSP domain) Writing 1 to this register will generate reset to corresponding module. This register is not self clearing. Write 0 before start using that module.	RW	0
1	CPGMACSS_SW_RST	SW reset for CPGMAC Writing 1 to this register will generate reset to corresponding module. This register is not self clearing. Write 0 before start using that module.	RW	0
0	USB20OTGSS_SW_RST	SW reset for USB20OTGSS Writing 1 to this register will generate reset to corresponding module. This register is not self clearing. Write 0 before start using that module.	RW	0

6.6.4.71 CONTROL_IPSS_CLK_CTRL

Table 6-150. CONTROL_IPSS_CLK_CTRL

Address Offset	0x0000 032C		
Physical Address	0x4800 259C	**Instance**	GENERAL
Description	Clock control register for VPFE, USBOTG, CPGMAC, and HECC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					VPFE_FUNC_CLK_EN	CPGMAC_FUNC_CLK_EN	USB20OTG_FUNC_CLK_EN	HECC_VBUSP_CLK_EN_ACK	VPFE_VBUSP_CLK_EN_ACK	CPGMAC_VBUSP_CLK_EN_ACK	USB20OTG_VBUSP_CLK_EN_ACK	HECC_VBUSP_CLK_EN	VPFE_VBUSP_CLK_EN	CPGMAC_VBUSP_CLK_EN	USB20OTG_VBUSP_CLK_EN

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Reserved	R	0x000000
11	Reserved	Reserved	RW	0
10	VPFE_FUNC_CLK_EN	0 : Disable the Func clock of VPFE 1 : Enable the Func clock of VPFE	RW	1
9	CPGMAC_FUNC_CLK_EN	0 : Disable the Func clock of CPGMAC 1 : Enable the Func clock of CPGMAC	RW	1
8	USB20OTG_FUNC_CLK_EN	0 : Disable the Func clock of USBOTG 1 : Enable the Func clock of USBOTG	RW	1
7	HECC_VBUSP_CLK_EN_ACK	Status of HECC VBUSP clock 0 : Clock Gated 1 : Enabled	R	
6	VPFE_VBUSP_CLK_EN_ACK	Status of VPFE VBUSP clock 0 : Clock Gated 1 : Enabled	R	
5	CPGMAC_VBUSP_CLK_EN_ACK	Status of CPGMAC VBUSP clock 0 : Clock Gated 1 : Enabled	R	
4	USB20OTG_VBUSP_CLK_EN_ACK	Status of USB VBUSP clock 0 : Clock Gated 1 : Enabled	R	
3	HECC_VBUSP_CLK_EN	0 : Disable the VBUSP clock of HECC 1 : Enable the VBUSP clock of HECC	RW	1
2	VPFE_VBUSP_CLK_EN	0 : Disable the VBUSP clock of VPFE 1 : Enable the VBUSP clock of VPFE	RW	1
1	CPGMAC_VBUSP_CLK_EN	0 : Disable the VBUSP clock of CPGMAC 1 : Enable the VBUSP clock of CPGMAC	RW	1
0	USB20OTG_VBUSP_CLK_EN	0 : Disable the VBUSP clock of USBOTG	RW	1

Bits	Field Name	Description	Type	Reset
		1 : Enable the VBUSP clock of USBOTG		

6.6.4.72 CONTROL_IDCODE

Table 6-151. CONTROL_IDCODE

Address Offset	0x307F94		
Physical Address	0x4830 A204	**Instance**	SYSC_GENERAL1
Description	Device IDCODE.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION				HAWKEYE												TI_IDM															Reserved

Bits	Field Name	Description	Type	Reset
31:28	VERSION	Revision number	R	See (1).
27:12	HAWKEYE	Hawkeye number	R	
11:1	TI_IDM	Manufacturer identity (TI)	R	See (1).
0	Reserved	Read returns reset value.	R	1

(1) For more information, see Section 1.5, *Distinguishing Between the Silicon Versions*, in Chapter 1, *Introduction*.

6.6.5 MEM_WKUP Register Descriptions

0x4800 2600 to 0x4800 29FC physical addresses are memories mapped for save and restore location (1Kbytes).

- 0x4800 2600 - 0x4800 2830: non accessible (pad configuration)
- 0x4800 2834 - 0x4800 29FC: user accessible

6.6.6 PADCONFS_WKUP Register Description

Each 32-bit PADCONF register gathers the configuration of two pads. For example, CONTROL.CONTROL_PADCONF_GPMC_7 is used to configure gpmc_7 pad (bits [15:0]) and gpmc_8 pad (bits [31:16]). See Figure 6-8 for more information about PADCONF registers.

According to the pad type, some features are configurable or not. Table 6-56 gives the description of a fully configurable pad.

Table 6-152 describes the reset values, the capabilities, and the corresponding register for each pad.

NOTE:

- All '-' assume that the corresponding bit is not available. These bits are considered as Reserved.
- The reset value for Reserved bits is 0.
- In the MUX Reset States and PU/PD Reset States columns of Table 6-152 , a dash (-) indicates that the field is hardwired to logic 0. No corresponding control block ports are implemented for these bits.

Table 6-152. CONTROL_PADCONF_WKUP_CAPABILITIES

REGISTER NAME	Pad Name	Physical Address	Input Enable	Reserved	PU/PD	MUX Mode
CONTROL_PADCONF_I2C4_SCL[15:0]	i2c4_scl	0x4800 2A00	0b1	0b000	0b11	0b000
CONTROL_PADCONF_I2C4_SCL[31:16]	i2c4_sda	0x4800 2A00	0b1	0b000	0b11	0b000
CONTROL_PADCONF_SYS_32K[15:0]	sys_32k	0x4800 2A04	0b1	0b000	--	---
CONTROL_PADCONF_SYS_32K[31:16]	sys_clkreq	0x4800 2A04	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_NRESWARM [15:0]	sys_ nreswarm	0x4800 2A08	0b1	0b000	0b11	0b000
CONTROL_PADCONF_SYS_NRESWARM [31:16]	sys_boot0	0x4800 2A08	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT1[15:0]	sys_boot1	0x4800 2A0C	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT1[31:16]	sys_boot2	0x4800 2A0C	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT3[15:0]	sys_boot3	0x4800 2A10	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT3[31:16]	sys_boot4	0x4800 2A10	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT5[15:0]	sys_boot5	0x4800 2A14	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT5[31:16]	sys_boot6	0x4800 2A14	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_BOOT7[15:0]	sys_boot7	0x4800 2218	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SDRC_STRBEN_DLY1[31:16]	sys_boot8	0x4800 2224	0b1	0b000	0b00	0b000
CONTROL_PADCONF_SYS_CLKOUT1 [31:16]	sys_clkout1	0x4800 2A18	0b1	0b000	0b01	0b111
CONTROL_PADCONF_JTAG_NTRST[15:0]	jtag_ntrst	0x4800 2A1C	0b1	0b000	0b01	---
CONTROL_PADCONF_JTAG_NTRST[31:16]	jtag_tck	0x4800 2A1C	0b1	0b000	0b01	---
CONTROL_PADCONF_JTAG_TMS_TMSC [15:0]	jtag_tms_ tmsc	0x4800 2A20	0b1	0b000	0b11	---
CONTROL_PADCONF_JTAG_TMS_TMSC [31:16]	jtag_tdi	0x4800 2A20	0b1	0b000	0b11	---
CONTROL_PADCONF_JTAG_EMU0[15:0]	jtag_emu0	0x4800 2A24	0b1	0b000	0b11	0b000
CONTROL_PADCONF_JTAG_EMU0[31:16]	jtag_emu1	0x4800 2A24	0b1	0b000	0b11	0b000
CONTROL_PADCONF_JTAG_RTCK [31:16]	jtag_rtck	0x4800 2A4C	-	0b000	0b01	---
CONTROL_PADCONF_JTAG_TDO[15:0]	jtag_tdo	0x4800 2A50	-	0b000	--	---

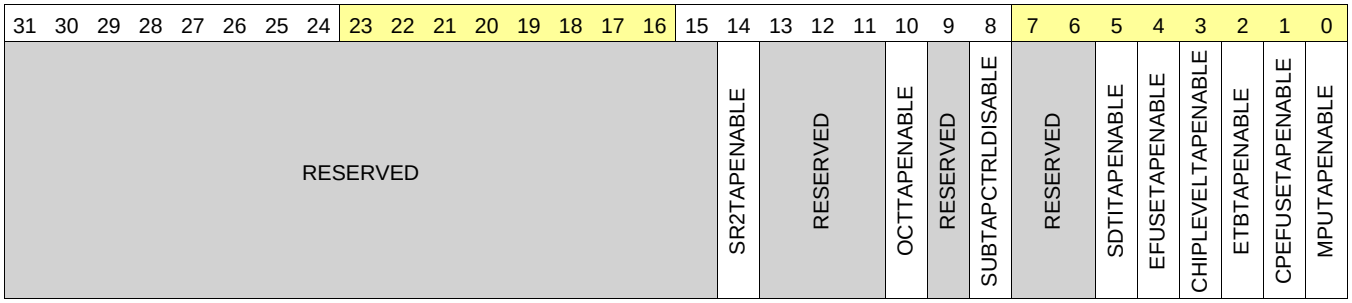
6.6.7 GENERAL_WKUP Register Descriptions

Table 6-153 through Table 6-169 describe the PADCONFS registers bits.

6.6.7.1 CONTROL_SEC_TAP

Table 6-153. CONTROL_SEC_TAP

Address Offset	0x0000 0000		
Physical Address	0x4800 2A60	**Instance**	GENERAL_WKUP
Description	Security TAP controllers register		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns reset value.	R	0x0000
14	SR2TAPENABLE	Smart Reflex2 TAP control 0x0: Smart Reflex2 TAP controller is disabled 0x1: Smart Reflex2 TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155
13:11	RESERVED	Read returns reset value.	R	0x0
10	OCTTAPENABLE	OCT TAP Control 0x0: OCT TAP controller is disabled 0x1: OCT TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155
9	RESERVED	Read returns reset value.	R	0x0
8	SUBTAPCTRLDISABLE	Restrict writable register list accessible through the Chip Level TAP 0x0: Writable register list is unrestricted 0x1: Writable register list is restricted	Refer to Table 6-154	0x0
7:6	RESERVED	Read returns reset value.	R	0x0
5	SDTITAPENABLE	SDTI TAP control 0x0: SDTI TAP controller is disabled 0x1: SDTI TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155
4	EFUSETAPENABLE	E-Fuse TAP control 0x0: E-Fuse TAP controller is disabled 0x1: E-Fuse TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155
3	CHIPLEVELTAPENABLE	Chip Level TAP control 0x0: Chip-Level TAP controller is disabled 0x1: Chip-Level TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155
2	ETBTAPENABLE	ETB TAP control 0x0: ETB TAP controller is disabled 0x1: ETB TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155

Bits	Field Name	Description	Type	Reset
1	CPEFUSETAPENABLE	CP Efuse TAP control 0x0: CP Efuse TAP controller is disabled 0x1: CP Efuse TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155
0	MPUTAPENABLE	MPU / ICECrusher / ETM / PSA TAP control 0x0: MPU TAP controller is disabled 0x1: MPU TAP controller is enabled	Refer to Table 6-154	Refer to Table 6-155

Table 6-154. Type Value For CONTROL_SEC_TAP Register

CONTROL_SEC_TAP[31] SECTAPWRDISABLE bit	0	1				
MPU Mode	NSP/SP	SP			NSP	
Device Type	E/T/S/B/G	S/B	E/T	G	E/T/S/B	G
SECTAPWRDISABLE	R	R/OCO	R/OCO	R/OCO	R	R/OCO
SR2TAPENABLE	R	R/W	R/W	R/W	R	R/W
OCTTAPENABLE	R	R/W	R/W	R/W	R	R/W
SUBTAPCTRLDISABLE	R	R/W1toClr	R/W1toClr	R/W1toClr	R	R/W1toClr
SDTITAPENABLE	R	R/W	R/W	R/W	R	R/W
EFUSETAPENABLE	R	R/W	R/W	R/W	R	R/W
CHIPLEVELTAPENABLE	R	R/W	R/W	R/W	R	R/W
ETBTAPENABLE	R	R/W	R/W	R/W	R	R/W
CPEFUSETAPENABLE	R	R/W	R/W	R/W	R	R/W
MPUTAPENABLE	R	R/W	R/W	R/W	R	R/W

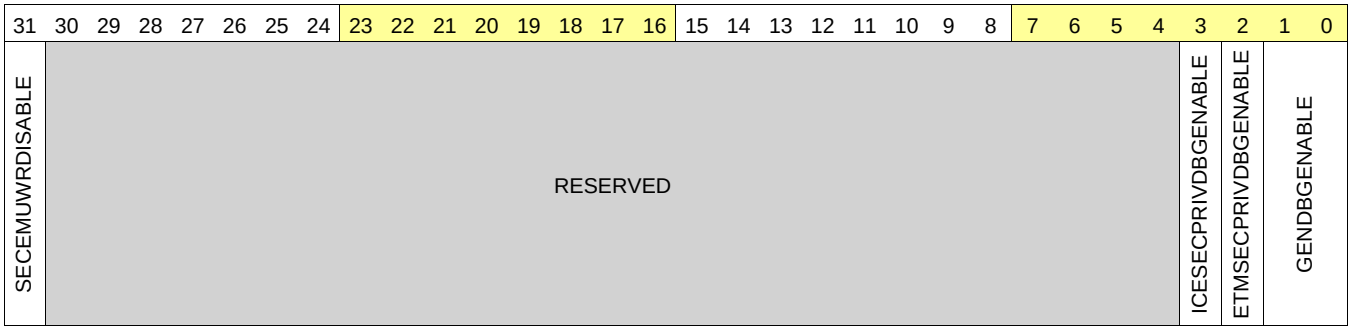
Table 6-155. Reset Value For CONTROL_SEC_TAP Register

MPU Mode	SP			NSP	
Device Type	S/B	E/T	G	E/T/S/B	G
SR2TAPENABLE	0x0	0x1	0x1	0x0	0x1
OCTTAPENABLE	0x1	0x1	0x1	0x0	0x1
SDTITAPENABLE	0x0	0x1	0x1	0x0	0x1
EFUSETAPENABLE	0x1	0x1	0x1	0x0	0x1
CHIPLEVELTAPENABLE	0x1	0x1	0x1	0x0	0x1
ETBTAPENABLE	0x0	0x1	0x1	0x0	0x1
CPEFUSETAPENABLE	0x1	0x1	0x1	0x0	0x1
MPUTAPENABLE	0x0	0x1	0x1	0x0	0x1

6.6.7.2 CONTROL_SEC_EMU

Table 6-156. CONTROL_SEC_EMU

Address Offset	0x0000 0004		
Physical Address	0x4800 2A64	**Instance**	GENERAL_WKUP
Description	Security emulation register		
Type	RW		



Bits	Field Name	Description	Type	Reset
31	SECEMUWRDISABLE	Security EMULATION register write disable control 0x0: Write in Security Emulation Register is allowed 0x1: Write in Security Emulation Register is forbidden	Refer to Table 6-157	Refer to Table 6-158
30:4	RESERVED	Read returns reset value.	R	0x0
3	ICESECPRIVDBGENABLE	ICE Secure Privilege debug control 0x0: MPU trace data are not captured along secure execution 0x1: MPU trace data are captured along secure execution	Refer to Table 6-157	Refer to Table 6-158
2	ETMSECPRIVDBGENABLE	ETM Secure Privilege Control 0x0: MPU trace data are not captured along public execution 0x1: MPU trace data are captured along public execution	Refer to Table 6-157	Refer to Table 6-158
1:0	GENDBGENABLE	Generic Debug Control 0x0: Generic debug is disabled 0x1: Strict Public debug mode; any attempt to execute secure code generates a security violation 0x2: Public Debug mode. Debug allowed in public code. Secure code can only be executed. 0x3: Secure debug mode. Debug allowed for public and secure code	Refer to Table 6-157	Refer to Table 6-158

Table 6-157. Type Value For CONTROL_SEC_EMU Register

CONTROL_SEC_EMU[31] SECEMUWRDISABLE bit	**0**	**1**				
MPU Mode	**NSP/SP**	**SP**			**NSP**	
Device Type	**E/T/S/B/G**	**S/B**	**E/T**	**G**	**E/T/S/B**	**G**
SECEMUWRDISABLE	R	R/OCO	R/OCO	R	R	R
ICESECPRIVDBGENABLE	R	R/OCO	R/W	R	R	R
ETMSECPRIVDBGENABLE	R	R/OCO	R/W	R	R	R
GENDBGENABLE	R	R/OCO	R/W	R	R	R

Table 6-158. Reset Value For CONTROL_SEC_EMU Register

MPU Mode	SP			NSP	
Device Type	S/B	E/T	G	E/T/S/B	G
SECEMUWRDISABLE	0x0	0x0	0x1	0x0	0x1
ICESECPRIVDBGENABLE	0x0	0x1	0x0	0x0	0x0
ETMSECPRIVDBGENABLE	0x0	0x1	0x0	0x0	0x0
GENDBGENABLE	0b00	0b11	0b10	0x0	0b010

6.6.7.3 CONTROL_WKUP_DEBOBS_0

Table 6-159. CONTROL_WKUP_DEBOBS_0

Address Offset	0x0000 0008		
Physical Address	0x4800 2A68	**Instance**	GENERAL_WKUP
Description	Select the WKUP domain set of signals to be observed for hw_dbg3, hw_dbg2, hw_dbg1, hw_dbg0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			OBSMUX3					RESERVED			OBSMUX2					RESERVED			OBSMUX1					RESERVED			OBSMUX0			

Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Read returns reset value.	R	0x0
28:24	OBSMUX3	Select the set of signals to be observed for hw_dbg3	Refer to Table 6-160	0x00
23:21	RESERVED	Read returns reset value.	R	0x0
20:16	OBSMUX2	Select the set of signals to be observed for hw_dbg2	Refer to Table 6-160	0x00
15:13	RESERVED	Read returns reset value.	R	0x0
12:8	OBSMUX1	Select the set of signals to be observed for hw_dbg1	Refer to Table 6-160	0x00
7:5	RESERVED	Read returns reset value.	R	0x0
4:0	OBSMUX0	Select the set of signals to be observed for hw_dbg0	Refer to Table 6-160	0x00

Table 6-160. Type Value For CONTROL_WKUP_DEBOBS_0 Register

Device Type	E/T/G		S/B
CONTROL_WKUP_DEBOBS_4[31] WKUPOBSERVABILITYDISABLE bit	**0**	**1**	**x**
OBSMUX3	RW	R	R
OBSMUX2	RW	R	R
OBSMUX1	RW	R	R
OBSMUX0	RW	R	R

6.6.7.4 CONTROL_WKUP_DEBOBS_1

Table 6-161. CONTROL_WKUP_DEBOBS_1

Address Offset	0x0000 000C		
Physical Address	0x4800 2A6C	**Instance**	GENERAL_WKUP
Description	Select the WKUP domain set of signals to be observed for hw_dbg7, hw_dbg6, hw_dbg5, hw_dbg4		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			OBSMUX7					RESERVED			OBSMUX6					RESERVED			OBSMUX5					RESERVED			OBSMUX4			

Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Read returns reset value.	R	0x0
28:24	OBSMUX7	Select the set of signals to be observed for hw_dbg7	Refer to Table 6-162	0x00
23:21	RESERVED	Read returns reset value.	R	0x0
20:16	OBSMUX6	Select the set of signals to be observed for hw_dbg6	Refer to Table 6-162	0x00
15:13	RESERVED	Read returns reset value.	R	0x0
12:8	OBSMUX5	Select the set of signals to be observed for hw_dbg5	Refer to Table 6-162	0x00
7:5	RESERVED	Read returns reset value.	R	0x0
4:0	OBSMUX4	Select the set of signals to be observed for hw_dbg4	Refer to Table 6-162	0x00

Table 6-162. Type Value For CONTROL_WKUP_DEBOBS_1 Register

Device Type	E/T/G		S/B
CONTROL_WKUP_DEBOBS_4[31] WKUPOBSERVABILITYDISABLE bit	**0**	**1**	**x**
OBSMUX7	RW	R	R
OBSMUX6	RW	R	R
OBSMUX5	RW	R	R
OBSMUX4	RW	R	R

6.6.7.5 CONTROL_WKUP_DEBOBS_2

Table 6-163. CONTROL_WKUP_DEBOBS_2

Address Offset	0x0000 0010		
Physical Address	0x4800 2A70	**Instance**	GENERAL_WKUP
Description	Select the WKUP domain set of signals to be observed for hw_dbg11 hw_dbg10, hw_dbg9, hw_dbg8		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			OBSMUX11					RESERVED			OBSMUX10					RESERVED			OBSMUX9					RESERVED			OBSMUX8			

Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Read returns reset value.	R	0x0
28:24	OBSMUX11	Select the set of signals to be observed for hw_dbg11	Refer to Table 6-164	0x00
23:21	RESERVED	Read returns reset value.	R	0x0
20:16	OBSMUX10	Select the set of signals to be observed for hw_dbg10	Refer to Table 6-164	0x00
15:13	RESERVED	Read returns reset value.	R	0x0
12:8	OBSMUX9	Select the set of signals to be observed for hw_dbg9	Refer to Table 6-164	0x00
7:5	RESERVED	Read returns reset value.	R	0x0
4:0	OBSMUX8	Select the set of signals to be observed for hw_dbg8	Refer to Table 6-164	0x00

Table 6-164. Type Value For CONTROL_WKUP_DEBOBS_2 Register

Device Type	E/T/G		S/B
CONTROL_WKUP_DEBOBS_4[31] WKUPOBSERVABILITYDISABLE bit	**0**	**1**	**x**
OBSMUX11	RW	R	R
OBSMUX10	RW	R	R
OBSMUX9	RW	R	R
OBSMUX8	RW	R	R

6.6.7.6 CONTROL_WKUP_DEBOBS_3

Table 6-165. CONTROL_WKUP_DEBOBS_3

Address Offset	0x0000 0014		
Physical Address	0x4800 2A74	**Instance**	GENERAL_WKUP
Description	Select the WKUP domain set of signals to be observed for hw_dbg15 hw_dbg14, hw_dbg13, hw_dbg12		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			OBSMUX15					RESERVED			OBSMUX14					RESERVED			OBSMUX13					RESERVED			OBSMUX12			

Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Read returns reset value.	R	0x0
28:24	OBSMUX15	Select the set of signals to be observed for hw_dbg15	Refer to Table 6-166	0x00
23:21	RESERVED	Read returns reset value.	R	0x0
20:16	OBSMUX14	Select the set of signals to be observed for hw_dbg14	Refer to Table 6-166	0x00
15:13	RESERVED	Read returns reset value.	R	0x0
12:8	OBSMUX13	Select the set of signals to be observed for hw_dbg13	Refer to Table 6-166	0x00
7:5	RESERVED	Read returns reset value.	R	0x0
4:0	OBSMUX12	Select the set of signals to be observed for hw_dbg12	Refer to Table 6-166	0x00

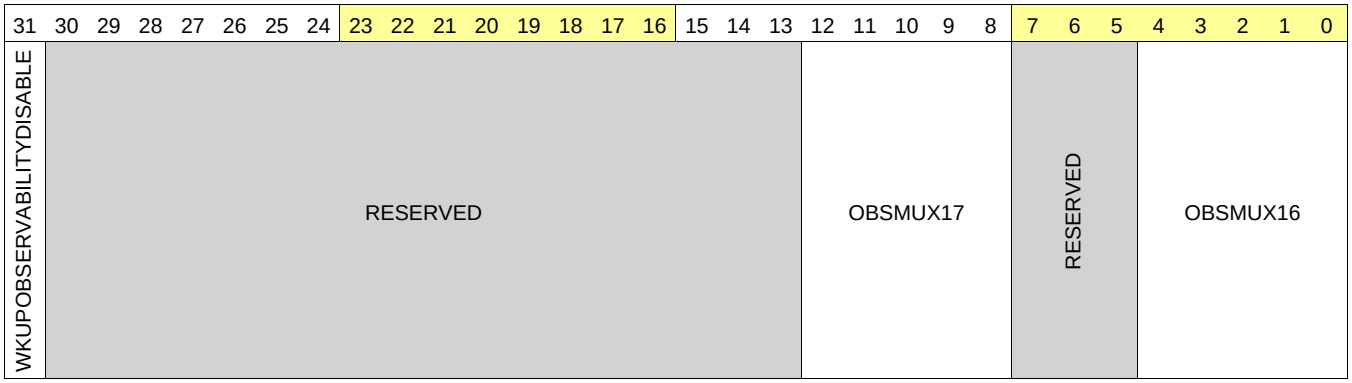
Table 6-166. Type Value For CONTROL_WKUP_DEBOBS_3 Register

Device Type	E/T/G		S/B
CONTROL_WKUP_DEBOBS_4[31] WKUPOBSERVABILITYDISABLE bit	**0**	**1**	**x**
OBSMUX15	RW	R	R
OBSMUX14	RW	R	R
OBSMUX13	RW	R	R
OBSMUX12	RW	R	R

6.6.7.7 CONTROL_WKUP_DEBOBS_4

Table 6-167. CONTROL_WKUP_DEBOBS_4

Address Offset	0x0000 0018		
Physical Address	0x4800 2A78	**Instance**	GENERAL_WKUP
Description	Select the WKUP domain set of signals to be observed for hw_dbg17, hw_dbg16		
Type	RW		



Bits	Field Name	Description	Type	Reset
31	WKUPOBSERVABILITY DISABLE	Control the observability feature 0x0: Observability can be configured through the ObsMux bit field in CONTROL_DEBOBS register 0x1: Observability is disabled. If pads are configured for the 'hardware debug', output is tied low	Refer to Table 6-168	0x0
30:13	RESERVED	Read returns reset value.	R	0x00000
12:8	OBSMUX17	Select the set of signals to be observed for hw_dbg17	Refer to Table 6-168	0x00
7:5	RESERVED	Read returns reset value.	R	0x0
4:0	OBSMUX16	Select the set of signals to be observed for hw_dbg16	Refer to Table 6-168	0x00

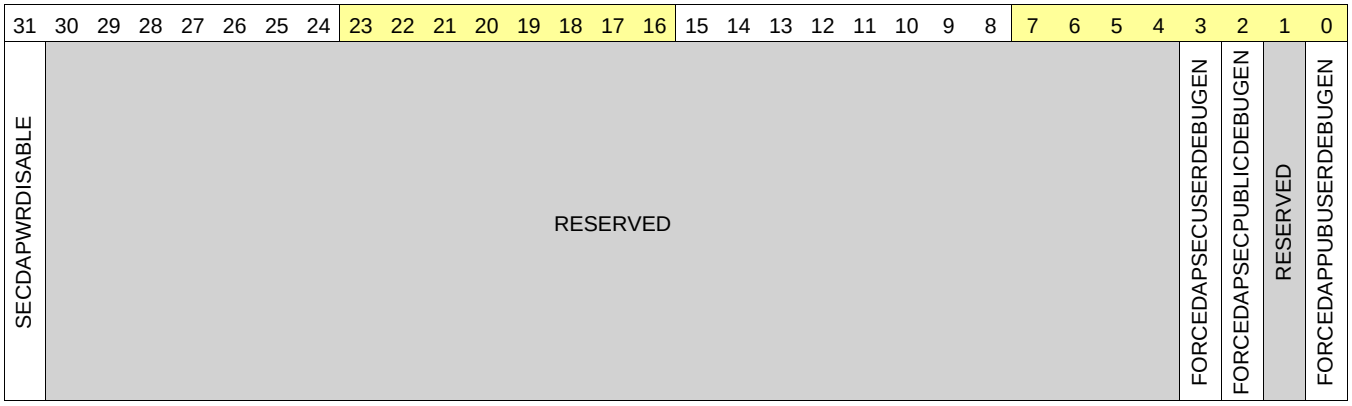
Table 6-168. Type Value For CONTROL_WKUP_DEBOBS_4 Register

Device Type	E/T/G		S/B
CONTROL_WKUP_DEBOBS_4[31] WKUPOBSERVABILITYDISABLE bit	**0**	**1**	**x**
WKUPOBSERVABILITYDISABLE	R/OCO	R	R
OBSMUX17	RW	R	R
OBSMUX16	RW	R	R

6.6.7.8 CONTROL_SEC_DAP

Table 6-169. CONTROL_SEC_DAP

Address Offset	0x0000 001C		
Physical Address	0x4800 2A7C	**Instance**	GENERAL_WKUP
Description	DAP Qualifiers generated using this register		
Type	RW		



Bits	Field Name	Description	Type	Reset
31	SECDAPWRDISABLE	Security DAP Register write disable control 0x0: Write in Security DAP Register is allowed 0x1: Write in Security DAP Register is forbidden	Refer to Table 6-170	0x0
30:4	RESERVED	Read returns reset value	R	0x0000000
3	FORCEDAPSECUSERDEBUGEN	Force MreqSupervisor to 0 for secure DAP accesses 0x0: DAP properties unchanged 0x1: DAP with MreqSupervisor = 0b0	Refer to Table 6-170	0x0
2	FORCEDAPSECSECUREDEBUGEN	Force MreqSecure to 0 for secure DAP accesses 0x0: DAP properties unchanged 0x1: DAP with MreqSecure = 0b0	Refer to Table 6-170	0x0
1	RESERVED	Read returns reset value	R	0x0
0	FORCEDAPPUBUSERDEBUGEN	Force MreqSupervisor to 0 for public DAP accesses 0x0: DAP properties unchanged 0x1: DAP with MreqSupervisor = 0b0	Refer to Table 6-170	0x0

Table 6-170. Type Value For CONTROL_SEC_DAP Register

CONTROL_SEC_DAP[31] SECDAPWRDISABLE bit	0	1	
MPU Mode	**NSP/SP**	**SP**	**NSP**
SECDAPWRDISABLE	R	R/OCO	R
FORCEDAPSECUSERDEBUGEN	R	R/W	R
FORCEDAPSECPUBLICDEBUGEN	R	R/W	R
FORCEDAPPUBUSERDEBUGEN	R	R/W	R

6.7 Revision History

Table 6-171 lists the changes made since the previous version of this document.

Table 6-171. Document Revision History

Reference	Additions/Modifications/Deletions
Global	Removed CONTROL_SEC_CTRL register.
Global	Removed the CONTROL_GENERAL_PURPOSE_STATUS register.
Global	Removed CONTROL_MSUSPENDMUX_3 register.
Global	Changed CONTROL_CBA_PRIORITY address offset from 0x0000 031C to 0x0000 0320.
Global	Changed CONTROL_CBA_PRIORITY physical address from 0x4800 258C to 0x4800 2590.
Global	Changed CONTROL_LVL_INTR_CLEAR address offset from 0x0000 031C & 0x0000 0320 to 0x0000 0324.
Global	Changed CONTROL_LVL_INTR_CLEAR physical address from 0x4800 258C & 0x4800 2590 to 0x4800 2594.
Global	Changed CONTROL_IP_SW_RESET address offset from 0x0000 0320 & 0x0000 0324 to 0x0000 0328.
Global	Changed CONTROL_IP_SW_RESET physical address from 0x4800 2590 & 0x4800 2594 to 0x4800 2598.
Global	Changed CONTROL_IPSS_CLK_CTRL address offset from 0x0000 0324 & 0x0000 0328 to 0x0000 032C.
Global	Changed CONTROL_IPSS_CLK_CTRL physical address from 0x4800 2594 & 0x4800 2598 to 0x4800 259C.
Global	Deleted reserved register CONTROL_DEVCONF4.
Global	Changed CONTROL_PADCONF_SDRC_CKE physical address from 0x4800 2260 to 0x4800 2264.
Global	Changed CONTROL_PADCONF_SDRC_CKE address offset from 0x0000 0230 to 0x0000 0234.
Global	Changed CONTROL_PADCONF_SDRC_CKE0 physical address from 0x4800 2260 to 0x4800 2264.
Global	Changed CONTROL_PADCONF_SDRC_CKE0 address offset from 0x0000 0230 to 0x0000 0234.
Global	Changed CONTROL_PADCONF_SDRC_CKE1 physical address from 0x4800 2260 to 0x4800 2264.
Global	Changed CONTROL_PADCONF_SDRC_CKE1 address offset from 0x0000 0230 to 0x0000 0234.
Global	Removed CONTROL_RAND_KEY_0 register.
Global	Removed CONTROL_RAND_KEY_1 register.
Global	Removed CONTROL_RAND_KEY_2 register.
Global	Removed CONTROL_RAND_KEY_3 register
Global	Removed CONTROL_CUST_KEY_0 register.
Global	Removed CONTROL_CUST_KEY_1 register.
Global	Removed CONTROL_CUST_KEY_2 register.
Global	Removed CONTROL_CUST_KEY_3 register.
Section 6.1	Changed bullets.
	Removed Memory DFT Read/Write Control Registers section.
	Removed Control Dynamic Power Framework Registers section.
Section 6.4.4	Removed 5th paragraph.
Table 6-32	Changed PRCM_SECURE_WD_RST to reserved.
Table 6-34	Changed PRCM_ICEPICK_POR to reserved.
Table 6-43	Changed PRCM_GlblSecureRst_n to reserved.
Table 6-44	Changed PRCM_MPU_SECURITY_VIOL_RST to reserved.
Section 6.4.6.1	Removed last 2 paragraphs.
Table 6-5	Removed hsusb1_tll_... signals from Mode 6 column.
Section 6.4.5.2	Added Caution.
	Removed Security Registers section.
	Removed Security Status Registers section.
Section 6.4.5.3	Changed section.
	Removed OCMROM Secure Debug Register section.
	Removed APE Firewall Default Secure Lock Register section.
	Removed External Security Control Register section.
	Removed Keys Access Registers section.
Section 6.6.4.12	Changed bits 2 and 3 to reserved.

Table 6-171. Document Revision History (continued)

Reference	Additions/Modifications/Deletions
Section 6.6.4.66	Changed bit 25 and 24 to reserved.
Section 6.6.7.1	Changed bit 31 to reserved.
	Removed Type Value For CONTROL_DEBOBS_0-8 Register tables.

DMA

This chapter describes the system direct memory access (SDMA).

Topic **Page**

7.1 SDMA Module Overview

The System Direct Memory Access (SDMA), also called DMA4, performs high-performance data transfers between memories and peripheral devices without microprocessor unit (MPU) support during transfer. A DMA transfer is programmed through a logical DMA channel, which allows the transfer to be optimally tailored to the requirements of the application.

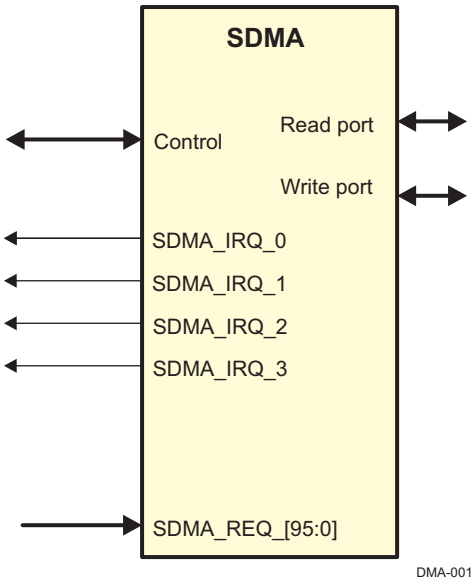
The device also embeds dedicated DMA controllers: the camera image signal processor (ISP) DMA; the enhanced DMA (EDMA), which is embedded in the IVA2.2 subsystem; the display DMA; and the universal serial bus (USB) high-speed (HS) DMA. For more information, see , *Camera ISP Subsystem*; , *IVA Subsystem*; Chapter 12, *Display Subsystem*; and Chapter 20, *High-Speed USB Controllers*.

The DMA controller includes the following main features:

- Data transfer support in either direction between:
  - Memory and memory
  - Memory and peripheral device
- 32 logical DMA channels supporting:
  - Multiple concurrent transfers
  - Independent transfer profile for each channel
  - 8-bit, 16-bit, or 32-bit data element transfer size
  - Software-triggered or hardware-synchronized transfers
  - Flexible source and destination address generation
  - Burst read and write
  - Chained multiple-channel transfers
  - Endianism conversion
- First-come, first-serve DMA scheduling with fixed priority
- Up to 96 DMA requests
- Constant fill
- Transparent copy
- Four programmable interrupt request output lines
- Software or hardware enabling
- FIFO depth: 256 x 32-bits
- Data buffering
- FIFO budget allocation
- Power-management support
- Auto-idle power-saving support
- Implementation of retention flip-flops (RFFs) to support dynamic power saving (DPS) between system power modes without MPU involvement

Figure 7-1 shows an overview of the SDMA module.

Figure 7-1. SDMA Overview



The SDMA module has two ports—one read and one write—and provides multiple logical channel support. A dynamically allocated FIFO queue memory pool provides buffering between the read and write ports.

The MPU configures the SDMA through the L4 interconnect.

7.2 SDMA Controller Environment

7.2.1 Environment Overview

The read and write ports of the SDMA controller are connected to the device through the L3 interconnect. The control port is connected to the L4 interconnect. The SDMA interrupt lines are connected to the interrupt lines of the MPU interrupt controller. The SDMA can handle up to 96 DMA requests, including four external hardware DMA requests.

7.2.2 SDMA Request Scheme

The hardware DMA request line schemes can be either edge-sensitive or transition-sensitive.

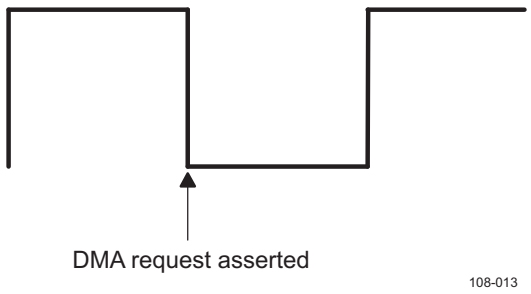
The sensitivity selection of the DMA request line can be configured in the system control module with the following register bits:

- CONTROL.CONTROL_DEVCONF0[0] SENSDMAREQ0 register bit for sys_ndmareq0
- CONTROL.CONTROL_DEVCONF0[1] SENSDMAREQ1 register bit for sys_ndmareq1
- CONTROL.CONTROL_DEVCONF1[7] SENSDMAREQ2 register bit for sys_ndmareq2
- CONTROL.CONTROL_DEVCONF1[8] SENSDMAREQ3 register bit for sys_ndmareq3

The default scheme is transition sensitive. All internal peripherals of the device use the transition-sensitive scheme.

Figure 7-2 shows the DMA request captured on a falling edge in the edge-sensitive scheme.

Figure 7-2. Edge-Sensitive DMA Request Scheme



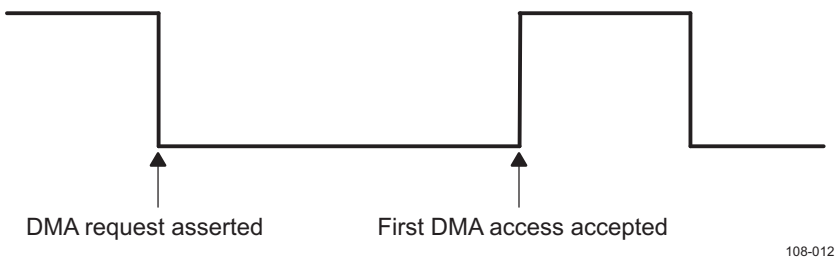
For a transition-sensitive DMA request (see Figure 7-3), the line must be maintained low (asserted) until the first DMA access is complete, after which the line must be maintained high (deasserted) for a minimum of one clock cycle (CORE_L3_ICLK):

When the deassertion time is less than one clock cycle, the SDMA might not detect the deassertion.

When the channel is enabled one cycle after a DMA request is disabled, the channel detects the DMA request and starts the corresponding transfer.

When the channel is enabled two cycles after the DMA request is disabled, the channel does not detect the DMA request.

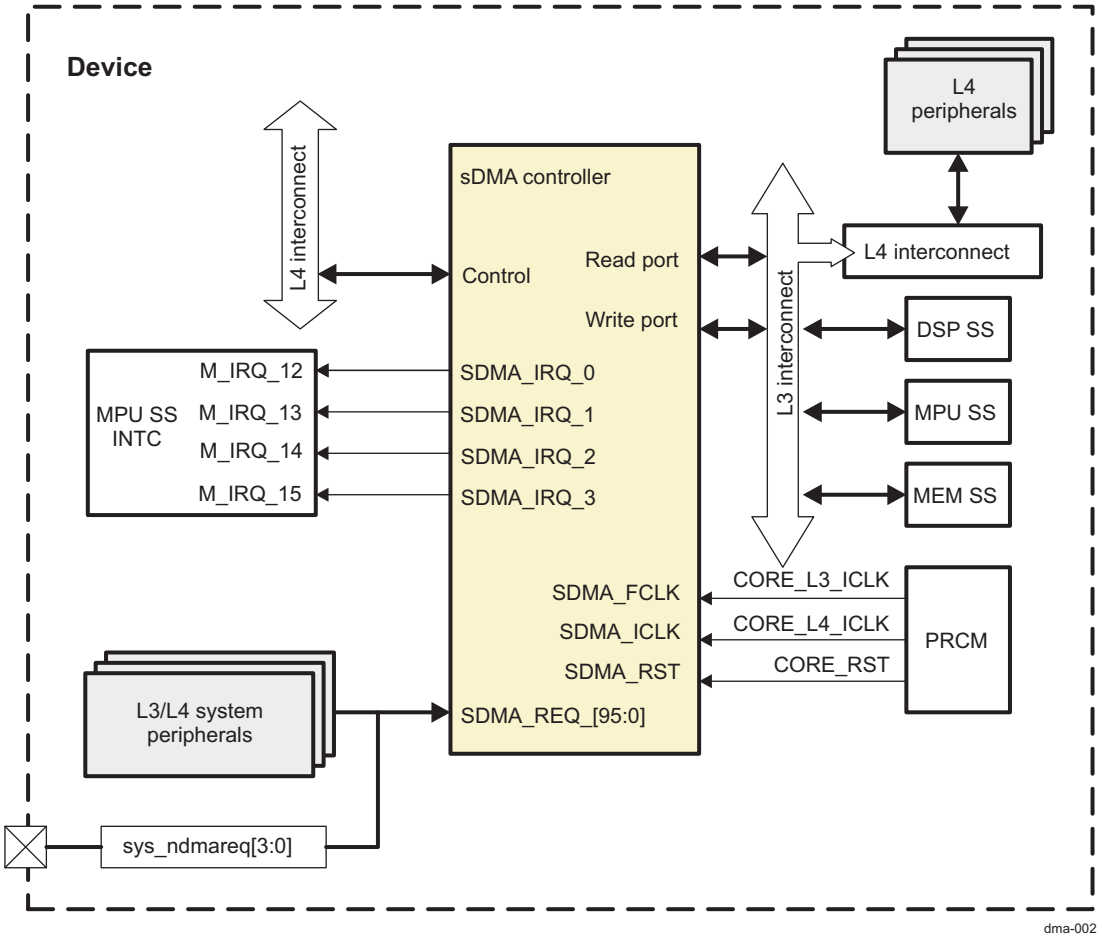
Figure 7-3. Transition-Sensitive DMA Request Scheme



7.3 SDMA Module Integration

Figure 7-4 highlights the SDMA controller integration.

Figure 7-4. SDMA Controller Integration



7.3.1 External SDMA Request Interface Description

The sys_ndmareq pins are optional external DMA requests used by external devices to establish direct hardware synchronization with the SDMA controller. A logical channel can be configured to respond to an external synchronization request. These requests can be configured to either active low on level or falling edge active by the control module.

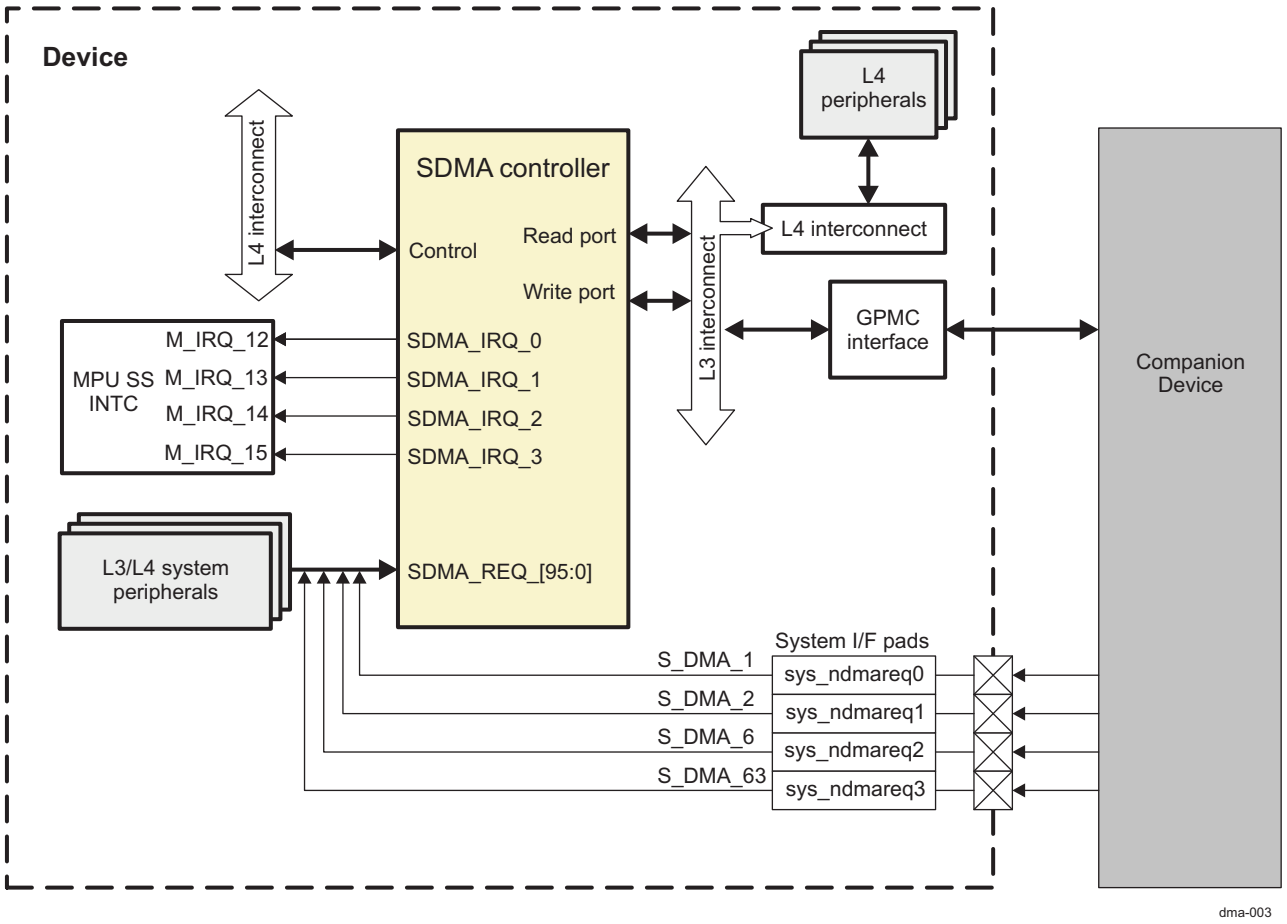
Table 7-1 shows the I/O description of the external DMA request pin.

Table 7-1. Description External DMA Request Pin

Signal Name	IO	Description	Reset
sys_ndmareq0	I	External DMA request to the SDMA controller	Unknown
sys_ndmareq1	I	External DMA request to the SDMA controller	Unknown
sys_ndmareq2	I	External DMA request to the SDMA controller	Unknown
sys_ndmareq3	I	External DMA request to the SDMA controller	Unknown

Figure 7-5 shows the SDMA environment with an example of how to use the external hardware DMA request pins.

Figure 7-5. Example of External DMA Requests Use to the SDMA Controller



For example, an external device can use the external DMA request pins to start a logical channel transfer over the general-purpose memory controller (GPMC) interface. The transfer can be a memory-to-memory transfer in which the source memory is in the external device.

The external DMA request signals are not available on external pins by default after cold reset. See Chapter 6, *System Control Module*, for instructions on multiplexing out the seven signal lines to pins.

7.3.2 Clocking, Reset, and Power-Management Scheme

7.3.2.1 Clocking

The SDMA controller uses two clock domains:

- CORE_L4_ICLK supports the configuration port.
- CORE_L3_ICLK is both a functional clock for all internal logic and for the two master read and write ports.

The SDMA controller supports a software-controlled standby mode with an input clock shutoff. Setting the PRCM.CM_IDLEST1_CORE[2] ST_DMA status bit to 1 allows detection of the SDMA power mode.

For more information about power management and clock idle, see Section 7.4.20, *Power Management*.

7.3.2.2 Resets

7.3.2.2.1 Asynchronous Hardware Reset

The SDMA controller is part of the CORE_RST reset domain.

7.3.2.2.2 Software Reset Through the Configuration Port

The SDMA controller can be reset independently by software through the SDMA.DMA4_OCP_SYSCONFIG[1] SOFTRESET bit. Setting the bit to 1 enables an active software reset that is functionally equivalent to a hardware reset.

Hardware and software resets initialize all of the logic in the SDMA module and the global registers and some of the per-channel registers (such as the enable bit hardware request line number and link bit field) implemented in flip-flops. However, all remaining per-channel registers are memory-based and, therefore, are not reset.

7.3.2.3 Power Domain

The SDMA controller is part of the CORE power domain.

7.3.3 Hardware Requests

7.3.3.1 Interrupts to the MPU Subsystem

DMA4 has four interrupt lines, numbered Lj with j=0..3. Each logical channel can request an interrupt over any line. The attachment of a channel interrupt event to one of these four external lines is programmable. The software determines whether it attaches a channel interrupt to a single IRQ line or to multiple IRQ lines.

There are two different registers per interrupt line:

- SDMA.DMA4_IRQSTATUS_Lj CH_31_0_Lj field shows the status of the different sources of interrupt. If the SDMA.DMA4_IRQENABLE_Lj bit *i* is 1, the channel *i* is the source of interrupt in line *j*. In contrast to the SDMA.DMA4_CSRi registers, the SDMA.DMA4_IRQSTATUS_Lj registers are updated regardless of the corresponding bits in the SDMA.DMA4_IRQENABLE_Lj registers.
- SDMA.DMA4_IRQENABLE_Lj CH_31_0_Lj_EN field masks/unmasks the channel interrupt. If the SDMA.DMA4_IRQENABLE_Lj bit *i* is set to 0, the channel interrupt *i* of the line *j* is masked.

Each logical channel can generate 10 different interrupt events when enabled (that is, set to 1) in the SDMA.DMA4_CICRi register. Each status bit is updated in the SDMA.DMA4_CSRi register only when the corresponding enable bit is enabled in the SDMA.DMA4_CICRi register.

To determine an interrupt source when an interrupt rises on an interrupt line L*j*, you must:

- Identify the channel (LCH*i*) generating the interrupt.

  Read the SDMA.DMA4_IRQSTATUS_Lj.LCH*i* (LCH0 to LCH31). If LCH*i* = 1, channel *i* is the originator of the interrupt.
- Identify the interrupt event.

  Read the LCH*i* SDMA.DMA4_CSRi. For example, if the drop event (the SDMA.DMA4_CSRi[1] DROP bit) is 1, there will be a request collision.

  The interrupt event status bit in the SDMA.DMA4_CSRi register is immediately reset after it is written to 1.

  The interrupt status bit in the SDMA.DMA4_IRQSTATUS_Lj register is cleared after it is written to 1.

Table 7-2 shows how the SDMA module interrupt lines are connected to the interrupt lines of the MPU interrupt controller.

Table 7-2. SDMA Interrupt Mapping to the MPU Subsystem

IRQ	Source	Description
M_IRQ_12	SDMA_IRQ_0	SDMA interrupt request 0
M_IRQ_13	SDMA_IRQ_1	SDMA interrupt request 1
M_IRQ_14	SDMA_IRQ_2	SDMA interrupt request 2
M_IRQ_15	SDMA_IRQ_3	SDMA interrupt request 3

7.3.3.2 DMA Requests to the SDMA Controller

This section gives information about all modules and features in the high-tier device. See Section 1.4, , to check availability of modules and features. Ensure that DMA requests of unavailable modules and features are masked in DMA subsystems.

All peripherals internal to the device use the transition-sensitive scheme for DMA requests. For more information on the transition-sensitive scheme, see Section 7.2.2, *SDMA Request Scheme*. Table 7-3 lists the SDMA request mapping.

Table 7-3. SDMA Request Mapping

DMA Request Line	Source	Description
S_DMA_0	Reserved	Reserved
S_DMA_1	SYS_DMA_REQ0	External DMA request 0 (system expansion)
S_DMA_2	SYS_DMA_REQ1	External DMA request 1 (system expansion)
S_DMA_3	GPMC_DMA	GPMC request from prefetch engine
S_DMA_4	Reserved	Reserved
S_DMA_5	DSS_LINE_TRIGGER	Display subsystem—frame update request
S_DMA_6	SYS_DMA_REQ2	External DMA request 2 (system expansion)
S_DMA_7	Reserved	Reserved
S_DMA_8	Reserved	Reserved
S_DMA_9	Reserved	Reserved
S_DMA_10	Reserved	Reserved
S_DMA_11	Reserved	Reserved
S_DMA_12	Reserved	Reserved
S_DMA_13	Reserved	Reserved
S_DMA_14	SPI3_DMA_TX0	McSPI module 3—transmit request channel 0
S_DMA_15	SPI3_DMA_RX0	McSPI module 3—receive request channel 0
S_DMA_16	MCBSP3_DMA_TX	MCBSP module 3—transmit request
S_DMA_17	MCBSP3_DMA_RX	MCBSP module 3—receive request
S_DMA_18	MCBSP4_DMA_TX	MCBSP module 4—transmit request
S_DMA_19	MCBSP4_DMA_RX	MCBSP module 4—receive request
S_DMA_20	MCBSP5_DMA_TX	MCBSP module 5—transmit request
S_DMA_21	MCBSP5_DMA_RX	MCBSP module 5—receive request
S_DMA_22	SPI3_DMA_TX1	McSPI module 3—transmit request channel 1
S_DMA_23	SPI3_DMA_RX1	McSPI module 3—receive request channel 1
S_DMA_24	I2C3_DMA_TX	I^2C module 3—transmit request
S_DMA_25	I2C3_DMA_RX	I^2C module 3—receive request
S_DMA_26	I2C1_DMA_TX	I^2C module 1—transmit request
S_DMA_27	I2C1_DMA_RX	I^2C module 1—receive request
S_DMA_28	I2C2_DMA_TX	I^2C module 2—transmit request
S_DMA_29	I2C2_DMA_RX	I^2C module 2—receive request
S_DMA_30	MCBSP1_DMA_TX	MCBSP module 1—transmit request
S_DMA_31	MCBSP1_DMA_RX	MCBSP module 1—receive request

Table 7-3. SDMA Request Mapping (continued)

DMA Request Line	Source	Description
S_DMA_32	MCBSP2_DMA_TX	MCBSP module 2—transmit request
S_DMA_33	MCBSP2_DMA_RX	MCBSP module 2—receive request
S_DMA_34	SPI1_DMA_TX0	McSPI module 1—transmit request channel 0
S_DMA_35	SPI1_DMA_RX0	McSPI module 1—receive request channel 0
S_DMA_36	SPI1_DMA_TX1	McSPI module 1—transmit request channel 1
S_DMA_37	SPI1_DMA_RX1	McSPI module 1—receive request channel 1
S_DMA_38	SPI1_DMA_TX2	McSPI module 1—transmit request channel 2
S_DMA_39	SPI1_DMA_RX2	McSPI module 1—receive request channel 2
S_DMA_40	SPI1_DMA_TX3	McSPI module 1—transmit request channel 3
S_DMA_41	SPI1_DMA_RX3	McSPI module 1—receive request channel 3
S_DMA_42	SPI2_DMA_TX0	McSPI module 2—transmit request channel 0
S_DMA_43	SPI2_DMA_RX0	McSPI module 2—receive request channel 0
S_DMA_44	SPI2_DMA_TX1	McSPI module 2—transmit request channel 1
S_DMA_45	SPI2_DMA_RX1	McSPI module 2—receive request channel 1
S_DMA_46	MMC2_DMA_TX	MMC/SD2 transmit request
S_DMA_47	MMC2_DMA_RX	MMC/SD2 receive request
S_DMA_48	UART1_DMA_TX	UART module 1—transmit request
S_DMA_49	UART1_DMA_RX	UART module 1—receive request
S_DMA_50	UART2_DMA_TX	UART module 2—transmit request
S_DMA_51	UART2_DMA_RX	UART module 2—receive request
S_DMA_52	UART3_DMA_TX	UART module 3—transmit request
S_DMA_53	UART3_DMA_RX	UART module 3—receive request
S_DMA_54	Reserved	Reserved
S_DMA_55	Reserved	Reserved
S_DMA_56	Reserved	Reserved
S_DMA_57	Reserved	Reserved
S_DMA_58	Reserved	Reserved
S_DMA_59	Reserved	Reserved
S_DMA_60	MMC1_DMA_TX	MMC/SD1 transmit request
S_DMA_61	MMC1_DMA_RX	MMC/SD1 receive request
S_DMA_62	Reserved	Reserved
S_DMA_63	SYS_DMA_REQ3	External DMA request 3 (system expansion)
S_DMA_64	Reserved	Reserved
S_DMA_65	Reserved	Reserved
S_DMA_66	Reserved	Reserved
S_DMA_67	Reserved	Reserved
S_DMA_68	Reserved	Reserved
S_DMA_69	SPI4_DMA_TX0	McSPI module 4—transmit request channel 0
S_DMA_70	SPI4_DMA_RX0	McSPI module 4—receive request channel 0
S_DMA_71	DSS_DMA0	Display subsystem DMA request 0 (DSI)
S_DMA_72	DSS_DMA1	Display subsystem DMA request 1 (DSI)
S_DMA_73	DSS_DMA2	Display subsystem DMA request 2 (DSI)
S_DMA_74	DSS_DMA3	Display subsystem DMA request 3 (DSI or RFBI)
S_DMA_75	Reserved	Reserved
S_DMA_76	MMC3_DMA_TX	MMC/SD3 transmit request
S_DMA_77	MMC3_DMA_RX	MMC/SD3 receive request
S_DMA_80		

Table 7-3. SDMA Request Mapping (continued)

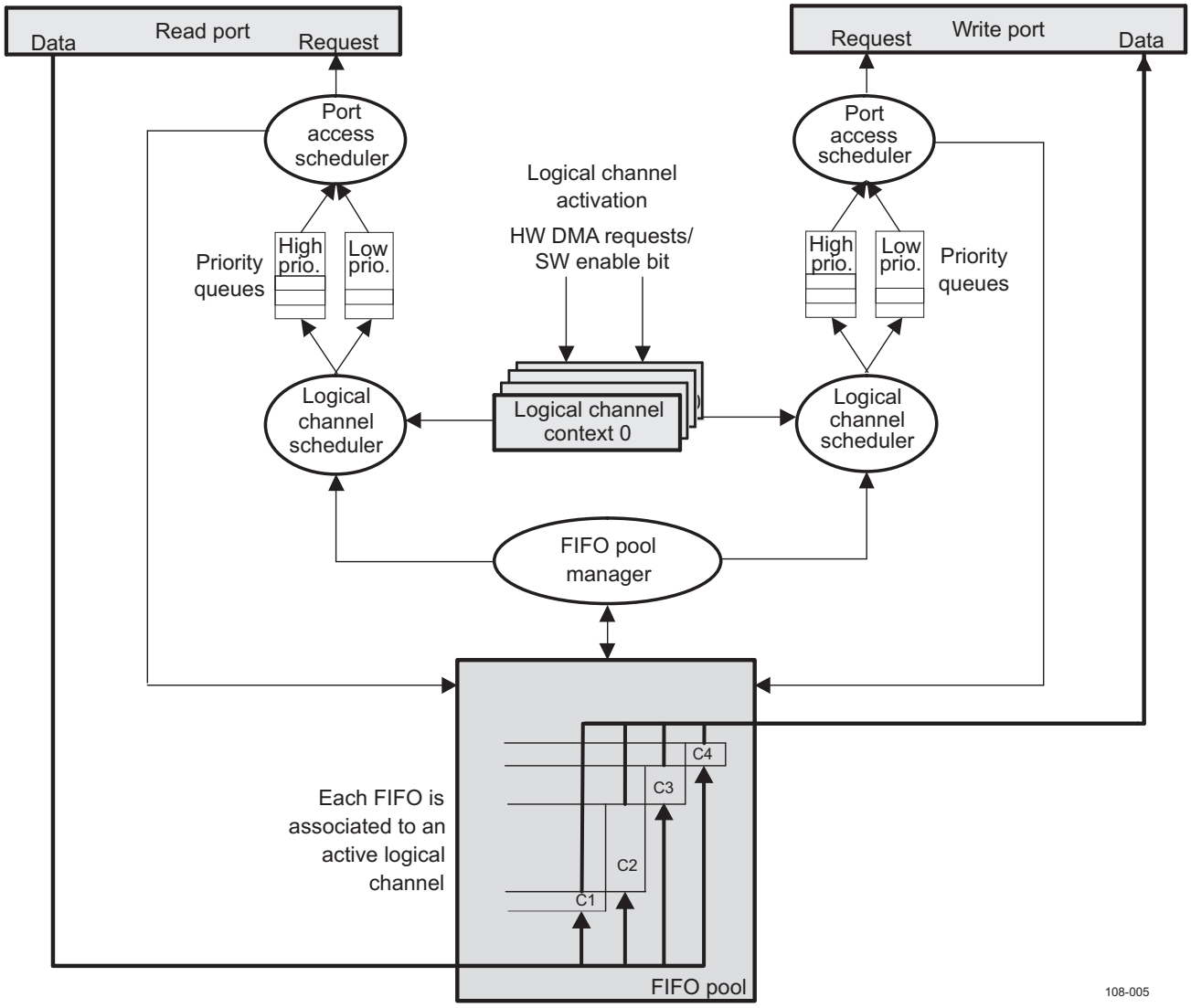
DMA Request Line	Source	Description
... S_DMA_96	Reserved	Reserved

7.4 SDMA Functional Description

The SDMA module provides high-performance data transfers between memories and peripheral devices with low processor use. A DMA transfer is programmed through a logical DMA channel, which allows the transfer to be optimally tailored to the requirements of the application.

Figure 7-6 shows the SDMA controller top-level block diagram.

Figure 7-6. SDMA Controller Top-Level Block Diagram



7.4.1 Logical Channel Transfer Overview

As Figure 7-6 shows, the SDMA module has one read port and one write port operating independently of each other. Buffering is provided between the read and write ports through a FIFO queue memory pool that is shared dynamically between the active logical channels.

- Logical channel synchronization

  A logical channel is described as hardware-synchronized when the DMA transfers are triggered by DMA requests from a hardware device. Alternatively, a logical channel is described as nonsynchronized when the DMA transfer is triggered by software.

- Logical channel activation

  A logical channel becomes active as follows:

  - For hardware-synchronized transfers, when the logical channel is enabled and the hardware DMA request line is asserted
  - For software-triggered (nonsynchronized) transfers, as soon as software enables the logical channel

- Logical channel transfer composition

  A DMA transfer is divided automatically into a number of transactions. Depending on the logical channel context configured, transfer size, the start address alignment, the addressing mode, and the configured maximum burst size, each transaction can be either a single access or a burst of accesses.

- Logical channel scheduling

  When several logical channels are active at the same time, schedulers manage the read and write ports. The scheduling of logical channel transfers is similar for both read and write ports. When a logical channel becomes active, it is added to the tail of a scheduling queue. If more than one logical channel becomes active at the same time, the one with the lower number is queued first. This mechanism provides a first-come, first-serve scheduling scheme between the concurrently active logical channels.

  In addition, each read and write port has a high-priority queue and a low-priority queue. The priority bit in the logical channel SDMA.DMA4_CCRi register determines if a logical channel is queued as high or low priority. The relative weighting of the scheduling of the high-priority queue to the low priority queue is programmable from 1:1 to 1:256 through the DMA global channel register (SDMA.DMA4_GCR).

NOTE: The SDMA.DMA4_GCR[23:16] ARBITRATION_RATE field is not dependent on the SDMA.DMA4_GCR[13:12] HI_THREAD_RESERVED field. The ARBITRATION_ RATE field is dependent on the SDMA.DMA4_CCRi[26] WRITE_PRIORITY bit and the SDMA.DMA4_CCRi[6] READ_PRIORITY bit.

- Read/write port access scheduling policy

  When either the read or write port becomes available, the port access scheduler selects the next logical channel for which to perform a DMA transaction from either the high- or low-priority queue.

  When the current DMA transaction (single or burst access) is complete and the full DMA transfer is not finished, the logical channel returns to the tail of the queue. Because the port access scheduling is on a per-transaction basis, a logical channel can be queued repeatedly this way several times during a single transfer.

The SDMA module can have up to four outstanding read transactions and two outstanding write transactions in the system interconnect; four read and two write thread IDs exist. For an arbitration cycle to occur, these two conditions must be met:

- At least one channel is requesting.
- At least one free thread ID is available.

On an arbitration cycle, the scheduler grants the highest priority channel that has an active request, allocates the thread ID, and tags this thread as busy. At a given time, a channel cannot be allocated for more than one thread ID.

NOTE: If more than one channel is active, each channel is given a thread ID for the current service only, not for the whole channel transfer.

When only one channel is active, one thread ID is allocated during the channel transfer. The access can be up to a 16 x 32-bit access (that is, 16 32-bit single access, four bursts of 16bytes, two bursts of 32bytes, or one burst of 64bytes) without rescheduling the channel at the end of each burst transfer.

When nonburst alignment is at the beginning of the transfer, the channel is rescheduled for each smaller access until burst-aligned. When the end of the transfer is not burst-aligned, the channel is rescheduled for each of the remaining smaller accesses.

For a logical channel transfer completion, when the last access is written to the destination, the logical channel becomes inactive. If enabled, an interrupt request is generated (see Section 7.4.12, *Interrupt Generation*).

7.4.2 FIFO Queue Memory Pool

A FIFO queue memory pool provides buffering between the read and write ports. The hardware allocates the space dynamically to a number of FIFO queues, and each queue is associated with an active logical channel.

To avoid a memory pool overflow, if there are fewer entries in the FIFO queue memory pool than are required for the maximum configured source burst size of the next logical channel to be scheduled, the logical channel is returned to the tail of the queue, and the port access scheduler continues to search the queue until it finds a logical channel that can be scheduled.

The maximum FIFO depth that can be allocated to each individual logical channel can be limited globally through the SDMA.DMA4_GCR register. This value should be configured to allow a fair allocation of the memory pool between the active channels.

A logical channel is scheduled if it has not yet reached its allocation limit, even if the access to be performed will exceed this limit. This means that the effective number of entries used by a particular logical channel is limited to the configured maximum entries per channel + channel maximum configured burst size (in words) -1.

7.4.3 Addressing Modes

A DMA transfer block consists of a number of frames (FN). Each frame consists of a number of elements, and each element can have a size of 8, 16, or 32 bits, as follows:

transfer block size = number of frames x number of elements per frame x element size

The FN, number of elements per frame (EN), and size of elements are common for both the source and destination. However, the way in which the data is represented (addressing profile/mode) is independently programmable for the source and destination devices, using one of these four addressing modes:

- Constant: The address remains the same for consecutive element accesses.
- Post-increment: The address increases by the element size (ES), even across consecutive frames.
- Single-index: The address increases by the ES, plus the element index (EI) value minus one (even across consecutive frames).
- Double-index: The address increases by the ES, plus the EI value minus one within a frame. When a full frame is transferred, the address increases by the ES plus the frame index (FI) value minus 1.

The ES, EI, and FI values are expressed in bytes. The EI and FI values can be positive or negative.

When calculating the EI and FI values, it is critical to note that, after an element is accessed, the logical channel address pointer equals the address of the last byte (highest address) of the accessed element. The correct value for the EI or FI should be such that, when added to the logical channel address pointer, results in the address of the first byte (lowest address) of the next element to be accessed.

The EI and FI values must be configured so that the address of each element in the transfer is aligned on an ES boundary.

Consequently, the single-index addressing mode with EI = 1 or double-index addressing mode with EI = 1 and FI = 1 is equivalent to post-increment addressing.

NOTE: The source and destination start addresses must also be aligned on an ES boundary.

When the address of an element to be accessed is not aligned on an ES boundary, the transfer is stopped and an address error interrupt occurs, if enabled (see Section 7.4.12, *Interrupt Generation*).

The SDMA.DMA4_CFNi register configures the FN in a block.

The SDMA.DMA4_CENi register configures the EN.

The SDMA.DMA4_CSDPi register configures the ES.

The SDMA.DMA4_CSSAi and SDMA.DMA4_CDSAi registers configure the source and destination start addresses.

The SDMA.DMA4_CCRi register configures the source and destination addressing modes.

The SDMA.DMA4_CSEi, SDMA.DMA4_CSFi, SDMA.DMA4_CDEi, and SDMA.DMA4_CDFi registers configure the source EI, source FI, destination EI, and destination FI, respectively.

The addressing profiles are expressed as equations as follows:

Equation 1. Constant addressing:

$$A(n+1) = A(n)$$

Equation 2. Post-increment addressing:

$$A(n+1) = A(n) + ES$$

Equation 3. Single-indexed addressing:

$$A(n+1) = A(n) + ES + (EI - 1)$$

Equation 4. Double-indexed addressing:

When not at the end of a frame or transfer (that is, when the element counter ≠ 0):

$$A(n+1) = A(n) + ES + (EI - 1)$$

When at the end of a frame but not at the end of the transfer (that is, when the element counter = 0 and the frame counter ≠ 0):

$$A(n+1) = A(n) + ES + (FI - 1)$$

Calculate the element and frame index as follows:

Equation 5. Element index

$$EI = [(Stride\ EI - 1) * ES] + 1$$

Equation 6. Frame index

$$FI = [(Stride\ FI - 1) * ES] + 1$$

where:

A(n): Byte address of the element n within the transfer.

ES is in bytes, ESε{1, 2, 4}.

EI is in bytes, specified in a configuration register, –32768 ≤ EI ≤ 32767.

Stride EI: The difference in the number of elements between the start of the current element, *n*, to the start of next element, n+1.

Element counter: A counter that is (re)initiated with the number of elements per frame or per transfer. Decreased by 1 for each element transferred. The initial value is configured in the register DMA channel element number, SDMA.DMA4_CENi.

F is in bytes, specified in a configuration register, –2147483648 ≤ FI ≤ 2147483647.

Stride FI: The difference in the number of elements between the start of the last element of the current frame and the beginning of the first element of the next frame.

Frame counter: A counter that is (re)initiated with the FN per transfer. Decreased by 1 for each frame transferred. The initial value is configured in the register DMA channel frame number, SDMA.DMA4_CFNi.

Figure 7-7 shows how a stride EI and FI are defined. When handling complex configurations, using strides can make it easier to calculate EI and FI because you can calculate in elements instead of bytes. (This approach is used in the 90Degrees clockwise image rotation example shown in Figure 7-11.) The double-index addressing example shown in Figure 7-7 has ES = 4, EN = 2, EI = 5, FI = 5, and FN = 2.

Figure 7-7 through Figure 7-10 show examples of addressing mode configurations. Table 7-4 lists parameter values for the examples.

Figure 7-7. Example Showing Double-Index Addressing, Elements, Frames, and Strides

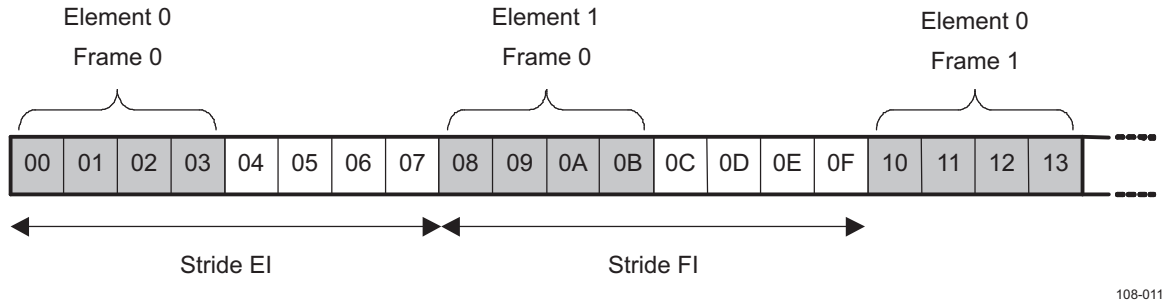


Figure 7-8. Addressing Mode Example (a)

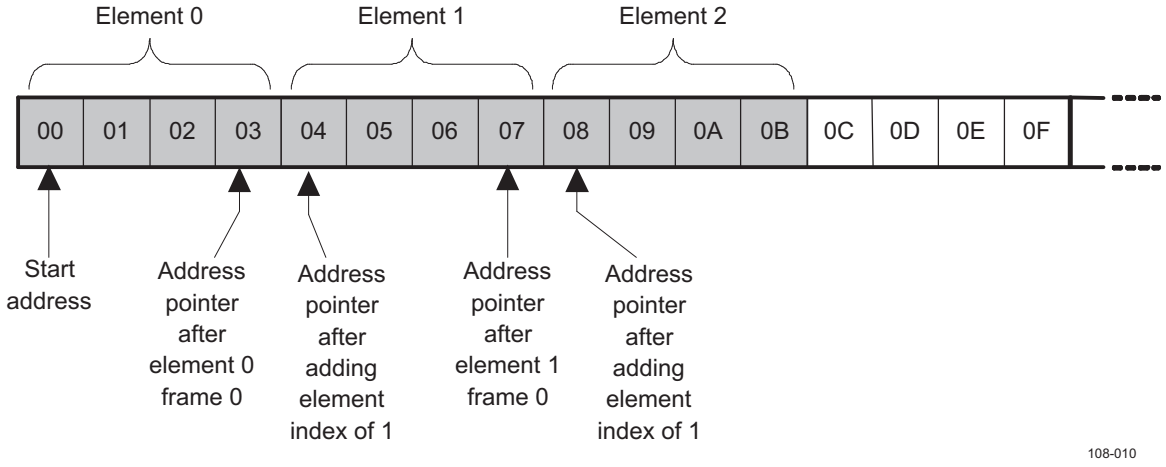


Figure 7-9. Addressing Mode Example (b)

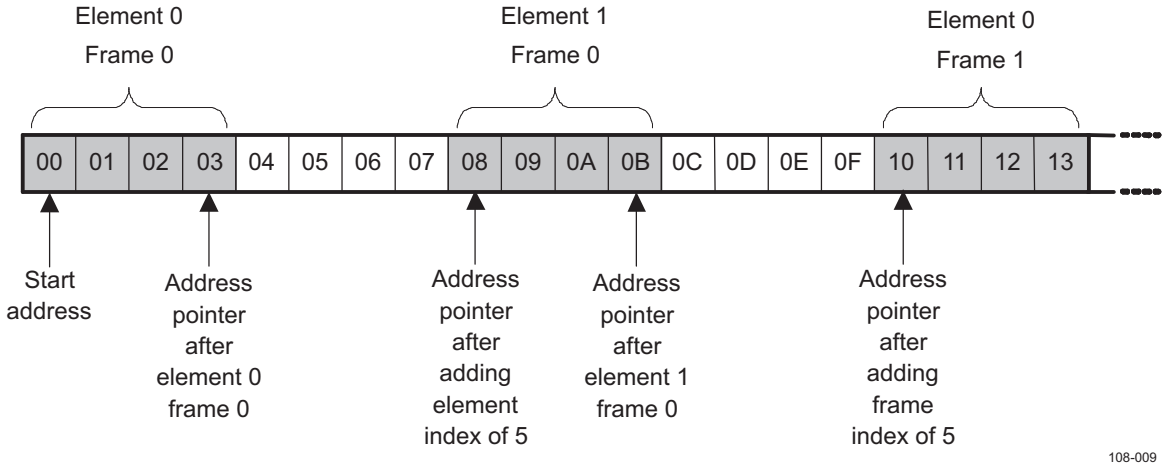


Figure 7-10. Addressing Mode Example (c)

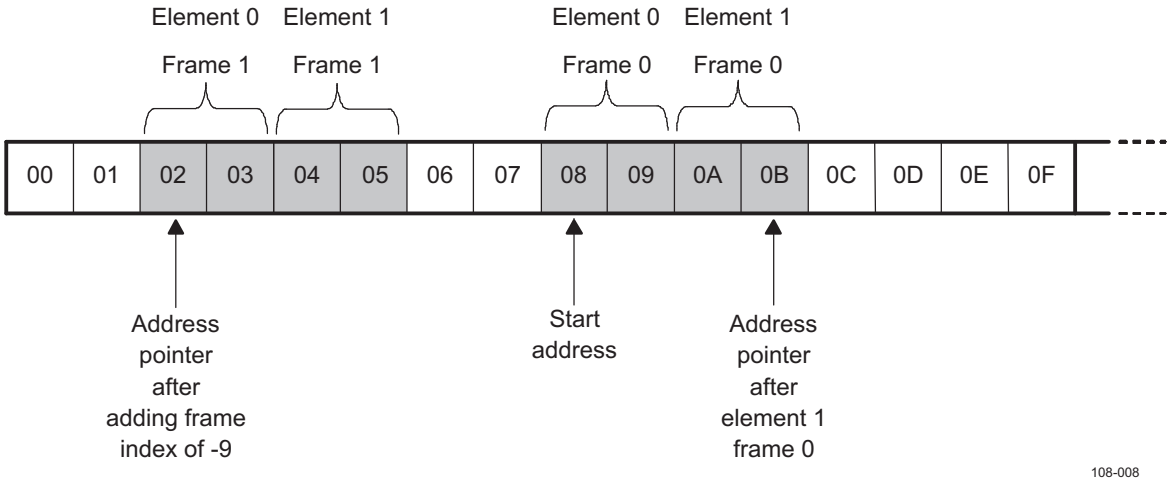


Table 7-4. Parameter Values for Addressing Mode Examples (a), (b), and (c)

Parameter	Example (a)	Example (b)	Example (c)
Addressing mode	Single index (or post-increment)	Double index	Double index
Start address	0	0	8
ES	4 (32-bit)	4 (32-bit)	2 (16-bit)
EN	3	2	2
EI	1	5	1
FN	1	2	2
Frame index	N/A	5	-9

Double indexing can occur either on source (read) or destination (write). Equations for rotation of xx degrees on destination are obtained by taking equations for rotation of (360-xx) degrees on source, and swapping x and y in them. The opposite is also true. Table 7-5 list the equations for rotation for 90, 180 and 270°.

Table 7-5. Equations for Rotation

		90° Rotation	180° Rotation	270° Rotation
Double indexing on destination (write)	Base address	ES*(y-1)	ES*(x*y-1)	ES*y*(x-1)
	Element index (EI)	ES*(y-1)+1	1-2*ES	1-ES*(y+1)
	Frame index (FI)	1 – ES*[(x-1)*y+2]	1-2*ES	1+ES*(x-1)*y
Double indexing on source (read)	Base address	ES*x*(y-1)	ES*(x*y-1)	ES*(x-1)
	Element index (EI)	1-ES*(x+1)	1-2*ES	ES*(x-1)+1
	Frame index (FI)	1+ES*(y-1)*x	1-2*ES	1 – ES*[(y-1)*x+2]

Table 7-6 and Figure 7-11 show the configuration required to perform a 90° clockwise image rotation of a 240 x 160 pixel, 32-bit image. The EI, frame size, and FI values are configured so that the image is rotated line by line starting at the left-hand end of the top line.

NOTE: The FI value for the destination is negative so that the first pixel of each subsequent line of the source image is written to the correct location at the destination.

Equation 5 and Equation 6 calculate the destination FI and EI. The example assumes that the image lines are stored at consecutive addresses in memory, meaning that both EI and FI on the source side are 1.

Rotations:

Section 7.5.7, *90° Clockwise Image Rotation*, describes how to program this example.

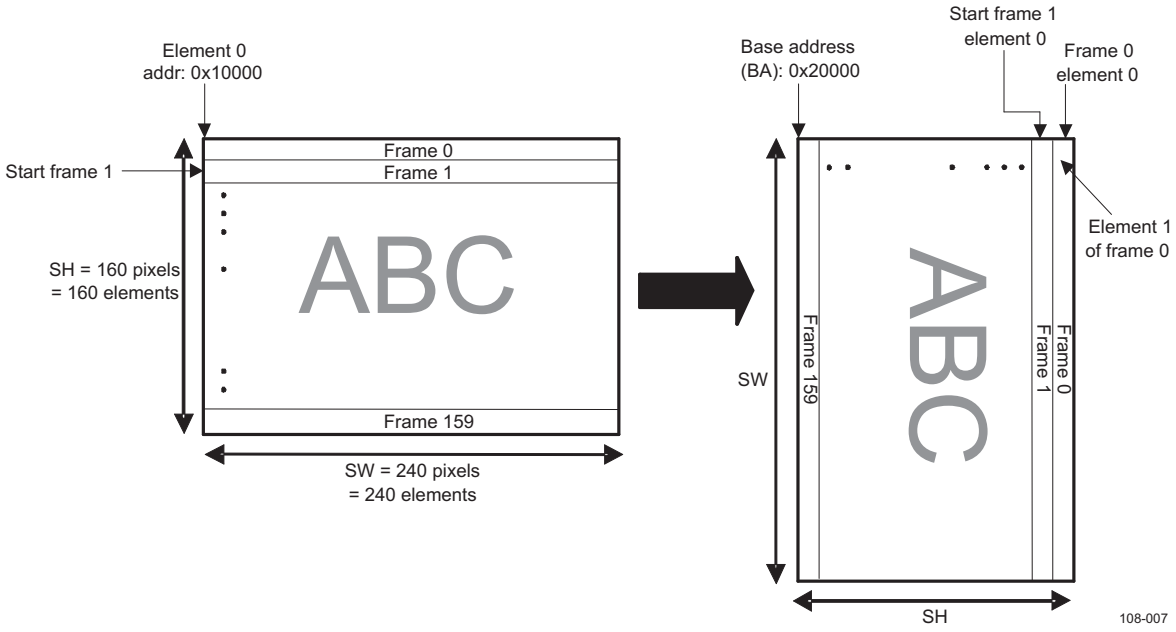
Observe that:

- One pixel = one element
- One line = one DMA frame
- Pixel size = element size = ES

Table 7-6. Example Parameter Values for a 90° Clockwise Image Rotation

Parameter	Source Value	Destination Value
Bits per pixel	32	32
ES	4	4
Image width	SW	SH
Image height	SH	SW
Stride elements (stride EI)	1 element	SH
Stride frames (stride FI)	1 element	-[(SW-1)*SH+1] = –38241 elements
Start address	0x100000	0x200000 + (SH – 1) x ES = 0x20027C
EN	SW	SW
EI	[(Stride EI – 1) * ES] + 1 = 1	[(Stride EI – 1) * ES] + 1 = 637
FN	SH	SH
FI	[(Stride FI – 1) * ES] + 1= 1	[(Stride FI – 1) * ES] + 1 = –152967

Figure 7-11. Example of a 90° Clockwise Image Rotation



7.4.4 Packed Accesses

When the logical channel ES is less than the DMA module read/write port size, and the addressing profile supports it (that is, post-increment mode or single- or double-index mode with EI = 1), the number of elements to transfer in each read/write port access may be maximized by specifying that the source or destination is packed through the channel SDMA.DMA4_CSDPi register. Thus:

- For a read/write port size of 32 bits, the source or destination can be configured as packed for transfer ESs of 8 bits (four elements per access) and 16 bits (two elements per access).
- For a read/write port size of 64 bits, the source or destination can be configured as packed for transfer ESs of 8 bits (eight elements per access), 16 bits (four elements per access), and 32 bits (two elements per access).

Depending on the start address and transfer length, the first or last packed access might be only partially filled. This is indicated to the source or destination using the byte-enable signals.

NOTE: When a constant addressing mode is specified, only nonpacked accesses are used, and specifying the accesses as packed has no effect.

7.4.5 Burst Transactions

Transfer performance can be improved, where the source or destination and addressing profile supports it, by configuring the logical channel to perform burst transactions consisting of multiple instead of single accesses. The channel can be programmed to use burst sizes equivalent to 16, 32, or 64 bytes through the SDMA.DMA4_CSDPi register, with the read burst size being programmable independently of the write burst size. Typically, the optimal burst size is 64 bytes (16 accesses for a 32-bit read/write port size or 8 accesses for a 64-bit read/write port size).

To obtain the maximum benefit from burst transactions, the source and destination start addresses should be aligned with the burst size. If this is not the case, the start of the transfer can consist of a number of smaller (single or burst) transactions until the first burst size boundary is reached.

Similarly, if the end of the transfer is not aligned on a burst size boundary, the final part of the transfer can consist of a number of smaller transactions.

NOTE: Except in the constant addressing mode, the source or destination must be specified as packed for burst transactions to occur.

7.4.6 Endianism Conversion

The source and destination are each specified as little-endian or big-endian through the SDMA.DMA4_CSDPi register for the particular logical channel. If the endianism of the source and destination differ, and the logical channel ES is less than the SDMA module read/write port size, an endianism conversion is applied to the data before it is written to the destination.

When transferring data between a source and a destination with different endianism, it is important to specify an ES that is equal to the type of data being transferred to preserve the correct data image at the destination.

In the system, endianism conversion can be performed in more than one place. It is possible to inform the source and/or destination to lock the endianism (that is, to not perform a conversion) through the logical DMA channel SDMA.DMA4_CSDPi register.

7.4.7 Transfer Synchronization

A logical channel can be programmed for either software-triggered or hardware synchronized transfers.

7.4.7.1 Software Synchronization

A transfer is software-triggered when the logical channel is set up and started by software. To specify a software-triggered transfer, set the channel DMA register bits SDMA.DMA4_CCRi[4:0] and SDMA.DMA4_CCRi[20:19] to 0. The transfer starts as soon as the DMA register bit SDMA.DMA4_CCRi[7] is set (that is, enters the scheduling process).

7.4.7.2 Hardware Synchronization

A transfer is hardware-synchronized if the logical channel activation is driven by hardware requests from either the source or destination target. A hardware synchronized transfer is specified by configuring the DMA request line number in the channel SDMA.DMA4_CCRi register to a value that corresponds to the DMA request line from the source or destination that generates the DMA requests. The DMA request numbers to be configured are specified in the DMA request mapping (see Table 7-8).

Specify the DMA request number in the channel DMA register bits SDMA.DMA4_CCRi[4:0] and SDMA.DMA4_CCRi[20:19]. After the DMA register bit SDMA.DMA4_CCRi[7] is set, the logical channel becomes enabled but not activated (that is, it does not enter the scheduling process), which means that channel registers are not updated until the first DMA request is received.

NOTE: DMA Request Line

A DMA request line must not be shared between concurrently enabled DMA channels. However, a DMA request line can be shared between several chained logical channels.

The channel synchronization control registers are 1-based. For example, to enable the S_DMA_1 request, SDMA.DMA4_CCRi[4:0] SYNCHRO_CONTROL must be set to 0x2 (DMA request number + 1).

For hardware synchronization, the amount of data to be transferred for each assertion of the DMA request line is configured through the frame synchronization (FS) and block synchronization (BS) bits in the logical channel SDMA.DMA4_CCRi register and the DMA register bits SDMA.DMA4_CCRi[5] and SDMA.DMA4_CCRi[18], respectively.

The amount of data can be any of the following:

- A single element transfer: A complete element defined by Data_type. For example, 8/16/32 bits are transferred in response to a DMA request.
- A full frame: A complete frame of several elements is transferred in response to a DMA request.
- A full block (a full channel transfer): A complete block of several frames is transferred in response to a DMA request.
- A full packet (a full channel transfer): A complete packet of several elements is transferred in response to a DMA request.

Packets allow the size of each part of the full DMA transfer to be configured independent of the organization of the data to be transferred (typically a number of elements). This can be useful when the source or destination has a buffer (such as a FIFO queue) with a size unrelated to the frame size of the transfer. The packet size then can be set to the size of the buffer.

Packet transfer must be used only where the source or destination is addressed in constant addressing mode, because FI registers are reused to specify size of the packet.

To support the burst mode, the logical channel must also be configured to use the source-port packed access mode. The packed address mode cannot be used with a constant address mode: it must be configured to use a post-increment address mode.

The packet size is configured based on the source/destination synchronization select bit in the SDMA.DMA4_CCRi register through either the channel SDMA.DMA4_CDFi register (source synchronized) or the SDMA.DMA4_CSFi register (destination synchronized).

When the logical channel transfer block is not an exact multiple of the packet size, the final packet consists of the remaining elements in the transfer, using burst or single accesses to complete the block transfer.

The maximum transfer size, regardless of the packet size, is always as follows:

Block_size = Number_of_Frame_in_Block * Number_of_Element_in_Frame * Element_Size

- Synchronized at the source

  The DMA module optimizes the transfer with respect to the number and size of burst transactions for the given source and destination addressing profiles and configured maximum burst sizes. When writing to the destination is slower than reading from the source, data is buffered in the channel FIFO queue. If the transfer is packet-synchronized at the source, the end-of-packet interrupt is disabled (see Section 7.4.11, *Reprogramming an Active Channel*).

  For a source synchronized transfer, buffering can be enabled or disabled by setting the SDMA.DMA4_CCRi[25] BUFFERING_DISABLE bit. For a packet source synchronization with buffering disabled and the packed/burst across the packet boundary, the last packed/burst write transaction is split in optimized smaller accesses to complete the packet transfer size. However, for a packet source synchronized transfer with buffering enabled and with the packed/burst across the packet boundary,

the DMA module waits for the next DMA request(s) to read enough data to issue an atomic packed/burst write transaction (assuming the address is packed/burst aligned).

NOTE: Regardless of whether buffering is enabled or not, buffering is not performed between frames. If the packed/burst is across the frame boundary, the last packed/burst write transaction is split into optimized smaller accesses to complete the frame transfer size.

- Synchronized at the destination

  The performance of a hardware-synchronized transfer can be improved by using the prefetch mode, enabled through the channel DMA register bit SDMA.DMA4_CCRi[23]. Data is prefetched on the read port side in advance of the DMA request received and buffered in the FIFO queue. Up to a full transfer block can be prefetched, although this can be limited by the specified maximum channel FIFO queue depth (see Section 7.4.2, *FIFO Queue Memory Pool*).

  Buffering disable is not allowed for a destination-synchronized transfer.

NOTE: Behavior is undefined when prefetch is enabled and a transfer is synchronized to the source.

Whether buffering is enabled or disabled, the last transaction in the frame or in the block is write nonposted (WNP) even if the write mode is specified as write last nonposted (WLNP; the WRITE_MODE bit field of the SDMA.DMA4_CSDPi register = 0x2). However, in a packet synchronization mode, the last transaction of each packet in the transfer is WNP only if buffering disable is on (even if the write mode is specified as WLNP).

Regardless of whether buffering disable is enabled or disabled, the packet interrupt is not generated in the packet source synchronized mode.

CAUTION

The BUFFERING_DISABLE bit field of the SDMA.DMA4_CCRi register must be filled with an allowed value, as specified in Table 7-7.

Table 7-7. Buffering Disable

	BUFFERING_DISABLE (0: buffering enable, 1: buffering disable)	
Destination synchronized	0	Allowed
	1	Not allowed
Source synchronized	0	Allowed
	1	Allowed

Synchronized transfer monitoring using CDAC (SDMA.DMA4_CDACi):

Context is restored only when the channel becomes active on a DMA request (not at software enable). The channel is software-enabled first, and then a DMA request is asserted followed by the first context restore.

The CDAC register is writable; thus, you can initialize the CDAC to monitor the transfer and determine if the transfer is started or not (see Section 7.5.4, *Synchronized Transfer Monitoring Using CDAC*, for more information).

NOTE: The CDAC register must be written or read so that the least-significant byte (LSByte) is read first; otherwise, the shadow registers do not update the CDAC registers.

This is not an issue for 32-bit read-write transactions. Nevertheless, for 16-bit transactions, start reading or writing the LSByte first to enable the register update.

7.4.8 Thread Budget Allocation

When several concurrent channels are latency critical and hardware synchronized, a specific latency cannot be ensured until the target is served. This situation occurs when the concurrent channel number is superior to the number of available threads.

NOTE: Four threads are available on the read port, and two threads are available on the write port.

For a hardware-synchronized transfer (memory to peripheral), a minimum bandwidth for a latency-critical transfer must be ensured to avoid collisions between two hardware requests.

Because it is latency critical, the software user is responsible for the following:

- Programming the synchronized channel as a high-priority channel
- Reserving one or several threads for high-priority channels

The proposed implementation is as follows (see Section 7.5.5):

Prevent the regular channel queue from exceeding more than a programmable (3, 2, or 1) number of threads on the read port and no more than one thread on the write port. This number can be set on the global register SDMA.DMA4_GCR[13:12].

The thread reservation is programmable for maximum use of thread resources for concurrent, low-priority channel transfer. Programmability can also allow a partial throughput control by limiting in software the number of concurrent outstanding requests that break the pipelining.

Depending on the SDMA.DMA4_GCR [13:12] value, the following threadID on the read/write ports are allocated for a high-priority channel:

Read port priority thread reservation:

- SDMA.DMA4_GCR[13:12] = 0x0 => No ThreadID is allocated for high-priority channels.
- SDMA.DMA4_GCR[13:12] = 0x1 => Read ThreadID 0 is allocated for high-priority channels.
- SDMA.DMA4_GCR[13:12] = 0x2 => Read ThreadID 0 and Read ThreadID 1 are allocated for high-priority channels.
- SDMA.DMA4_GCR[13:12] = 0x3 => Read ThreadID 0, Read ThreadID 1, and Read ThreadID 2 are allocated for high-priority channels.

Write port priority thread reservation:

- SDMA.DMA4_GCR[13:12] = 0x0 => No ThreadID is allocated for high-priority channels
- SDMA.DMA4_GCR[13:12] = 0x1 => Write ThreadID 0 is allocated for high-priority channels.
- SDMA.DMA4_GCR[13:12] = 0x2 => Write ThreadID 0 is allocated for high-priority channels.
- SDMA.DMA4_GCR[13:12] = 0x3 => Write ThreadID 0 is allocated for high-priority channels.

Regardless whether or not the enabled channels are of high priority, only the setting of the SDMA.DMA4_GCR[13:12] value forces the thread reservation to these values. Set the appropriate value to avoid losing threads using only regular channels.

To have an independent read and write priority context, a per-channel bit SDMA.DMA4_CCRi[26] is added for write priority, and the previous priority bit becomes read priority bit SDMA.DMA4_CCRi[6].

NOTE: The device has one priority bit per logical channel, not one per port.

7.4.9 FIFO Budget Allocation

To avoid fully occupying the FIFO with a high-priority transfer while low-priority channels wait in the arbitration queue, two separate FIFO budgets are specified: one for high-priority channels and one for low-priority channels. This is defined in the SDMA.DMA4_GCR register, allowing the user to share the FIFO budget between the low- and high-priority channels. The amount of the FIFO allocated by the low- and high-priority channels is fixed by the value set in the SDMA.DMA4_GCR[15:14] HI_LO_FIFO_BUDGET field. The maximum channel FIFO depth is limited by the HI_LO_FIFO_BUDGET field as follows:

If the channel is low priority:

- When HI_LO_FIFO_BUDGET = 0x1, then low priority cannot exceed 75 percent of the total FIFO.
- When HI_LO_FIFO_BUDGET = 0x2, then low priority cannot exceed 25 percent of the total FIFO.
- When HI_LO_FIFO_BUDGET = 0x3, then low priority cannot exceed 50 percent of the total FIFO.

If channel is high priority

- When HI_LO_FIFO_BUDGET = 0x1, then high priority cannot exceed 25 percent of the total FIFO.
- When HI_LO_FIFO_BUDGET = 0x2, then high priority cannot exceed 75 percent of the total FIFO.
- When HI_LO_FIFO_BUDGET = 0x3, then high priority cannot exceed 50 percent of the total FIFO.

The user is responsible for performing the following equation:

- For a high-priority channel: (Per_Channel_Maximum FIFO Depth + 1) x Number of High Channel =< High Budget FIFO
- For a low-priority channel: (Per_Channel_Maximum FIFO Depth + 1) x Number of Low Channel =< Low Budget FIFO

NOTE: Ensure that *Number of High Channel* means *Number of Active High-Priority Channel* and that *Number of Low Channel* means *Number of Active Low-Priority Channel*.

7.4.10 Chained Logical Channel Transfers

Chaining multiple logical channels permits transfers consisting of multiple parts to be executed without repeated software intervention. This results in better performance than the alternative of software setting up and starting each transfer separately. Each part of a chained transfer can have the data addressed in a different manner that permits the programming of a variety of complex transfers. For example:

- Interlaced video data with one logical channel configured to transfer the even lines and another logical channel configured to transfer the odd lines
- Protocol headers with a separate DMA4 channel configured to transfer each field in the header

Channels can be chained through each channel SDMA.DMA4_CLNK_CTRLi register. When the transfer for the first channel completes, the next channel in the chain is enabled. The number of channels in the chain that are configured for hardware-synchronized transfers is flexible (although typically it might be all, none, or just the first one). The DMA request line number should be set to 0 to specify that any or all of the channels in a chain are software-triggered or nonsynchronized.

The last channel in a chain can be chained to the first channel to create a continuously looping chain. The continuously looping transfer can be stopped on the fly at a specific channel by disabling the SDMA.DMA4_CLNK_CTRLi[15] ENABLE_LNK bit. The looping transfer stops after the specified channel transfer is complete.

NOTE: DMA Request Line

A DMA request line must not be shared between concurrently enabled DMA channels. However, a DMA request line can be shared between several chained logical channels.

For more information on the programming model, see Section 7.5, *SDMA Basic Programming Model*.

7.4.11 Reprogramming an Active Channel

A currently active logical DMA channel can be disabled through the SDMA.DMA4_CCRi[7] ENABLE bit. When any ongoing transaction is complete and the read-active and write-active bits in the SDMA.DMA4_CCRi register (SDMA.DMA4_CCRi[9] RD_ACTIVE and SDMA.DMA4_CCRi[10] WR_ACTIVE) are reset, the channel can be reprogrammed for a new transfer.

7.4.12 Interrupt Generation

The SDMA module has four interrupt request output lines, SDMA_IRQ_0 to SDMA_IRQ_3. One or more logical channels can be programmed to generate an interrupt request on any of these lines when any one of the maskable DMA events listed in Table 7-8 occurs.

Table 7-8. Logical DMA Channel Events

Event	Description
End of packet	A packet transfer completed.
End of block	A block transfer completed.
End of frame	A frame transfer completed.
Half of frame	Half of the current frame transferred.
Start of last frame	The first element of the last frame transferred.
Transaction error	A transaction error returned by the interconnect in either the read or write port.
Address error	An attempt was made to perform a DMA access to an address not aligned on an ES boundary.
Supervisor transaction error	An error occurred, for example, when an unauthorized initiator (that is not a supervisor) tries to use a supervisor transfer.
Synchronization error	A new DMA request arrived before completion of the transfer because of the previous DMA request.
Drain end	Drain is completed (SDMA.DMA4_CCRi[10] WR_ACTIVE becomes 0).
System transaction error	An error has occurred, for example, when an unauthorized initiator tries to set a normal channel to a System channel or tries to change a System channel to a normal channel or tries to access (write) any register of a System channel registers.
Drop error	A drop event interrupt is generated when a DMA request is being serviced while a second one is asserted and a third one arrives before the second DMA request is serviced.

The logical DMA channels that generate an interrupt on a particular IRQ output are specified through the SDMA.DMA4_IRQENABLE_Lj register (where *j* is the IRQ number: 0, 1, 2, or 3). The events that generate an interrupt for a particular channel can be configured through the channel SDMA.DMA4_CICRi register.

When an interrupt is detected, the logical DMA channel generating the event can first be identified by reading the SDMA.DMA4_IRQSTATUS_Lj register. The event causing the interrupt then can be identified by reading the interrupt status via the relevant DMA channel SDMA.DMA4_CSRi register.

7.4.13 Packet Synchronization

A packet transfer notion is related to the behavior of some peripheral, which have certain buffering capability and requires to transfer the buffer content once an element number threshold is reached (a hardware DMA request is generated). To associate a frame synchronization to each DMA request is possible, but this limits the maximum transfer size. Indeed the maximum transfer size is proportional to the FIFO depth of the peripheral:
maximum_transfer_size =peripheral_FIFO_depth x number_of_frame_in_block.
The packet synchronization allows to dissociate the transfer size from the FIFO depth of the peripheral. Only Constant addressing mode is allowed on RD port or WR port if source target or destination target is packet synchronized respectively.

Example:
Let's consider a camera interface, which have a FIFO_depth of 128 Words and a FIFO_element_number_threshold of 128 and a picture to transfer with a size 320 lines per 240 columns. If frame synchronization is associated to each DMA request then the maximum transfer size that can be performed is $128x2^{16}$ words. In this case, a frame is 128 words long, which does not fit the size of a line. Then it's not possible to generate an interrupt at the end of line. However with introducing the packet transfer notion, which is related to the peripheral FIFO behavior/structure, the maximum transfer size (maximum_transfer_size = 2^{24} x 2^{16} words) is independent of both peripheral_FIFO_depth and FIFO_element_number_threshold. This allows, making an enough long transfer within one channel context and perform rotation operation on big image format.

The main features of DMA Packet transfer are as follows:

- DMA Packet_Data_Size for each DMA Request: typically this will be

Peripheral_element_number_threshold Number of elements in a packet shares the SDMA.DMA4_CSFIi and SDMA.DMA4_CDFIi configuration registers. Indeed if the peripheral is the source target, respectively destination target, that means the used addressing mode is constant, consequently SDMA.DMA4_CSFIi[15:0], respectively SDMA.DMA4_CDFI[15:0], is used to specify the packet data size (PKT_ELNT_NBR) and the bit fields [31:16] are unused. To specify the Packet data size in the SDMA.DMA4_CSFIi or SDMA.DMA4_CDFIi, the user must set the SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC respectively to 1 or 0.

NOTE: The packet size can be a sub-multiple or non sub-multiple of a frame size. If DMA Packet_Data_Size is aligned on DMA channel block data size boundary, DMA will transfer the last data in channel block boundary and stop at block boundary for the last packet DMA request. If the Packet_Data_size is not aligned on the block boundary, the remaining data smaller than a packet size are transferred using burst or single accesses to complete the block.

- DMA Packet_Data_Transfer does not affect DMA channel capabilities in term of packing and bursting.

The Packet synchronization mode is active when SDMA.DMA4_CCRi[5] FS = SDMA.DMA4_CCRi[18] BS = 1. Then

- if SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC=0; SDMA.DMA4_CDFIi[15:0] gives the number of element in packet and SDMA.DMA4_CDFIi[31:16] is unused for the packet size.
- if SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC=1; SDMA.DMA4_CSFIi[15:0] gives the number of element in packet and SDMA.DMA4_CSFIi[31:16] is unused for the packet size.

NOTE: The maximum transfer size, regardless of the packet size, is always: Block_size = Number_of_Frame_in_Block x Number_of_Element_in_Frame x Element_Size. If the DMA channel packet/burst access is across packet boundary, DMA hardware automatically splits this packing/burst access into multiple smaller accesses, which will be aligned on packet boundary. Otherwise, the DMA transfers data as usual packing/burst access.

7.4.14 Graphics Acceleration Support

The SDMA supports two graphic acceleration features:

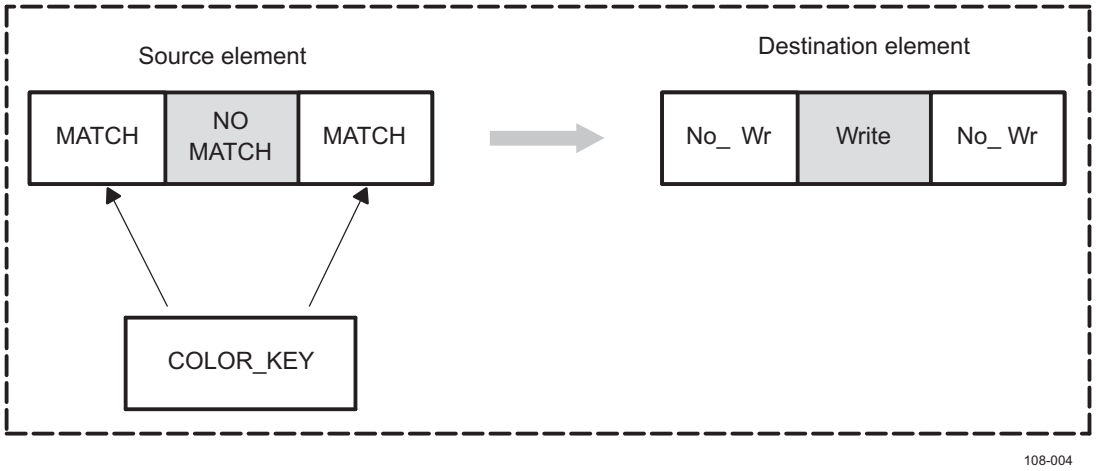
- Transparent copy
- Constant fill

Only one of these features can be enabled at any given time through the SDMA.DMA4_CCRi register for the particular logical DMA channel.

The transparent copy feature enables specification of a particular color through the SDMA.DMA4_COLORi register, so that when it is recognized in the data from the source, it is not copied to the corresponding location in the destination, but instead leaves the data in the corresponding location in the destination as it is.

Figure 7-12 shows the 2-D graphic transparent color block diagram.

Figure 7-12. 2-D Graphic Transparent Color Block Diagram



The constant fill feature provides the ability to specify a particular color through the SDMA.DMA4_COLORi register for every specified location in the destination. In this case, the transfer consists only of writing to the destination without reading from a source.

Both features support 8 bpp (bits per pixel), 16 bpp, and 24 bpp, depending on what is specified as the DMA transfer ES via the SDMA.DMA4_CSDPi register. An ES of 32 bits corresponds to 24 bpp. During a 32-bit (24 bpp) transfer, the most-significant 8 bits ([31:24]) are 0. Both features are compatible with packed and burst transactions.

7.4.15 Supervisor Modes

A logical DMA channel can be configured to operate in supervisor mode through the corresponding bits in the channel SDMA.DMA4_CCRi register. This must be done using supervisor access. Once a channel is configured in supervisor mode, the channel configuration is protected from nonsupervisor accesses. All DMA transactions on a supervisor channel are supervisor transactions.

7.4.16 Posted and Nonposted Writes

A logical channel can be configured in its DMA register bits SDMA.DMA4_CSDPi[17:16] to use one of three write access handshake modes for the destination:

- Nonposted write: Each write must complete before transfer can continue or complete.
- Posted write: Transfer continues without waiting for each write to complete (might improve performance with slow devices).
- Posted with final write nonposted: Transfer continues without waiting for each write to complete, but final write is completed before transfer can complete.

7.4.17 Disabling a Channel During Transfer

When a channel is disabled during a transfer, the channel will undergo an abort, except if the channel was hardware source synchronized with buffering Enabled (SDMA.DMA4_CCRi[25] BUFFERING_DISABLE='0'). In that case, the fifo will be drained in order to avoid losing data. See Section 7.4.18 for details on this feature.

7.4.18 FIFO Draining Mechanism

When a source synchronized channel is disabled during a transfer, then the current hardware request (element/packet/frame/block) service is completed and the channel SDMA.DMA4_CCRi[9] RD_ACTIVE bit is set to 0, which means the channel is not active on the read port. The remaining data in the corresponding disabled channel FIFO is drained onto the write port and transferred to the programmed destination as in normal transfer.

At the end of the draining the SDMA.DMA4_CCRi[10] WR_ACTIVE bit is set to 0 (channel is no more active on the write port) and if the SDMA.DMA4_CICRi[12] DRAIN_END_IE is set to 1, the status bit DMA4_CSRi[12] DRAIN_END is updated and an interrupt is generated.

Once a channel is disabled during a transfer, it needs to wait for SDMA.DMA4_CCRi[9] RD_ACTIVE and SDMA.DMA4_CCRi[10] WR_ACTIVE to become '0' before being re-enabled for a new transfer. The FIFO drain for a channel will happen only in the following cases:

- If the channel is a source synchronized channel and SDMA.DMA4_CCRi[25] BUFFERING_DISABLE='0' and
- If the channel is not a solid fill channel and
- If the channel is not a transparent and copy channel

NOTE: In case of a self-linked or chain-linked channel, it is user responsibility to disable the SDMA.DMA4_CLNK_CTRLi[15] ENABLE_LINK bit before disabling the channel.

In all other cases, the channel will undergo an abort.

NOTE: If the channel intended to be used is a drain candidate, always enable DRAIN END INTERRUPT for that channel.

When the sDMA is in smart standby mode, before disabling a drain candidate channel (DMA4_CCRi[7] ENABLE = 0x0), the sDMA must be configured to be in force standby mode (DMA4_OCP_SYSCONFIG[13:12] = 0x0) or in no standby mode (DMA4_OCP_SYSCONFIG[13:12] = 0x1). After the occurrence of DRAIN END INTERRUPT EVENT from that channel, the sDMA can be switched back to smart standby mode (DMA4_OCP_SYSCONFIG[13:12] = 0x2).

7.4.19 Reset

Following a software or hardware reset, all fields in the logical channel registers have undefined values, except 5 bits in the SDMA.DMA4_CCRi and SDMA.DMA4_CICRi registers. Thus, when programming a channel for the first time, the remaining fields in all channel registers must be configured before enabling the channel.

NOTE: After a reset, the global registers take their specified reset values.

7.4.20 Power Management

The SDMA module provides two methods to reduce power consumption:

- Interconnect clock auto-idle
- Automatic standby mode

7.4.20.1 Interconnect Clock Auto-Idle

The interconnect clock auto-idle power-saving mode is enabled or disabled in DMA register bit SDMA.DMA4_OCP_SYSCONFIG[0]. When this mode is enabled and there is no activity on the interconnect interface, the interconnect clock is disabled internally to the module to reduce power consumption. When there is new activity on the interconnect interface, the interconnect clock is restarted without any latency penalty. After reset, this mode is disabled by default. Enabling this mode is recommended to reduce power consumption.

7.4.20.2 Automatic Standby Mode

As part of the system-wide power-management scheme, the module can go into a standby mode at the request of the power, reset, and clock management (PRCM) module (for more information, see , *Power, Reset, and Clock Management*).

The module can be configured to one of the following standby modes using the DMA register bits SDMA.DMA4_OCP_SYSCONFIG[13:12]:

- No standby mode: The module never goes into standby mode.
- Force standby mode: The module goes into standby mode only when all the DMA channels are disabled.
- Smart standby mode: The module enters standby mode when:
  - All DMA channels are disabled.
  - No nonsynchronized channel is enabled, no DMA request line is asserted, and no DMA request is pending in the module.

7.5 SDMA Basic Programming Model

7.5.1 Setup Configuration

After a software or hardware reset, program all fields in the logical channel registers to default values for any channels used, because most fields are undefined following reset.

Before programming any DMA transfers, the priority arbitration rate and the maximum FIFO depth must be configured through the SDMA.DMA4_GCR register, and any required interrupts must be enabled through the SDMA.DMA4_IRQENABLE_Lj registers and the logical channel SDMA.DMA4_CICRi registers.

Software clears the SDMA.DMA4_CSRi register and the IRQSTATUS bit for the different interrupt lines before enabling the channel.

7.5.2 Software-Triggered (Nonsynchronized) Transfer

To program a software-triggered DMA transfer:

1. Configure the transfer parameters in the logical DMA channel registers:
   - SDMA.DMA4_CSDPi:
     - Transfer ES (8 bits, 16 bits, or 32 bits) and DMA register bits SDMA.DMA4_CSDPi[1:0]
     - Read and write port access types (single/burst), DMA register bits SDMA.DMA4_CSDPi[8:7] and SDMA.DMA4_CSDPi[15:14]
     - Source and destination endianism, DMA register bits SDMA.DMA4_CSDPi[21] and SDMA.DMA4_CSDPi[19]
     - Write mode (posted or nonposted) and DMA register bits SDMA.DMA4_CSDPi[17:16]
     - Source or destination packed or nonpacked (if the ES is less than the read/write port size), DMA register bits SDMA.DMA4_CSDPi[6] and SDMA.DMA4_CSDPi[13]
   - SDMA.DMA4_CENi: EN
   - SDMA.DMA4_CFNi: FN per transfer block
   - SDMA.DMA4_CSSAi and SDMA.DMA4_CDSAi: Source and destination start address (aligned with transfer ES)
   - SDMA.DMA4_CCRi:
     - Read and write port addressing modes, DMA register bits SDMA.DMA4_CCRi[13:12] and SDMA.DMA4_CCRi[15:14]
     - Priority bit for both read and write ports, DMA register bits SDMA.DMA4_CCRi[6] and SDMA.DMA4_CCRi[26]
     - DMA request number (set to 0 for a software-triggered transfer) and DMA register bits SDMA.DMA4_CCRi[4:0] = 0 and SDMA.DMA4_CCRi[20:19] = 0
   - SDMA.DMA4_CSEi, SDMA.DMA4_CSFi, SDMA.DMA4_CDEi, and SDMA.DMA4_CDFi: Source and destination element and frame indexes (depending on addressing mode)
2. Start the transfer through the enable bit in the channel SDMA.DMA4_CCRi register and DMA register bit SDMA.DMA4_CCRi[7]

The example below perform a DMA transfer on channel 10 of a 240*160 picture from RAM to RAM (0x80C00000 to 0x80F00000) :

```
 UWORD32 RegVal = 0;
DMA4_t *DMA4;

DMA4 = (DMA4_t *)malloc(sizeof(DMA4_t));
/* Init. parameters */
DMA4->DataType = 0x2;                          // DMA4_CSDPi[1:0]
DMA4->ReadPortAccessType = 0;                  // DMA4_CSDPi[8:7]
DMA4->WritePortAccessType = 0;                 // DMA4_CSDPi[15:14]
DMA4->SourceEndiansim = 0;                     // DMA4_CSDPi[21]
DMA4->DestinationEndianism = 0;                // DMA4_CSDPi[19]
DMA4->WriteMode = 0;                           // DMA4_CSDPi[17:16]
DMA4->SourcePacked = 0;                        // DMA4_CSDPi[6]
DMA4->DestinationPacked = 0;                   // DMA4_CSDPi[13]
DMA4->NumberOfElementPerFrame = 240;           // DMA4_CENi
DMA4->NumberOfFramePerTransferBlock = 160;     // DMA4_CFNi
DMA4->SourceStartAddress = 0x80C00000;         // DMA4_CSSAi
DMA4->DestinationStartAddress = 0x80F00000;    // DMA4_CDSAi
DMA4->SourceElementIndex = 1;                  // DMA4_CSEi
DMA4->SourceFrameIndex = 1;                    // DMA4_CSFi
DMA4->DestinationElementIndex = 1;             // DMA4_CDEi
DMA4->DestinationFrameIndex = 1;               // DMA4_CDFi
DMA4->ReadPortAccessMode = 1;                  // DMA4_CCRi[13:12]
DMA4->WritePortAccessMode = 1;                 // DMA4_CCRi[15:14]
DMA4->ReadPriority = 0;                        // DMA4_CCRi[6]
DMA4->WritePriority = 0;                       // DMA4_CCRi[23]
DMA4->ReadRequestNumber = 0;                   // DMA4_CCRi[4:0]
DMA4->WriteRequestNumber = 0;                  // DMA4_CCRi[20:19]


/* 1) Configure the transfer parametres in the logical DMA registers */
/*-------------------------------------------------------------------*/
 /* a) Set the data type CSDP[1:0], the Read/Write Port access type CSDP[8:7]/[15:14], the
Source/dest endiansim CSDP[21]/CSDP[19], write mode
CSDP[17:16], source/dest packed or non-packed CSDP[6]/CSDP[13]*/
// Read CSDP
RegVal = DMA4_CSDP_CH10;

// Build reg
RegVal = ((RegVal & ~ 0x3) | DMA4->DataType );
RegVal = ((RegVal & ~(0x3 << 7))  | (DMA4->ReadPortAccessType << 7));
RegVal = ((RegVal & ~(0x3 << 14)) | (DMA4->WritePortAccessType << 14));
RegVal = ((RegVal & ~(0x1 << 21)) | (DMA4->SourceEndiansim << 21));
RegVal = ((RegVal & ~(0x1 << 19)) | (DMA4->DestinationEndianism << 19));
RegVal = ((RegVal & ~(0x3 << 16)) | (DMA4->WriteMode << 16));
RegVal = ((RegVal & ~(0x1 << 6))  | (DMA4->SourcePacked << 6));
RegVal = ((RegVal & ~(0x1 << 13)) | (DMA4->DestinationPacked << 13));

// Write CSDP
DMA4_CSDP_CH10 = RegVal;

/* b) Set the number of element per frame CEN[23:0]*/
DMA4_CEN_CH10 = DMA4->NumberOfElementPerFrame;

/* c) Set the number of frame per block CFN[15:0]*/
DMA4_CFN_CH10 = DMA4->NumberOfFramePerTransferBlock;

/* d) Set the Source/dest start address index CSSA[31:0]/CDSA[31:0]*/
DMA4_CSSA_CH10 = DMA4->SourceStartAddress; // address start
DMA4_CDSA_CH10 = DMA4->DestinationStartAddress; // address dest

/* e) Set t1he Read Port adressing mode CCR[13:12], the Write Port adressing mode CCR[15:14],
read/write priority CCR[6]/CCR[26], the current LCH CCR[20:19]=00 and CCR[4:0]=00000*/
// Read CCR
```

```
RegVal = DMA4_CCR_CH10;

// Build reg
RegVal = ((RegVal & ~(0x3 << 12)) | (DMA4->ReadPortAccessMode << 12));
RegVal = ((RegVal & ~(0x3 << 14)) | (DMA4->WritePortAccessMode << 14));
RegVal = ((RegVal & ~(0x1 << 6))  | (DMA4->ReadPriority << 6));
RegVal = ((RegVal & ~(0x1 << 26)) | (DMA4->WritePriority << 26));
RegVal&= 0xFFCFFFE0 ;

// Write CCR
DMA4_CCR_CH10 = RegVal;

/* f)- Set the source element index CSEI[15:0]*/
DMA4_CSEI_CH10 = DMA4->SourceElementIndex;

/* - Set the source frame index CSFI[15:0]*/
DMA4_CSFI_CH10 = DMA4->SourceFrameIndex ;

/* - Set the destination element index CDEI[15:0]*/
DMA4_CDEI_CH10 = DMA4->DestinationElementIndex;

/* - Set the destination frame index CDFI[31:0]*/
DMA4_CDFI_CH10 = DMA4->DestinationFrameIndex;

/* 2) Start the DMA transfer by Setting the enable bit CCR[7]=1 */
/*-------------------------------------------------------------*/
//write enable bit
DMA4_CCR_CH10        |= 1 << 7; /* start */
```

7.5.3 Hardware-Synchronized Transfer

To monitor a hardware synchronized DMA transfer, initialize the SDMA.DMA4_CDACi register before the software enable.

To configure an LCh to synchronize by element, packet, frame, or block, the frame synchronization SDMA.DMA4_CCRi[5] FS bit and the block synchronization SDMA.DMA4_CCRi[18] BS bit register must be programmed. For all the following synchronized transfers (element, packet, frame or block synchronized transfers) User must set first : SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC to 1 when the source triggers on the DMA request and SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC to 0 when the Destination triggers on the DMA request. Note: User must take care when setting the SDMA.DMA4_CCRi[23] PREFETCH bit it is in conjunction with SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC bit .

- To configure an LCh to transfer one element per DMA request:

1. Set the number of DMA request associated to the current LCH in the SDMA.DMA4_CCRi[20:19] SYNCHRO_CONTROL_UPPER and SDMA.DMA4_CCRi[4:0] SYNCHRO bit field.
2. Set the data type, also referenced as element size (ES), in the SDMA.DMA4_CSDPi[1:0] DATA_TYPE bit field.
3. Set the Read Port access type (single or burst access) in the SDMA.DMA4_CSDPi[8:7] SRC_BURST_EN bit field.
4. Set the Write Port access type (single or burst access) in the SDMA.DMA4_CSDPi[15:14] DST_BURST_EN bit field.
5. Set the Read Port addressing mode in the SDMA.DMA4_CCRi[13:12] SRC_AMODE bit field.
6. Set the Write Port addressing mode in the SDMA.DMA4_CCRi[15:14] DST_AMODE bit field.
7. Set the Read start address in the SDMA.DMA4_CSSAi[31:0] SRC_START_ADRS bit field.
8. Set the Write start address in the SDMA.DMA4_CDSAi[31:0] DST_START_ADRS bit field.
9. Set both FS and BS to 0 in SDMA.DMA4_CCRi[5] FS and SDMA.DMA4_CCRi[18] BS.
10. Set to 1 the channel enable bit SDMA.DMA4_CCRi[7] EN bit.

- To configure an LCh to transfer one frame per DMA request:

1. Set the number of DMA request associated to the current LCH in the SDMA.DMA4_CCRi[20:19] SYNCHRO_CONTROL_UPPER and SDMA.DMA4_CCRi[4:0] SYNCHRO bit field.
2. Set the data type, also referenced as element size (ES), in the SDMA.DMA4_CSDPi[1:0] DATA_TYPE bit field.
3. Set the number of element per frame in the SDMA.DMA4_CENi[23:0] CHANNEL_ELMNT_NBR bit field.
4. Set the Read Port access type (single or burst access) in the SDMA.DMA4_CSDPi[8:7] SRC_BURST_EN bit field.
5. Set the Write Port access type (single or burst access) in the SDMA.DMA4_CSDPi[15:14] DST_BURST_EN bit field.
6. Set the Read Port addressing mode in the SDMA.DMA4_CCRi[13:12] SRC_AMODE bit field.
7. Set the Write Port addressing mode in the SDMA.DMA4_CCRi[15:14] DST_AMODE bit field.
8. Set the Read start address in the SDMA.DMA4_CSSAi[31:0] SRC_START_ADRS bit field.
9. Set the Write start address in the SDMA.DMA4_CDSAi[31:0] DST_START_ADRS bit field.
10. Set FS to 1 and BS to 0 respectively in SDMA.DMA4_CCRi[5] FS and SDMA.DMA4_CCRi[18] BS.
11. Set to 1 the channel enable bit SDMA.DMA4_CCRi[7] EN bit.

- To configure an LCh to transfer one block per DMA request:

1. Set the number of DMA request associated to the current LCH in the SDMA.DMA4_CCRi[20:19] SYNCHRO_CONTROL_UPPER and SDMA.DMA4_CCRi[4:0] SYNCHRO bit field.
2. Set the data type, also referenced as element size (ES), in the SDMA.DMA4_CSDPi[1:0] DATA_TYPE bit field.
3. Set the number of element per frame in the SDMA.DMA4_CENi[23:0] CHANNEL_ELMNT_NBR bit field.
4. Set in the SDMA.DMA4_CFNi[15:0] CHANNEL_FRAME_NBR bit field the number of frame (transfers), to take place before the LCH is disabled.
5. Set the Read Port access type (single or burst access) in the SDMA.DMA4_CSDPi[8:7] SRC_BURST_EN bit field.
6. Set the Write Port access type (single or burst access) in the SDMA.DMA4_CSDPi[15:14] DST_BURST_EN bit field.
7. Set the Read Port addressing mode in the SDMA.DMA4_CCRi[13:12] SRC_AMODE bit field.
8. Set the Write Port addressing mode in the SDMA.DMA4_CCRi[15:14] DST_AMODE bit field.
9. Set the Read start address in the SDMA.DMA4_CSSAi[31:0] SRC_START_ADRS bit field.
10. Set the Write start address in the SDMA.DMA4_CDSAi[31:0] DST_START_ADRS bit field.
11. Set FS to 0 and BS to 1 respectively in SDMA.DMA4_CCRi[5] FS and SDMA.DMA4_CCRi[18] BS.
12. Set to 1 the channel enable bit SDMA.DMA4_CCRi[7] EN bit.

- To configure an LCh to transfer one packet per DMA request:

1. Set the number of DMA request associated to the current LCH in the SDMA.DMA4_CCRi[20:19] SYNCHRO_CONTROL_UPPER and SDMA.DMA4_CCRi[4:0] SYNCHRO bit field.
2. Set the data type, also referenced as element size (ES), in the SDMA.DMA4_CSDPi[1:0] DATA_TYPE bit field.
3. Set the number of element per packet to transfer: If the packet requestor is in the source, set SDMA.DMA4_CCR.Sel_Src_Dst_Sync to 1 and set the packet element number in the SDMA.DMA4_CSFIi register; else, if the packet requestor is in the destination, set the SDMA.DMA4_CCRi[24] SEL_SRC_DST_SYNC to 0 and set the packet element number in the SDMA.DMA4_CDFIi register.
4. Set the number of element per frame in the SDMA.DMA4_CENi[23:0] CHANNEL_ELMNT_NBR bit field.
5. Set in the SDMA.DMA4_CFNi[15:0] CHANNEL_FRAME_NBR bit field the number of frame (transfers), to take place before the LCH is disabled.
6. Set the element number in the packet in the SDMA.DMA4_CSFIi[15:0] PKT_ELNT_NBR, if constant

addressing or post-incremented addressing modes are used in the source side. However, the number of elements in the packet is set in the SDMA.DMA4_CDFIi[15:0] PKT_ELNT_NBR if constant addressing mode is used in the destination side.

7. Set the Read Port access type (single or burst access) in the SDMA.DMA4_CSDPi[8:7] SRC_BURST_EN bit field.
8. Set the Write Port access type (single or burst access) in the SDMA.DMA4_CSDPi[15:14] DST_BURST_EN bit field.
9. Set the Read Port addressing mode in the SDMA.DMA4_CCRi[13:12] SRC_AMODE bit field.
10. Set the Write Port addressing mode in the SDMA.DMA4_CCRi[15:14] DST_AMODE bit field.
11. Set the Read start address in the SDMA.DMA4_CSSAi[31:0] SRC_START_ADRS bit field.
12. Set the Write start address in the SDMA.DMA4_CDSAi[31:0] DST_START_ADRS bit field.
13. Set FS to 1 and BS to 1 respectively in SDMA.DMA4_CCRi[5] FS and SDMA.DMA4_CCRi[18] BS.
14. Set to 1 the channel enable bit SDMA.DMA4_CCRi[7] EN bit.

NOTE: It is possible to stop a transfer by disabling the channel. This is done by resetting the ENABLE bit in the SDMA.DMA4_CCRi register.

7.5.4 Synchronized Transfer Monitoring Using CDAC

The SDMA.DMA4_CDACi register is writable and non-initialized (value undefined). It can be initialized to monitor a transfer by applying the following programming model:

1. Write 0 in the SDMA.DMA4_CDACi
2. Enable the channel.
3. If timeout occurs, read SDMA.DMA4_CDACi
4. If SDMA.DMA4_CDACi!= SDMA.DMA4_CDACi reset value:

   Then transfer starts. User can then rely on SDMA.DMA4_CCENi and SDMA.DMA4_CCFNi element and frame counters.

   Else, if SDMA.DMA4_CDACi = SDMA.DMA4_CDACi reset value:

   Then transfer does not start.

7.5.5 Concurrent Software and Hardware Synchronization

This section describes thread allocation only, not the entire transfer. Because synchronized transfers are latency critical, you must allocate a thread on the synchronized target side at least.

Even for multiple concurrent channels, thread reservation guarantees that as soon as an HW DMA request comes in, the read/write scheduler finds available thread(s) to initiate a channel schedule and issue a read/write transaction.

Consider these six concurrent channels:

- Channels 0/1/2/3 are dedicated to memory-memory transfer: they are software triggered and not synchronized.
- Channel 4 is dedicated to memory→peripheral transfer, hardware triggered, and synchronized on the write side.
- Channel 5 is dedicated to peripheral→memory transfer, hardware triggered, and synchronized on the read side.

1. Allow thread reservation for priority channel 4 and channel 5:

   Reserve one thread (Read ThreadID 0) on the read port: Set SDMA.DMA4_GCR[13:12] = 0x1.

   Reserve one thread (Write ThreadID 0) on the write port: Set SDMA.DMA4_GCR[13:12] = 0x1.
2. Specify channel priority:

   Channel 4 is a write high priority channel: Set SDMA.DMA4_CCRi[26] = 1.

   Channel 5 is a read high priority channel: Set SDMA.DMA4_CCRi[6] = 1.

7.5.6 Chained Transfer

A chained DMA transfer can be programmed as follows:

1. Configure the transfer parameters for each logical DMA channel in the chain as in step 1 for either the synchronized or non-synchronized transfers above.
2. For each channel in the chain, configure the SDMA.DMA4_CLNK_CTRLi register as follows:
   - Next logical DMA channel number (for a looping chained transfer link last channel to first channel number), in DMA register bits SDMA.DMA4_CLNK_CTRLi[4:0].
   - Include the logical channel to the chain and enable link by setting the DMA register bit SDMA.DMA4_CLNK_CTRLi[15].
   - For a non-looping chain, the last logical channel in the chain must have the DMA register bit SDMA.DMA4_CLNK_CTRLi[15] set to 0 to indicate the end of the chain.
3. Enable the transfer via the enable bit in the first logical channel DMA register bit SDMA.DMA4_CCRi[7]. All other channels in the chain must be configured as disabled. Each channel is enabled automatically in turn when the previous logical channel transfer completes. A non-synchronized transfer starts immediately; a hardware-synchronized transfer starts when the DMA request line corresponding to the first DMA channel in the chain is asserted.

To stop a looping chained transfer, disable the NEXTLCH_ID bit, DMA register bit SDMA.DMA4_CLNK_CTRLi[15](ENABLE_LNK bit set to 0x0), of the final channel transfer.

In the RAM to RAM copy example, to copy in loop it's possible to link channel 10 on itself. The following line can be added in the channel configuration :

```
 /* g) Set link for loop */
DMA4_CLINK_CTRL_CH10 = 0x0000800A;
```

7.5.7 90-Degree Clockwise Image Rotation

The 90-degree clockwise image rotation example described in Section 7.4.3, *Addressing Modes*, can be programmed as follows:

1. Configure the transfer parameters in the logical DMA channel registers:
   - SDMA.DMA4_CSDPi:
     - Transfer ES = 32-bit (32 bpp), DMA register bits SDMA.DMA4_CSDPi[1:0]
     - Read and write port access types = maximum burst size supported by memory device, DMA register bits SDMA.DMA4_CSDPi[8:7] and SDMA.DMA4_CSDPi[15:14]
     - Source and destination endianism, DMA register bits SDMA.DMA4_CSDPi[21] and SDMA.DMA4_CSDPi[19]
     - Write mode = posted with last element nonposted, DMA register bits SDMA.DMA4_CSDPi[17:16]
     - Source and destination packed = Yes (although destination writes will not benefit because EI>1), DMA register bits SDMA.DMA4_CSDPi[6] and SDMA.DMA4_CSDPi[13]
   - SDMA.DMA4_CENi: EN = 240
   - SDMA.DMA4_CFNi: FN per transfer block = 160
   - SDMA.DMA4_CSSAi: Source start address = 0x100000
   - SDMA.DMA4_CDSAi: destination start address = 0x20013E
   - SDMA.DMA4_CCRi:
     - Read and write port addressing modes = double-index addressing mode for both or post-increment addressing on source and double-index addressing on destination, DMA register bits SDMA.DMA4_CCRi[13:12] and SDMA.DMA4_CCRi[15:14]
     - Low or high priority, DMA register bit SDMA.DMA4_CCRi[6]
     - DMA request number = 0 (for software-triggered transfer), DMA register bits SDMA.DMA4_CCRi[4:0] and SDMA.DMA4_CCRi[20:19]
   - SDMA.DMA4_CSEi: Source EI = 1
   - SDMA.DMA4_CSFi: Source frame index = 1

- SDMA.DMA4_CDEi: destination EI = 637
- SDMA.DMA4_CDFi: destination frame index = –152967

2. Start the transfer via the enable bit in the channel SDMA.DMA4_CCRi register.

Below are the parameters to perform this rotation from 0x80C00000 RAM address to 0x80F00000, with the same code as in Section 7.5.2:

```
 /* Init. parameters */
DMA4->DataType = 0x2;                       // DMA4_CSDPi[1:0]
DMA4->ReadPortAccessType = 0x3;             // DMA4_CSDPi[8:7]
DMA4->WritePortAccessType = 0x3;            // DMA4_CSDPi[15:14]
DMA4->SourceEndiansim = 0;                  // DMA4_CSDPi[21]
DMA4->DestinationEndianism = 0;             // DMA4_CSDPi[19]
DMA4->WriteMode = 0x2;                      // DMA4_CSDPi[17:16]
DMA4->SourcePacked = 0x1;                   // DMA4_CSDPi[6]
DMA4->DestinationPacked = 0x1;              // DMA4_CSDPi[13]
DMA4->NumberOfElementPerFrame = 240;        // DMA4_CENi
DMA4->NumberOfFramePerTransferBlock = 160;  // DMA4_CFNi
DMA4->SourceStartAddress = 0x80C00000;      // DMA4_CSSAi
DMA4->DestinationStartAddress = 0x80F00000; // DMA4_CDSAi
DMA4->SourceElementIndex = 1;               // DMA4_CSEi
DMA4->SourceFrameIndex = 1;                 // DMA4_CSFi
DMA4->DestinationElementIndex = 637;        // DMA4_CDEi
DMA4->DestinationFrameIndex = -152967;      // DMA4_CDFi
DMA4->ReadPortAccessMode = 0x3;             // DMA4_CCRi[13:12]
DMA4->WritePortAccessMode = 0x3;            // DMA4_CCRi[15:14]
DMA4->ReadPriority = 0;                     // DMA4_CCRi[6]
DMA4->WritePriority = 0;                    // DMA4_CCRi[23]
DMA4->ReadRequestNumber = 0;                // DMA4_CCRi[4:0]
DMA4->WriteRequestNumber = 0;               // DMA4_CCRi[20:19]
```

7.5.8 Graphic Operations

- Transparent copy:
  - 1. Set the SDMA.DMA4_CCRi[17] Transparent_Copy_Enable bit field to 1.
  - 2. Set the SDMA.DMA4_CCRi[16] Constant_Fill_Enable bit field to 0.
  - 3. Set the value of key the color in the SDMA.DMA4_COLORi[15:0] color_key bit field.

  To perform this graphic operation, the following lines can be added to the example of Section 7.5.2

```
DMA4_CCR_CH10 &= ~(0x1 << 16);
DMA4_CCR_CH10 |= 0x1 << 17;
DMA4_COLOR_CH10 = 0x00000003;
```

- Solid Color fill:
  - 1. Set the SDMA.DMA4_CCRi[16] Constant_Fill_Enable bit field to 1.
  - 2. Set the SDMA.DMA4_CCRi[17] Transparent_Copy_Enable bit field to 0.
  - 3. Set the value of key the color in the DMA4_COLORi[15:0] solid_color bit field.

  To perform this graphic operation, the following lines can be added to the example of Section 7.5.2

```
DMA4_CCR_CH10 &= ~(0x1 << 17);
DMA4_CCR_CH10 |= 0x1 << 16;
DMA4_COLOR_CH10 = 0x00000003;
```

7.6 SDMA Use Cases and Tips

7.6.1 Camcorder Use Case: How to Configure SDMA to Handle Transfers With McBSP2

and MMC to External DRAM

7.6.1.1 Introduction

In this use case, the SDMA manages:

- The audio stream between the McBSP and the external DRAM
- The audio stream between the external DRAM and the MMC buffer
- The video stream between the external DRAM and the MMC buffer

The following sections describe how to configure the SDMA controller with MMC and McBSP in this use case.

7.6.1.2 SDMA Configuration to Transfer Data Between the McBSP and External DRAM

7.6.1.2.1 Overview

The SDMA gets data from the McBSP_DRR register and copies it into three rolling buffers in the external DRAM.

Figure 7-13 is an overview of the audio path.

Figure 7-13. Overview

For the camcorder use case, the DMA transfer data format is:

- Frame size = 2048 x 32 bits
- Three DMA channels to manage three transfers to three memory buffers
- Interrupt generated at the end of each frame transfer
- Transfer triggered by McBSP2

7.6.1.2.2 Environment

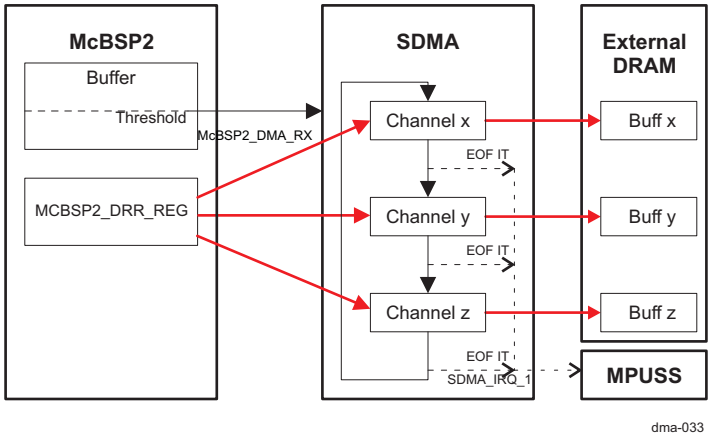
The DMA accesses MCBSP2_DRR_REG through the interconnect and receives a DMA request from this McBSP module for driving transfers. The DMA accesses the external DRAM through the interconnect. The McBSP is directly connected to the device. Figure 7-14 shows the environment of this scenario.

Figure 7-14. Environment

7.6.1.2.3 Data Path

The transfer is started when the DMA transfer is enabled and the MCBSP2_DMA_REQ is asserted (that is, when the MCBSP FIFO threshold is reached). The DMA starts copying the first frame into the first memory buffer. When a whole frame has been transferred (2048 data), an interrupt is asserted to the MPU, and then the second channel is enabled and fills the second buffer, and so on. Figure 7-15 shows the data flow.

Figure 7-15. Data Flow



7.6.1.2.4 Programming Flow

To establish a configuration for the DMA transfers, the following points have been considered.

The McBSP2_DRR_REG register is 32 bits wide, but in the camcorder use case, only the 16 LSBs are used (monophonic 16-bit-wide audio channel).

The McBSP receives data in a FIFO buffer. As long as the occupied locations level in this read buffer is greater than or equal to the threshold value + 1 , the DMA request is asserted. After transferring the configured (THRSH1_REG value + 1) number of words, the receive DMA request is deasserted and then reasserted when the conditions are met again. Therefore, the DMA can receive at least threshold value data, with a threshold maximum value = 1280 elements = size of McBSP2 FIFO.

The frame synchronization transfer mode of the SDMA can be used with McBSP for frames with a size equal to the threshold. In the use case, however, the audio data frame is equal to 2048 elements, which is greater than the maximum transfer length possible with the McBSP.

Because the SDMA cannot be configured in frame synchronization mode, packet synchronization mode must be used. This mode enables to specify the maximum size of a transfer and to divide a frame into as many packet as needed. It is then possible to configure a frame of n elements and generate an interrupt at the end-of-frame transfer, while the transfer is segmented into packets.

The management of the three memory buffers is done by configuring three DMA channels linked between themselves: channel x is linked to channel y, channel y is linked to channel z, and channel z is linked to channel x. The three channels have the same configuration, except for the destination address, which corresponds to the three buffer addresses. Channels 11, 12, and 13 are used. Each channel generates the end-of-frame interrupts on the L1 interrupt line of the SDMA, to warn the MPU of the end of a frame transfer.

With these considerations, the DMA register of the three channels (i = 11, 12, and13) is configured as follows:

1. Channel source destination parameters: DMA4_CSDPi
   - DMA4_CSDPi[1:0] DATA_TYPE = 0x1: Read 16-bit elements from the McBSP_DRR_REG register.
   - DMA4_CSDPi[5:2] RD_ADD_TRSLT = 0: Not useful
   - DMA4_CSDPi[6] SRC_PACKED = 0x0: Cannot pack source data
   - DMA4_CSDPi[8:7] SRC_BURST_EN = 0x0: Cannot burst source
   - DMA4_CSDPi[12:9] WR_ADD_TRSLT = 0: Undefined
   - DMA4_CSDPi[13] DST_PACKED = 0x1: Pack two 16-bit elements in one 32-bit packet to optimize transfer.
   - DMA4_CSDPi[15:14] DST_BURST_EN = 0x3: Burst at 16x32-bit
   - DMA4_CSDPi[17:16] WRITE_MODE = 0x1: Write posted

- DMA4_CSDPi[18] DST_ENDIAN_LOCK = 0x0: Endianness adapt
- DMA4_CSDPi[19] DST_ENDIAN = 0x0: Little endian type at destination
- DMA4_CSDPi[20] SRC_ENDIAN_LOCK = 0x0: Endianness adapt
- DMA4_CSDPi[21] SRC_ENDIAN = 0x0: Little endian type at source
- DMA4_CSDPi[31:22] RESERVED = 0x0: For future compatibility

2. Channel control register: DMA4_CCRi
   - DMA4_CCRi[4:0] SYNCHRO_CONTROL = DmaReq & 0x1F = 0x2: 5 first bits of McBSP2_DMA_RX
   - DMA4_CCRi[5] FS = 1: Packet mode with BS = 0x1
   - DMA4_CCRi[6] READ_PRIORITY = 0x0: Low priority on read side
   - DMA4_CCRi[7] ENABLE = 0x0: The logical channel is disabled.
   - DMA4_CCRi[8] SUSPEND_SENSITIVE = 0
   - DMA4_CCRi[9] RD_ACTIVE: Read status, read-only access
   - DMA4_CCRi[10] WR_ACTIVE: Write status, read-only access
   - DMA4_CCRi[11] RESERVED = 0: Write 0s for future compatibility.
   - DMA4_CCRi[13:12] SRC_AMODE = 0x0: Constant address mode; DMA always reads McBSP2_DRR_REG.
   - DMA4_CCRi[15:14] DST_AMODE = 0x1: Post-incremented address mode
   - DMA4_CCRi[16] CONST_FILL_ENABLE = 0x0: Constant fill mode is disabled.
   - DMA4_CCRi[17] TRANSPARENT_COPY_ENABLE = 0x0: Transparent copy mode is disabled.
   - DMA4_CCRi[18] BS = 0x1: Packet mode with FS = 0x1
   - DMA4_CCRi[20:19] SYNCHRO_CONTROL_UPPER = 0x1: Two MSBs of McBSP2_DMA_RX
   - DMA4_CCRi[22] SUPERVISOR = 0x0: Supervisor mode is disabled.
   - DMA4_CCRi[23] PREFETCH = 0x0: Prefetch mode is disabled, cannot prefetch a constant adressing source.
   - DMA4_CCRi[24] SEL_SRC_DST_SYNC = 0x1: Transfer is triggered by the source. The packet element number is specified in the DMA4_CSFI register.
   - DMA4_CCRi[25] BUFFERING_DISABLE = 0x0
   - DMA4_CCRi[26] WRITE_PRIORITY = 0x0: Channel has low priority on Write side during the arbitration process.
   - DMA4_CCRi[31:27] RESERVED = 0x0: Write 0s for future compatibility.
3. Channel parameters: DMA4_CENi, DMA4_CFNi, DMA4_CSSAi, DMA4_CDSAi, DMA4_CSEi, DMA4_CSFi, DMA4_CDEi
   - DMA4_CENi[23:0] CHANNEL_ELMNT_NBR = 2048: 2048 elements
   - DMA4_CFNi[15:0] CHANNEL_FRAME_NBR = 1: One frame
   - DMA4_CSSAi[31:0] SRC_START_ADRS = 0x49022000: Channel source start address = MCBSP2_DRR_REG
   - DMA4_CDSAi[31:0] DST_START_ADRS= X, Y or Z: Channel destination start address in external DRAM, where X, Y, and Z represent the addresses of the three buffers
   - DMA4_CSEi[31:0] CHANNEL_SRC_ELMNT_INDEX = 1: Channel source element index
   - DMA4_CSFi[15:0] 16BIT_PKT_ELNT_NBR = 0x80: 16-bit Packet size = MCBSP_FIFO_THRESHOLD + 1
   - DMA4_CDEi[15:0] CHANNEL_DST_ELMNT_INDEX = 0x1: Channel destination element index
4. Channel linking: DMA4_CLNK_CTRLi
   - DMA4_CLNK_CTRL11[31:0] = 0x0000800C: Channel 11 is linked to channel 12.
   - DMA4_CLNK_CTRL12[31:0] = 0x0000800D: Channel 12 is linked to channel 13.
   - DMA4_CLNK_CTRL13[31:0] = 0x0000800B: Channel 13 is linked to channel 11.
5. Interrupt management: DMA4_CICRi and DMA4_IRQENABLE_Lj

- DMA4_CICRi[31:0] = 0x00000008: Enables the end of frame interrupt and disables others (the value of these registers is unknown after a reset; it is necessary to clear other bits)
- DMA4_IRQENABLE_L1[31:0] = 0x00001C00; interrupts of channels 11, 12, and 13 are unmasked on IRQ line L1.

6. Launch transfer: DMA4_CCRi[7] ENABLE = 0x1: Enables the three channels, starting with channel 13, then 12, and finally 11; the last one triggers the others.

Table 7-9 lists the registers of DMA channel 11 after a first transfer.

Table 7-9. Registers Print

Register Name	Address	Value	Value Description
DMA4_IRQENABLE_L1	0x4805601c	0x00001C00	Interrupts of channels 11, 12, and 13 are unmasked on IRQ line L1.
DMA4_CICR11	0x480564a8	0x00000008	End-of-frame interrupt enabled
DMA4_CCR11	0x480564a0	0x010c40a2	Channel control register
DMA4_CSDP11	0x480564b0	0x0001e001	Channel source destination parameters
DMA4_CEN11	0x480564b4	0x00000800	Channel element number
DMA4_CFN11	0x480564b8	0x00000001	Channel frame number
DMA4_CSSA11	0x480564bc	0x49022000	Channel source start address
DMA4_CDSA11	0x480564c0	0x81e00000	Channel destination start address
DMA4_CSFI11	0x480564c8	0x00000080	Packet size
DMA4_CSAC11	0x480564d4	0x49022000	Source address counter (read only)
DMA4_CDAC11	0x480564d8	0x81e01000	Destination address counter (read only)
DMA4_CCEN11	0x480564dc	0x00000800	Channel current transferred element number in the current frame (read only)
DMA4_CCFN11	0x480564e0	0x00000001	Channel current transferred frame number in the current transfer (read only)
DMA4_CLNK_CTRL11	0x480564a4	0x0000800C	Channel link control register: Link to channel 12

Channels 12 and 13 have a similar configuration; only the destination address and channel link change.

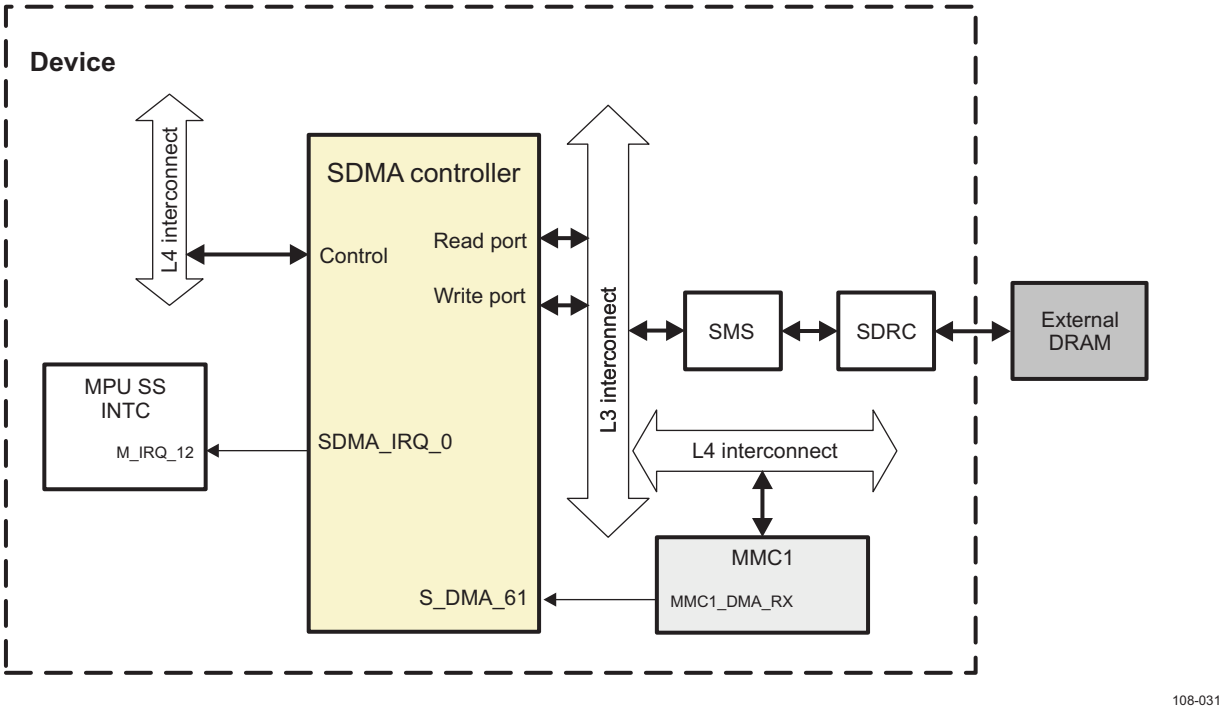
7.6.1.3 SDMA Configuration to Transfer Data Between MMC and External DRAM

7.6.1.3.1 Overview

The SDMA gets data from the MMC.MMCHS_DATA register and copies it into the external DRAM. The SDMA accesses the MMCHS_DATA through the interconnect and receives a DMA request from the MMC module for driving transfers. The DMA accesses the external DRAM through the interconnect. The transfer is started when the DMA transfer is enabled and the MMC1_DMA_REQ is asserted (that is, when the MMC buffer is ready).

Figure 7-16 is an overview of the path.

Figure 7-16. Overview



For the camcorder use case, the DMA transfer data format is:

- Frame size = 4096 x 32 bits
- packet size = 512 x 32 bits (512 elements ready in MMC when a DMA request is asserted)
- DMA channel to manage the transfer
- Interrupt generated a the end of each frame transfer
- Transfer triggered by MMC

7.6.1.3.2 Programming Flow

To establish a configuration for the DMA transfers, the following points have been considered.

The MMCHS_DATA register is 32 bits wide.

The SDMA is configured in packet synchronization mode. This mode enables to specify the transfer of any size frame and then to divide the frame into as many packet as needed. It is therefore possible to configure a frame of n elements and generate an interrupt at the end-of-frame transfer, while the transfer is segmented into packets.

The MMC puts its received data into a 1K-byte ping-pong buffer. When 512 x 32 bits are ready in this buffer, the DMA request is asserted. Then the DMA controller transfers the defined amount of elements divided into packets of 512 x 32-bit element (PKT_ELNT_NBR) transfers.

In the use case, the size of the transfer is set at 4096 x 32 bits, chosen after performance benches.

With these considerations, the DMA register of channel i is configured as follows:

1. Channel source destination parameters: DMA4_CSDPi
   - DMA4_CSDPi[1:0] DATA_TYPE = 0x2: Read 32-bit elements from the MMCHS_DATA register.
   - DMA4_CSDPi[5:2] RESERVED: Write 0s for future compatibility. Read returns 0.
   - DMA4_CSDPi[6] SRC_PACKED = 0x0: Cannot pack source data
   - DMA4_CSDPi[8:7] SRC_BURST_EN = 0x0: Cannot burst source
   - DMA4_CSDPi[12:9] RESERVED: Write 0s for future compatibility. Read returns 0.

- DMA4_CSDPi[13] DST_PACKED = 0x0: No packing
- DMA4_CSDPi[15:14] DST_BURST_EN = 0x3: Burst at 16x32 bits
- DMA4_CSDPi[17:16] WRITE_MODE = 0x1: Write posted
- DMA4_CSDPi[18] DST_ENDIAN_LOCK = 0x0: Endianness adapt
- DMA4_CSDPi[19] DST_ENDIAN = 0x0: Little Endian type at destination
- DMA4_CSDPi[20] SRC_ENDIAN_LOCK = 0x0: Endianness adapt
- DMA4_CSDPi[21] SRC_ENDIAN = 0x0: Little endian type at source
- DMA4_CSDPi[31:22] RESERVED = 0x0: For future compatibility

2. Channel control register: DMA4_CCRi
   - DMA4_CCRi[4:0] SYNCHRO_CONTROL = DmaReq & 0x1F = 0x1E: 5 first bits of MMC1_DMA_RX(62)
   - DMA4_CCRi[5] FS = 1: Packet mode with BS = 0x1
   - DMA4_CCRi[6] READ_PRIORITY = 0x0: Low priority on read side
   - DMA4_CCRi[7] ENABLE = 0x0: The logical channel is disabled.
   - DMA4_CCRi[8] SUSPEND_SENSITIVE = 0
   - DMA4_CCRi[9] RD_ACTIVE: Read status, read-only access
   - DMA4_CCRi[10] WR_ACTIVE: Write status, read-only access
   - DMA4_CCRi[11] RESERVED = 0: Write 0s for future compatibility.
   - DMA4_CCRi[13:12] SRC_AMODE = 0x0: Constant address mode; DMA always reads the MMCHS_DATA.
   - DMA4_CCRi[15:14] DST_AMODE = 0x1: Post-incremented address mode
   - DMA4_CCRi[16] CONST_FILL_ENABLE = 0x0: Constant fill mode is disabled.
   - DMA4_CCRi[17] TRANSPARENT_COPY_ENABLE = 0x0: Transparent copy mode is disabled.
   - DMA4_CCRi[18] BS = 0x1: Packet mode with FS = 0x1
   - DMA4_CCRi[20:19] SYNCHRO_CONTROL_UPPER = 0x1: Two MSB of MMC1_DMA_RX(62)
   - DMA4_CCRi[22] SUPERVISOR = 0x0: Supervisor mode is disabled.
   - DMA4_CCRi[23] PREFETCH = 0x0: Prefetch mode is disabled; cannot prefetch a constant adressing source.
   - DMA4_CCRi[24] SEL_SRC_DST_SYNC = 0x1: Transfer is triggered by the source. The packet element number is specified in the DMA4_CSFI register.
   - DMA4_CCRi[25] BUFFERING_DISABLE = 0x0
   - DMA4_CCRi[26] WRITE_PRIORITY = 0x0: Channel has low priority on write side during the arbitration process.
   - DMA4_CCRi[31:27] RESERVED = 0x0: Write 0s for future compatibility.
3. Channel parameters: DMA4_CENi, DMA4_CFNi, DMA4_CSSAi, DMA4_CDSAi, DMA4_CSEi, DMA4_CSFi, DMA4_CDEi
   - DMA4_CENi[23:0] CHANNEL_ELMNT_NBR = 4096: 4096 elements
   - DMA4_CFNi[15:0] CHANNEL_FRAME_NBR = 1: One frame
   - DMA4_CSSAi[31:0] SRC_START_ADRS = 0x4809c120: Channel source start address = MMCHS_DATA
   - DMA4_CDSAi[31:0] DST_START_ADRS= X: Channel destination start address in external DRAM
   - DMA4_CSEi[31:0] CHANNEL_SRC_ELMNT_INDEX = 1: Channel source element index
   - DMA4_CSFi[15:0] 16BIT_PKT_ELNT_NBR = 0x200: Packet size = number odd data available in MMC after DMA req
   - DMA4_CDEi[15:0] CHANNEL_DST_ELMNT_INDEX = 0x1: Channel destination element Index
4. Interrupt management: DMA4_CICRi and DMA4_IRQENABLE_Lj
   - DMA4_CICRi[31:0] = 0x00000008: Enables the end-of-frame interrupt and disables others (the value of these registers is unknown after a reset; it is necessary to clear other bits)

- DMA4_IRQENABLE_L0[31:0] = 0xi: Interrupt of channel i is unmasked on IRQ line L0.

5. Launch transfer: DMA4_CCRi[7] ENABLE = 0x1: Enables the channel

Table 7-10 lists the DMA channel 2 registers after a first transfer.

Table 7-10. Registers Print

Register Name	Address	Value	Value Description
DMA4_IRQENABLE_L0	0x48056018	0x00000003	Interrupt of channel 2 is unmasked on IRQ line L0
DMA4_CICR2	0x48056148	0x00000008	End-of-frame interrupt enabled
DMA4_CCR2	0x48056140	0x010c403e	Channel control register
DMA4_CSDP2	0x48056150	0x0001c003	Channel source destination parameters
DMA4_CEN2	0x48056154	0x00000080	Channel element number
DMA4_CFN2	0x48056158	0x00000001	Channel frame number
DMA4_CSSA2	0x4805615c	0x4809c120	Channel source start address
DMA4_CDSA2	0x48056160	0x8071aafc	Channel destination start address
DMA4_CSFI2	0x48056168	0x00000080	Packet size
DMA4_CSAC2	0x48056174	0x49022000	Source address counter (read only)
DMA4_CDAC2	0x48056178	0x8071acfc	Destination address counter (read only)
DMA4_CCEN2	0x4805617c	0x00000080	Channel current transferred element number in the current frame (read only)
DMA4_CCFN12	0x48056180	0x00000000	Channel current transferred frame number in the current transfer (read only)
DMA4_CLNK_CTRL2	0x48056144	0x00000000	Channel link control register: No link

7.7 SDMA Registers Manual

This section provides a global view of the memory mapping as seen from the MPU for SDMA. Table 7-11 show the DMA register base address.

7.7.1 SDMA Instance Summary

Table 7-11. SDMA Instances Summary

Module Name	Base Address	Size
SDMA	0x4805 6000	4K bytes

7.7.2 SDMA Register Summary

Table 7-12 lists all DMA4 controller registers and their physical addresses. Table 7-13 through Table 7-67 describe the individual register bits.

Index *i* represents the logical channel number (i = 0 to 31). The offset address for some registers is calculated from channel *c* number. For example, register SDMA.DMA4_CCR10 (channel 10) has an offset address of 10*0x60 = 0x3C0, and so a physical address of 0x4800 A080 + 0x3C0 = 0x4800 A440.

Index *j* represents the interrupt line number (j = 0 to 3) The offset address for some registers is calculated from channel *c*number. For example, register SDMA.DMA4_IRQSTATUS_L3 (line 3) has an offset address of 3*0x4 = 0xC, and so a physical address of 0x4800 A008 + 0xC = 0x4800 A014.

Table 7-12. SDMA Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	SDMA Physical Address
DMA4_REVISION	R	32	0x0000 0000	0x4805 6000
DMA4_IRQSTATUS_Lj	RW	32	0x0000 0008 + (j * 0x4)	0x4805 6008 + (j * 0x4)
DMA4_IRQENABLE_Lj	RW	32	0x0000 0018 + (j * 0x4)	0x4805 6018 + (j * 0x4)
DMA4_SYSSTATUS	R	32	0x0000 0028	0x4805 6028
DMA4_OCP_SYSCONFIG	RW	32	0x0000 002C	0x4805 602C
DMA4_CAPS_0	R	32	0x0000 0064	0x4805 6064
DMA4_CAPS_2	R	32	0x0000 006C	0x4805 606C
DMA4_CAPS_3	R	32	0x0000 0070	0x4805 6070
DMA4_CAPS_4	R	32	0x0000 0074	0x4805 6074
DMA4_GCR	RW	32	0x0000 0078	0x4805 6078
DMA4_CCRi	RW	32	0x0000 0080 + (i * 0x60)	0x4805 6080 + (i * 0x60)
DMA4_CLNK_CTRLi	RW	32	0x0000 0084 + (i * 0x60)	0x4805 6084 + (i * 0x60)
DMA4_CICRi	RW	32	0x0000 0088 + (i * 0x60)	0x4805 6088 + (i * 0x60)
DMA4_CSRi	RW	32	0x0000 008C + (i * 0x60)	0x4805 608C + (i * 0x60)
DMA4_CSDPi	RW	32	0x0000 0090 + (i * 0x60)	0x4805 6090 + (i * 0x60)
DMA4_CENi	RW	32	0x0000 0094 + (i * 0x60)	0x4805 6094 + (i * 0x60)
DMA4_CFNi	RW	32	0x0000 0098 + (i * 0x60)	0x4805 6098 + (i * 0x60)
DMA4_CSSAi	RW	32	0x0000 009C + (i * 0x60)	0x4805 609C + (i * 0x60)
DMA4_CDSAi	RW	32	0x0000 00A0 + (i * 0x60)	0x4805 60A0 + (i * 0x60)
DMA4_CSEIi	RW	32	0x0000 00A4 + (i * 0x60)	0x4805 60A4 + (i * 0x60)
DMA4_CSFIi	RW	32	0x0000 00A8 + (i * 0x60)	0x4805 60A8 + (i * 0x60)
DMA4_CDEIi	RW	32	0x0000 00AC + (i * 0x60)	0x4805 60AC + (i * 0x60)
DMA4_CDFIi	RW	32	0x0000 00B0 + (i * 0x60)	0x4805 60B0 + (i * 0x60)
DMA4_CSACi	R	32	0x0000 00B4 + (i * 0x60)	0x4805 60B4 + (i * 0x60)
DMA4_CDACi	RW	32	0x0000 00B8 + (i * 0x60)	0x4805 60B8 + (i * 0x60)
DMA4_CCENi	R	32	0x0000 00BC + (i * 0x60)	0x4805 60BC + (i * 0x60)
DMA4_CCFNi	R	32	0x0000 00C0 + (i * 0x60)	0x4805 60C0 + (i * 0x60)
DMA4_COLORi	RW	32	0x0000 00C4 + (i * 0x60)	0x4805 60C4 + (i * 0x60)

7.7.3 SDMA Register Description

This section describes the registers used in the DMA4 controller.

NOTE: Some registers have no reset value (marked with -) because of hardware implementation in memory. Software must ensure the correct programming of these registers, if needed.

The shadow registers are used to read run time registers such as CCEN, CCFN, CDAC, or CSAC. Typically, when accessed in 8-bit or 16-bit access for two consecutive accesses, the value of the previous registers may change. This shadow register is used to hold the whole value to allow the next access to recover the remaining 24 bits or 16 bits.

The CSAC, CDAC, CCEN, and CCFN registers must be written or read in a way that enables the LSByte; otherwise, the shadow registers do not update the register.

There is no issue for 32-bit read-write transactions. For 16-bit transactions, read or write the LSByte first to enable the register update.

Table 7-13 through Table 7-67 describe the DMA register bits.

Table 7-13. DMA4_REVISION

Address Offset	0x0000 0000		
Physical Address	0x4805 6000	**Instance**	SDMA
Description	This register contains the DMA revision code		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000000
7:0	REV	[7:4] DMA4 major revision code [3:0] DMA4 minor revision code	R	TI internal data

Table 7-14. Register Call Summary for Register DMA4_REVISION

SDMA Registers Manual

- SDMA Register Summary: [0]

Table 7-15. DMA4_IRQSTATUS_Lj

Address Offset	0x0000 0008 + (j * 0x4)	**Index**	j = 0 to 3
Physical Address	0x4805 6008 + (j * 0x4)	**Instance**	SDMA
Description	The interrupt status register regroups all the status of the DMA4 channels that can generate an interrupt over line Lj.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH_31_0_Lj																															

Bits	Field Name	Description	Type	Reset
31:0	CH_31_0_Lj	Channel 31 Interrupt on Lj: When an interrupt is seen on the line Lj the status of a interrupting channel i is read in the bit field i. Read 0x0: Channel Interrupt Lj false Write 0x0: Channel Interrupt Lj status bit unchanged Read 0x1: Channel Interrupt Lj true (pending) Write 0x1: Channel Interrupt Lj status bit is reset	RW	0x00000000

Table 7-16. Register Call Summary for Register DMA4_IRQSTATUS_Lj

SDMA Module Integration • Interrupts to the MPU Subsystem: [0] [1] [2] [3]
SDMA Functional Description • Interrupt Generation: [4]
SDMA Registers Manual • SDMA Register Summary: [5]

Table 7-17. DMA4_IRQENABLE_Lj

Address Offset	0x0000 0018 + (j * 0x4)	**Index**	j = 0 to 3
Physical Address	0x4805 6018 + (j * 0x4)	**Instance**	SDMA
Description	The interrupt enable register allows to mask/unmask the module internal sources of interrupt, on line Lj		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH_31_0_Lj_EN																															

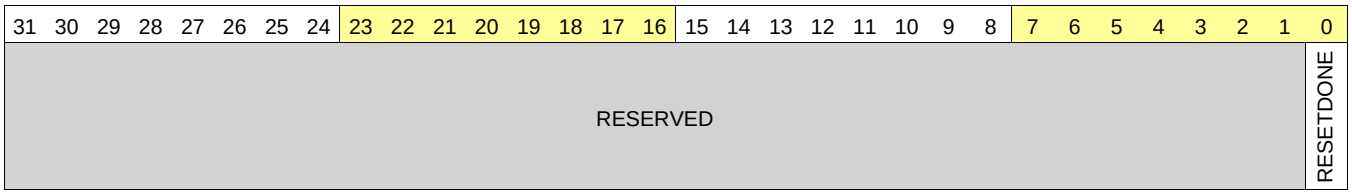
Bits	Field Name	Description	Type	Reset
31:0	CH_31_0_Lj_EN	Channel Interrupt on Lj mask/unmask : to Mask/Unmask a channel i interrupt on Lj the user writes 0/1 on the bit field i. 0x0: Channel Interrupt Lj is masked 0x1: Channel Interrupt Lj generates an interrupt when it occurs	RW	0x00000000

Table 7-18. Register Call Summary for Register DMA4_IRQENABLE_Lj

SDMA Module Integration • Interrupts to the MPU Subsystem: [0] [1] [2] [3]
SDMA Functional Description • Interrupt Generation: [4]
SDMA Basic Programming Model • Setup Configuration: [5]
SDMA Use Cases and Tips • Programming Flow: [6] • Programming Flow: [7]
SDMA Registers Manual • SDMA Register Summary: [8]

Table 7-19. DMA4_SYSSTATUS

Address Offset	0x0000 0028		
Physical Address	0x4805 6028	**Instance**	SDMA
Description	The register provides status information about the module excluding the interrupt status information (see interrupt status register)		
Type	R		



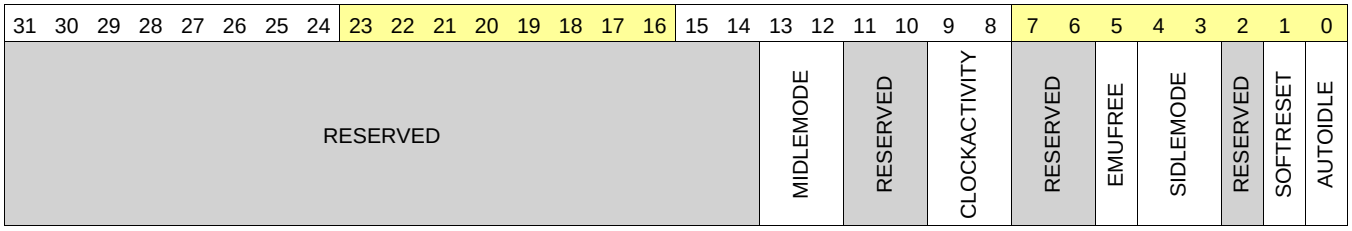
Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved for module-specific status information	RW	0x00000000
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is on-going 0x1: Reset completed	R	0x1

Table 7-20. Register Call Summary for Register DMA4_SYSSTATUS

SDMA Registers Manual
- SDMA Register Summary: [0]

Table 7-21. DMA4_OCP_SYSCONFIG

Address Offset	0x0000 002C		
Physical Address	0x4805 602C	**Instance**	SDMA
Description	This register controls the various parameters of the OCP interface		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Write 0s for future compatibility, Reads return 0	RW	0x00000
13:12	MIDLEMODE	Read write power management, standby/wait control 0x0: Force-standby: MStandby is asserted only when all the DMA channels are disabled 0x1: No-Standby: MStandby is never asserted 0x2: Smart-Standby: MStandby is asserted if at least one of the following two conditions is satisfied: 1. All the channels are disabled, OR 2. There is no non-synchronized channel enabled AND [if hardware synchronized channel is enabled, then no DMA request input is asserted and no requests are pending to be serviced.] 0x3: reserved for second smart-standby mode if needed	RW	0x0
11:10	RESERVED	Reserved for clocks activities extension	RW	0x0
9:8	CLOCKACTIVITY	Clocks activities during wake-up Bit 8: OCP interface clock 0 OCP clock can be switched-off Bit 9: Functional clock 0 Functional clock can be switched-off	R	0x0
7:6	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0

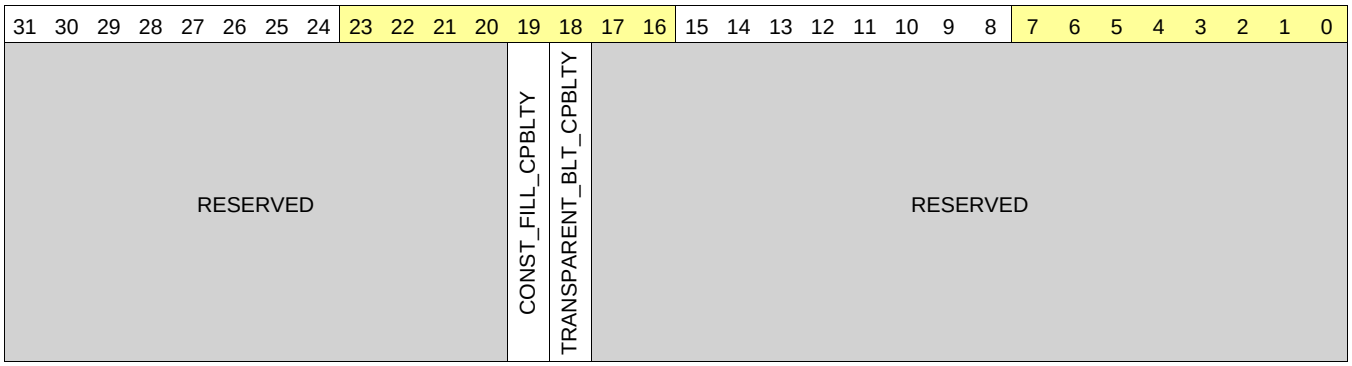
Bits	Field Name	Description	Type	Reset
5	EMUFREE	Enable sensitivity to MSuspend 0x0: DMA4 freezes its internal logic upon MSuspend assertion 0x1: DMA4 ignores the MSuspend input	RW	0x0
4:3	SIDLEMODE	Configuration port power management, Idle req/ack control 0x0: Force-idle. An idle request is acknowledged unconditionally 0x1: No-idle. An idle request is never acknowledged 0x2: Smart-idle. Idle acknowledge is given by DMA4 if all of the conditions ar true: 1. All the channels are disabled. 2. If hardware synchronized channel is enabled, then no DMA request input is asserted and no requests are pending to be serviced. 3. All transactions are completed on all the DMA ports. 4. No interrupts are pending to be serviced. 0x3: reserved - do not use.	RW	0x0
2	RESERVED	Write 0s for future compatibility, Reads return 0	RW	0x0
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. During reads, it always returns 0. 0x0: No effect 0x1: Reset	RW	0x0
0	AUTOIDLE	Internal OCP clock gating strategy 0x0: OCP clock is free running 0x1: Automatic OCP clock gating strategy is applied, based on the OCP interface activity.	RW	0x0

Table 7-22. Register Call Summary for Register DMA4_OCP_SYSCONFIG

SDMA Module Integration • Software Reset Through the Configuration Port: [0]
SDMA Functional Description • FIFO Draining Mechanism: [1] [2] [3] • Interconnect Clock Auto-Idle: [4] • Automatic Standby Mode: [5]
SDMA Registers Manual • SDMA Register Summary: [6]

Table 7-23. DMA4_CAPS_0

Address Offset	0x0000 0064		
Physical Address	0x4805 6064	**Instance**	SDMA
Description	DMA Capabilities Register 0 LSW		
Type	R		



Bits	Field Name	Description	Type	Reset
31:20	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000
19	CONST_FILL_CPBLTY	Constant_Fill_Capability 0x0: No LCH supports constant fill copy 0x1: any LCH supports constant fill copy	R	0x1
18	TRANSPARENT_BLT_CPBLTY	Transparent_BLT_Capability 0x0: No LCH supports transparent BLT copy 0x1: any LCH supports transparent BLT copy	R	0x1
17:0	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00000

Table 7-24. Register Call Summary for Register DMA4_CAPS_0

SDMA Registers Manual
- SDMA Register Summary: [0]

Table 7-25. DMA4_CAPS_2

Address Offset	0x0000 006C		
Physical Address	0x4805 606C	**Instance**	SDMA
Description	DMA Capabilities Register 2		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							SEPARATE_SRC_AND_DST_	DST_DOUBLE_INDEX_ADRS_	DST_SINGLE_INDEX_ADRS_	DST_POST_INCRMNT_ADRS_	DST_CONST_ADRS_CPBLTY	SRC_DOUBLE_INDEX_ADRS_	SRC_SINGLE_INDEX_ADRS_	SRC_POST_INCREMENT_ADRS_	SRC_CONST_ADRS_CPBLTY

Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000000
8	SEPARATE_SRC_AND_DST_INDEX_CPBLTY	Separate_source/destination_index_capability 0x0: Does not support separate src/dst index for 2D addressing 0x1: Supports separate src/dest index for 2D addressing	R	0x1
7	DST_DOUBLE_INDEX_ADRS_CPBLTY	Destination_double_index_address_capability 0x0: Does not support double index address mode on the destination port 0x1: Supports double index address mode on the destination port	R	0x1
6	DST_SINGLE_INDEX_ADRS_CPBLTY	Destination_single_index_address_capability 0x0: Does not support single index address mode on the destination port 0x1: Supports single index address mode on the destination port	R	0x1

Bits	Field Name	Description	Type	Reset
5	DST_POST_INCRMNT_ADRS_CPBLTY	Destination_post_increment_address_capability 0x0: Does not supports post-increment address mode in the destination port 0x1: Supports post-increment address mode in the destination port	R	0x1
4	DST_CONST_ADRS_CPBLTY	Destination_constant_address_capability 0x0: Does not supports constant address mode in the destination port 0x1: Supports constant address mode in the destination port	R	0x1
3	SRC_DOUBLE_INDEX_ADRS_CPBLTY	Source_double_index_address_capability 0x0: Does not support double index address mode on the source port 0x1: Supports double index address mode on the source port	R	0x1
2	SRC_SINGLE_INDEX_ADRS_CPBLTY	Source_single_index_address_capability 0x0: Does not support single index address mode on the source port 0x1: Supports single index address mode in the source port	R	0x1
1	SRC_POST_INCREMENT_ADRS_CPBLTY	Source_post_increment_address_capability 0x0: Does not supports post-increment address mode in the source port 0x1: Supports post-increment address mode in the source port	R	0x1
0	SRC_CONST_ADRS_CPBLTY	Source_constant_address_capability 0x0: Does not supports constant address mode in the source port 0x1: Supports constant address mode in the source port	R	0x1

Table 7-26. Register Call Summary for Register DMA4_CAPS_2

SDMA Registers Manual

- SDMA Register Summary: [0]

Table 7-27. DMA4_CAPS_3

Address Offset	0x0000 0070		
Physical Address	0x4805 6070	**Instance**	SDMA
Description	DMA Capabilities Register 3		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								BLOCK_SYNCHR_CPBLTY	PKT_SYNCHR_CPBLTY	CHANNEL_CHAINING_CPBLTY	CHANNEL_INTERLEAVE_CPBLTY	RESERVED		FRAME_SYNCHR_CPBLTY	ELMNT_SYNCHR_CPBLTY

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000000
7	BLOCK_SYNCHR_CPBLTY	Block_synchronization_capability 0x0: Does not support synchronization transfer on block boundary 0x1: Supports synchronization transfer on block boundary	R	0x1
6	PKT_SYNCHR_CPBLTY	Packet_synchronization_capability 0x0: Does not support synchronization transfer on packet boundary 0x1: Supports synchronization transfer on packet boundary	R	0x1
5	CHANNEL_CHAINING_CPBLTY	Channel_Chaining_capability 0x0: Does not support Channel Chaining capability 0x1: Supports Channel Chaining capability	R	0x1
4	CHANNEL_INTERLEAVE_CPBLTY	Channel_interleave_capability 0x0: Does not support Channel interleave capability 0x1: Supports Channel_interleave capability	R	0x1
3:2	RESERVED		RW	0x0
1	FRAME_SYNCHR_CPBLTY	Frame_synchronization_capability 0x0: Does not support synchronization transfer on Frame boundary 0x1: Supports synchronization transfer on Frame boundary	R	0x1
0	ELMNT_SYNCHR_CPBLTY	Element_synchronization_capability 0x0: Does not support synchronization transfer on Element boundary 0x1: Supports synchronization transfer on Element boundary	R	0x1

Table 7-28. Register Call Summary for Register DMA4_CAPS_3

SDMA Registers Manual
- SDMA Register Summary: [0]

Table 7-29. DMA4_CAPS_4

Address Offset	0x0000 0074		
Physical Address	0x4805 6074	**Instance**	SDMA
Description	DMA Capabilities Register 4		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		RESERVED	DRAIN_END_INTERRUPT_CPBLTY	MISALIGNED_ADRS_ERR_INTERRUPT_CPBLTY	SUPERVISOR_ERR_INTERRUPT_CPBLTY	RESERVED	TRANS_ERR_INTERRUPT_CPBLTY	PKT_INTERRUPT_CPBLTY	SYNC_STATUS_CPBLTY	BLOCK_INTERRUPT_CPBLTY	LAST_FRAME_INTERRUPT_	FRAME_INTERRUPT_CPBLTY	HALF_FRAME_INTERRUPT_	EVENT_DROP_INTERRUPT_	RESERVED

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000000
13	RESERVED	Reserved.	R	1
12	DRAIN_END_INTERRUPT_CPBLTY	Drain End detection capability.	R	1
11	MISALIGNED_ADRS_ERR_INTERRUPT_CPBLTY	Misaligned error detection capability.	R	1
10	SUPERVISOR_ERR_INTERRUPT_CPBLTY	Supervisor error detection capability.	R	1
9	RESERVED	Reserved for non-GP devices.	R	1
8	TRANS_ERR_INTERRUPT_CPBLTY	Transaction error detection capability.	R	1
7	PKT_INTERRUPT_CPBLTY	End of Packet detection capability. 0x0: Does not support end of packet interrupt generation capability 0x1: Supports end of packet interrupt generation capability	R	0x1
6	SYNC_STATUS_CPBLTY	Sync_status_capability 0x0: Does not support synchronized transfer status bit generation 0x1: Supports synchronized transfer status bit generation	R	0x1
5	BLOCK_INTERRUPT_CPBLTY	End of block detection capability. 0x0: Does not support end of block interrupt generation capability 0x1: Supports end of block interrupt generation capability	R	0x1
4	LAST_FRAME_INTERRUPT_CPBLTY	Start of last frame detection capability. 0x0: Does not support last frame interrupt generation capability 0x1: Supports last frame interrupt generation capability	R	0x1
3	FRAME_INTERRUPT_CPBLTY	End of frame detection capability. 0x0: Does not support end of frame interrupt generation capability 0x1: Supports end of frame interrupt generation capability	R	0x1
2	HALF_FRAME_INTERRUPT_CPBLTY	Detection capability of the half of frame end. 0x0: Does not support half of frame interrupt generation capability 0x1: Supports half of frame interrupt generation capability	R	0x1

Bits	Field Name	Description	Type	Reset
1	EVENT_DROP_INTERRUPT_CPBLTY	Request collision detection capability. 0x0: Does not support event drop interrupt generation capability 0x1: Supports event drop interrupt generation capability	R	0x1
0	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 7-30. Register Call Summary for Register DMA4_CAPS_4

SDMA Registers Manual
- SDMA Register Summary: [0]

Table 7-31. DMA4_GCR

Address Offset	0x0000 0078		
Physical Address	0x4805 6078	**Instance**	SDMA
Description	FIFO sharing between high and low priority channel. The Maximum per channel FIFO depth is bounded by the low and high channel FIFO budget. The high respectively low priority channels maximum burst size must be less than the min (high respectively low priority channel FIFO budget , per channel maximum FIFO depth)		
Type	RW		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14	13 12	11 10 9 8	7 6 5 4 3 2 1 0
RESERVED	ARBITRATION_RATE	HI_LO_FIFO_BUDGET	HI_THREAD_RESERVED	RESERVED	MAX_CHANNEL_FIFO_DEPTH

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00
23:16	ARBITRATION_RATE	Arbitration switching rate between prioritized and regular channel queues	RW	0x01
15:14	HI_LO_FIFO_BUDGET	Allow to have a separate Global FIFO budget for high and low priority channels. For Hi priority Channel: (Per_channel_Maximum FIFO depth + 1) x Number of active High priority Channel =< High Budget FIFO For Low priority channel: (Per_channel_Maximum FIFO depth + 1) x Number of active Low priority Channel =< Low Budget FIFO 0x0: no fixed budget for neither higher nor lower priority channel 0x1: 75% of FIFO for low priority and 25% for high priority channels 0x2: 25% of FIFO for low priority and 75% for high priority channels 0x3: 50% of FIFO for low priority and 50% for high priority channels	RW	0x0

Bits	Field Name	Description	Type	Reset
13:12	HI_THREAD_RESERVED	Allow thread reservation for high priority channel on both read and write ports. 0x0: No ThreadID is reserved on the Read Port for high priority channels. No ThreadID is reserved on the Write Port for high priority channels. 0x1: Read Port ThreadID 0 is reserved for high priority channels. Write Port ThreadID 0 is reserved for high priority channels. 0x2: Read port ThreadID 0 and ThreadID 1 are reserved for high priority channels. Write Port ThreadID 0 is reserved for high priority channels. 0x3: Read PortThreadID 0, ThreadID 1 and ThreadID 2 are reserved for high priority channels. Write Port ThreadID 0 is reserved for high priority channels.	RW	0x0
11:8	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
7:0	MAX_CHANNEL_FIFO_DEPTH	Maximum FIFO depth allocated to one logical channel. Maximum FIFO depth can not be 0x0. It should be at least 0x1 or greater. Note that If channel limit is less than destination burst size enough data will not be accumulated in the data FIFO and it will never be sent out on the WR port. The burst size should be less than the FIFO limit specified in this bit field.	RW	0x10

Table 7-32. Register Call Summary for Register DMA4_GCR

SDMA Functional Description
- Logical Channel Transfer Overview: [0] [1] [2]
- FIFO Queue Memory Pool: [3]
- Thread Budget Allocation: [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]
- FIFO Budget Allocation: [15] [16]

SDMA Basic Programming Model
- Setup Configuration: [17]
- Concurrent Software and Hardware Synchronization: [18] [19]

SDMA Registers Manual
- SDMA Register Summary: [20]

Table 7-33. DMA4_CCRi

Address Offset	0x0000 0080 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 6080 + (i* 0x60)	**Instance**	SDMA
Description	Channel Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					WRITE_PRIORITY	BUFFERING_DISABLE	SEL_SRC_DST_SYNC	PREFETCH	SUPERVISOR	RESERVED	SYNCHRO_CONTROL_UPPER		BS	TRANSPARENT_COPY_ENABLE	CONST_FILL_ENABLE	DST_AMODE		SRC_AMODE		RESERVED	WR_ACTIVE	RD_ACTIVE	SUSPEND_SENSITIVE	ENABLE	READ_PRIORITY	FS	SYNCHRO_CONTROL				

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00
26	WRITE_PRIORITY	Channel priority on the Write side 0x0: Channel has low priority on the Write side during the arbitration process 0x1: Channel has high priority on Write sided during the arbitration process	RW	0x0
25	BUFFERING_DISABLE	This bit allows to disable the default buffering functionality when transfer is source synchronized. 0x0: buffering is enable across element/packet when source is synchronized to element, packet, frame or blocks 0x1: buffering is disabled across element/packet when source is synchronized to element, packet, frame or blocks	RW	0x-
24	SEL_SRC_DST_SYNC	Specifies that element, packet, frame or block transfer (depending on CCR.bs and CCR.fs) is triggered by the source or the destination on the DMA request 0x0: Transfer is triggered by the destination. If synch on packet the packet element number is specified in the CDFI register 0x1: Transfer is triggered by the source. If synchronized on packet the packet element number is specified in the CSFI register	RW	0x-
23	PREFETCH	Enables the prefetch mode 0x0: Prefetch mode is disabled. When Sel_Src_Dst_Sync=1 transfers are buffered and pipelined between DMA requests 0x1: Prefetch mode is enabled. Prefetch mode is active only when destination is synchronized. It is SW user responsibility not to have at the same time Prefetch=1 when Sel_Src_Dst_Sync=1. This mode is not supported	RW	0x0
22	SUPERVISOR	Enables the supervisor mode 0x0: Supervisor mode is disabled 0x1: Supervisor mode is enabled	RW	0x0
21	RESERVED	Reserved for non-GP devices	RW	0x0
20:19	SYNCHRO_CONTROL_ UPPER	Channel Synchronization control upper (used in conjunction with the 5 bits of synchro channel DMA4_CCRi[4:0]) Used in conjunction, as two msb, with the five bits of the synchro channel bit field.	RW	0x0
18	BS	Block synchronization This bit used with the fs to see how the DMA request is serviced in a synchronized transfer	RW	0x-
17	TRANSPARENT_COPY_ ENABLE	Transparent copy enable 0x0: Transparent copy mode is disabled 0x1: Transparent copy mode is enabled	RW	0x-
16	CONST_FILL_ENABLE	Constant fill enable 0x0: Constant fill mode is disabled 0x1: Constant fill mode is enabled	RW	0x0
15:14	DST_AMODE	Selects the addressing mode on the Write Port of a channel. 0x0: Constant address mode 0x1: Post-incremented address mode 0x2: Single index address mode 0x3: Double index address mode	RW	0x-
13:12	SRC_AMODE	Selects the addressing mode on the Read Port of a channel. 0x0: Constant address mode 0x1: Post-incremented address mode 0x2: Single index address mode 0x3: Double index address mode	RW	0x-
11	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
10	WR_ACTIVE	Indicates if the channel write context is active or not 0x0: Channel is not active on the write port 0x1: Channel is active on the write port	R	0x0
9	RD_ACTIVE	Indicates if the channel read context is active or not 0x0: Channel is not active on the read port 0x1: Channel is currently active on the read port	R	0x0
8	SUSPEND_SENSITIVE	Logical channel suspend enable bit 0x0: The channel ignores the MSuspend even if EMUFree is set to 0. 0x1: If EMUFree is set to 0 and MSuspend comes in then all current OCP services (single transaction or burst transaction as specified in the corresponding CSDP register) have to be completed before stopping processing any more transactions	RW	0x0
7	ENABLE	Logical channel enable. It is SW responsibility to clear the CSR register and the IRQSTATUS bit for the different interrupt lines before enabling the channel. 0x0: The logical channel is disabled 0x1: The logical channel is enabled	RW	0x0
6	READ_PRIORITY	Channel priority on the read side 0x0: Channel has low priority on the Read side during the arbitration process 0x1: Channel has high priority on read sided during the arbitration process	RW	0x0
5	FS	Frame synchronization This bit used with the BS to see how the DMA request is serviced in a synchronized transfer FS=0 and BS=0: An element is transferred once a DMA request is made. FS=0 and BS=1: An entire block is transferred once a DMA request is made. FS=1 and BS=0: An entire frame is transferred once a DMA request is made. FS=1 and BS=1: A packet is transferred once a DMA request is made. All these different transfers can be interleaved on the port with other DMA requests.	RW	0x-
4:0	SYNCHRO_CONTROL	Channel synchronization control This bit field used with the second_level_synchro_control_upper (as two msb) 0000000 : Is reserved for non synchronized LCH transfer xxxxxxx (from 1 to 127) There are 127 possible DMA request to assign to any LCH. **Note:** The channel synchronization control registers are 1-based. For example, to enable the S_DMA_1 request, SYNCHRO_CONTROL bits (formed by DMA_CCRi[20:19] and DMA4_CCRi[4:0]) must be set to 0x2 (DMA request number + 1).	RW	0x0

Table 7-34. Register Call Summary for Register DMA4_CCRi

SDMA Functional Description

- Logical Channel Transfer Overview: [0] [1] [2]
- Addressing Modes: [3]
- Software Synchronization: [4] [5] [6]
- Hardware Synchronization: [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18]
- Thread Budget Allocation: [19] [20]
- Reprogramming an Active Channel: [21] [22] [23] [24]
- Interrupt Generation: [25]
- Packet Synchronization: [26] [27] [28] [29] [30]
- Graphics Acceleration Support: [31]
- Supervisor Modes: [32]
- Disabling a Channel During Transfer: [33]
- FIFO Draining Mechanism: [34] [35] [36] [37] [38] [39]
- Reset: [40]

SDMA Basic Programming Model

- Software-Triggered (Nonsynchronized) Transfer: [41] [42] [43] [44] [45] [46] [47] [48] [49]
- Hardware-Synchronized Transfer: [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85]
- Concurrent Software and Hardware Synchronization: [86] [87]
- Chained Transfer: [88]
- 90° Clockwise Image Rotation: [89] [90] [91] [92] [93] [94] [95]
- Graphic Operations: [96] [97] [98] [99]

SDMA Use Cases and Tips

- Programming Flow: [100] [101] [102] [103] [104] [105] [106] [107] [108] [109] [110] [111] [112] [113] [114] [115] [116] [117] [118] [119] [120] [121]
- Programming Flow: [122] [123] [124] [125] [126] [127] [128] [129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [140] [141] [142] [143]

SDMA Registers Manual

- SDMA Register Summary: [144]
- SDMA Register Description: [145] [146]

Table 7-35. DMA4_CLNK_CTRLi

Address Offset	0x0000 0084 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 6084 + (i* 0x60)	**Instance**	SDMA
Description	Channel Link Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																ENABLE_LNK	RESERVED							NEXTLCH_ID							

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	ENABLE_LNK	Enables or disable the channel linking. 0x0: Channel linking mode is disabled When set on the fly to 0 the current channel will complete the transfer and stops the chain linking 0x1: Channel linking mode is enabled. The logical channel defined in the NextLCH_ID is enabled at the end of the current transfer	RW	0x0
14:5	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000
4:0	NEXTLCH_ID	Defines the NextLCh_ID, which is used to build logical channel chaining queue.	RW	0x–

Table 7-36. Register Call Summary for Register DMA4_CLNK_CTRLi

SDMA Functional Description • Chained Logical Channel Transfers: [0] [1] • FIFO Draining Mechanism: [2]
SDMA Basic Programming Model • Chained Transfer: [3] [4] [5] [6] [7]
SDMA Use Cases and Tips • Programming Flow: [8]
SDMA Registers Manual • SDMA Register Summary: [9]

Table 7-37. DMA4_CICRi

Address Offset	0x0000 0088 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 6088 + (i* 0x60)	**Instance**	SDMA
Description	Channel Interrupt Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RESERVED			DRAIN_IE	MISALIGNED_ERR_IE	SUPERVISOR_ERR_IE	RESERVED	TRANS_ERR_IE	PKT_IE	RESERVED	BLOCK_IE	LAST_IE	FRAME_IE	HALF_IE	DROP_IE	RESERVED

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:13	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x-
12	DRAIN_IE	Enables the end of draining interrupt 0x0: Disables end of channel draining interrupt 0x1: Enable end of channel draining interrupt	RW	0x0
11	MISALIGNED_ERR_IE	Enables the address misaligned error event interrupt 0x0: Disables the misaligned address error event interrupt 0x1: Enables the misaligned address error event interrupt	RW	0x-
10	SUPERVISOR_ERR_IE	Enables the supervisor transaction error event interrupt 0x0: Disables the supervisor transaction error event interrupt 0x1: Enables the supervisor transaction error event interrupt	RW	0x1
9	RESERVED	Reserved for non-GP devices	RW	0x1
8	TRANS_ERR_IE	Enables the transaction error event interrupt 0x0: Disables the transaction error event interrupt 0x1: Enables the transaction error event interrupt	RW	0x-
7	PKT_IE	Enables the end of Packet interrupt 0x0: Disables the end of Packet transfer interrupt 0x1: Enables the end of Packet transfer interrupt	RW	0x-
6	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
5	BLOCK_IE	Enables the end of block interrupt 0x0: Disables the end of block interrupt 0x1: Enables the end of block interrupt	RW	0x-

Bits	Field Name	Description	Type	Reset
4	LAST_IE	Last frame interrupt enable (start of last frame) 0x0: Disables the last frame interrupt 0x1: Enables the last frame interrupt	RW	0x-
3	FRAME_IE	Frame interrupt enable (end of frame) 0x0: Disables the end of frame interrupt 0x1: Enables the end of frame interrupt	RW	0x-
2	HALF_IE	Enables or disables the half frame interrupt. 0x0: Disables the half frame interrupt 0x1: Enables the half frame interrupt	RW	0x-
1	DROP_IE	Synchronization event drop interrupt enable (request collision) 0x0: Disables the event drop interrupt 0x1: Enables the event drop interrupt	RW	0x0
0	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 7-38. Register Call Summary for Register DMA4_CICRi

SDMA Module Integration • Interrupts to the MPU Subsystem: [0] [1]
SDMA Functional Description • Interrupt Generation: [2] • FIFO Draining Mechanism: [3] • Reset: [4]
SDMA Basic Programming Model • Setup Configuration: [5]
SDMA Use Cases and Tips • Programming Flow: [6] [7] • Programming Flow: [8] [9]
SDMA Registers Manual • SDMA Register Summary: [10]

Table 7-39. DMA4_CSRi

Address Offset	0x0000 008C + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 608C + (i* 0x60)	**Instance**	SDMA
Description	Channel Status Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RESERVED			DRAIN_END	MISALIGNED_ADRS_ERR	SUPERVISOR_ERR	RESERVED	TRANS_ERR	PKT	SYNC	BLOCK	LAST	FRAME	HALF	DROP	RESERVED

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:13	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0
12	DRAIN_END	End of channel draining Read 0x0: Status bit unchanged Write 0x0: No drain end in the current transfer Read 0x1: The current channel draining is completed Write 0x1: Status bit is reset	RW	0x0
11	MISALIGNED_ ADRS_ERR	Misaligned address error event Read 0x0: No address error Write 0x0: Status bit unchanged Read 0x1: An address error has been occurred Write 0x1: Status bit is reset	RW	0x0
10	SUPERVISOR_ERR	Supervisor transaction error event Read 0x0: No supervisor transaction error Write 0x0: Status bit unchanged Read 0x1: A supervisor transaction error has been occurred Write0x1: Status bit is reset	RW	0x0
9	RESERVED	Reserved for non-GP devices	RW	0x0
8	TRANS_ERR	Transaction error event Read 0x0: No transaction error Write 0x0: Status bit unchanged Read 0x1: A transaction error has been occurred Write 0x1: Status bit is reset	RW	0x0
7	PKT	End of Packet transfer Read 0x0: The current packet transfer has not been finished Write 0x0: Status bit unchanged Read 0x1: The current packet has been transferred Write 0x1: Status bit is reset	RW	0x0
6	SYNC	Synchronization status of a channel. Read 0x0: Logical channel is not scheduled or servicing a non synchronized DMA request. Write 0x0: Status bit unchanged Read 0x1: Logical channel is servicing a synchronized DMA request Write 0x1: Status bit is reset	RW	0x0
5	BLOCK	End of block event Read 0x0: The current block transfer has not been finished Write 0x0: Status bit unchanged Read 0x1: The current block has been transferred Write 0x1: Status bit is reset	RW	0x0
4	LAST	Last frame (start of last frame) Read 0x0: The start of the last frame to transfer is not reached Write 0x0: Status bit unchanged Read 0x1: The start of the last frame to transfer is reached Write 0x1: Status bit is reset	RW	0x0
3	FRAME	End of frame event Read 0x0: The end of current transferred frame is not reached Write 0x0: Status bit unchanged Read 0x1: The end of current transferred frame is reached Write 0x1: Status bit is reset	RW	0x0

Bits	Field Name	Description	Type	Reset
2	HALF	Half of frame event. Read 0x0: The half of current transferred frame is not reached Write 0x0: Status bit unchanged Read 0x1: The half of current transferred frame is reached Write 0x1: Status bit is reset	RW	0x0
1	DROP	Synchronization event drop occurred during the transfer Read 0x0: No synchronization collision Write 0x0: Status bit unchanged Read 0x1: A synchronization collision has been occurred Write 0x1: Status bit is reset	RW	0x0
0	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0

Table 7-40. Register Call Summary for Register DMA4_CSRi

SDMA Module Integration • Interrupts to the MPU Subsystem: [0] [1] [2] [3] [4]
SDMA Functional Description • Interrupt Generation: [5] • FIFO Draining Mechanism: [6]
SDMA Basic Programming Model • Setup Configuration: [7]
SDMA Registers Manual • SDMA Register Summary: [8]

Table 7-41. DMA4_CSDPi

Address Offset	0x0000 0090 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 6090 + (i* 0x60)	**Instance**	SDMA
Description	Channel Source Destination Parameters		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										SRC_ENDIAN	SRC_ENDIAN_LOCK	DST_ENDIAN	DST_ENDIAN_LOCK	WRITE_MODE		DST_BURST_EN		DST_PACKED	RESERVED				SRC_BURST_EN		SRC_PACKED	RESERVED					DATA_TYPE	

Bits	Field Name	Description	Type	Reset
31:22	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x000
21	SRC_ENDIAN	Channel source endianness control 0x0: Source has Little Endian type 0x1: Source has Big Endian type	RW	0x-
20	SRC_ENDIAN_LOCK	Endianness Lock 0x0: Endianness adapt 0x1: Endianness lock	RW	0x-
19	DST_ENDIAN	Channel Destination endianness control 0x0: Destination has Little Endian type 0x1: Destination has Big Endian type	RW	0x-

Bits	Field Name	Description	Type	Reset
18	DST_ENDIAN_LOCK	Endianness Lock 0x0: Endianness adapt 0x1: Endianness lock	RW	0x-
17:16	WRITE_MODE	Used to enable writing mode without posting or with posting 0x0: Write nonposted (WRNP) 0x1: Write (Posted) 0x2: All transaction are mapped on the Write command as posted except for the last transaction in the transfer mapped on a Write nonposted 0x3: Undefined	RW	0x-
15:14	DST_BURST_EN	Used to enable bursting on the Write Port. Smaller burst size than the programmed burst size is also allowed 0x0: Single access 0x1: 16 bytes or 4x32-bit/2x64-bit burst access 0x2: 32 bytes or 8x32-bit/4x64-bit burst access 0x3: 64 bytes or 16x32-bit/8x64-bit burst access	RW	0x0
13	DST_PACKED	Destination receives packed data. 0x0: The destination target is nonpacked 0x1: The destination target is packed	RW	0x-
12:9	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x-
8:7	SRC_BURST_EN	Used to enable bursting on the Read Port. Smaller burst size than the programmed burst size is also allowed 0x0: Single access 0x1: 16 bytes or 4x32-bit/2x64-bit burst access 0x2: 32 bytes or 8x32-bit/4x64-bit burst access 0x3: 64 bytes or 16x32-bit/8x64-bit burst access	RW	0x-
6	SRC_PACKED	Source provides packed data. 0x0: The source target is nonpacked 0x1: The source target is packed	RW	0x-
5:2	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x-
1:0	DATA_TYPE	Defines the type of the data moved in the channel. 0x0: 8 bits scalar 0x1: 16 bits scalar 0x2: 32 bits scalar 0x3: Undefined	RW	0x-

Table 7-42. Register Call Summary for Register DMA4_CSDPi

SDMA Functional Description
- Addressing Modes: [0]
- Packed Accesses: [1]
- Burst Transactions: [2]
- Endianism Conversion: [3] [4]
- Hardware Synchronization: [5]
- Graphics Acceleration Support: [6]
- Posted and Nonposted Writes: [7]

SDMA Basic Programming Model
- Software-Triggered (Nonsynchronized) Transfer: [8] [9] [10] [11] [12] [13] [14] [15] [16]
- Hardware-Synchronized Transfer: [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28]
- 90° Clockwise Image Rotation: [29] [30] [31] [32] [33] [34] [35] [36] [37]

SDMA Use Cases and Tips
- Programming Flow: [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51]
- Programming Flow: [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65]

Table 7-42. Register Call Summary for Register DMA4_CSDPi (continued)

SDMA Registers Manual • SDMA Register Summary: [66]

Table 7-43. DMA4_CENi

Address Offset	0x0000 0094 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 6094 + (i* 0x60)	**Instance**	SDMA
Description	Channel Element Number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CHANNEL_ELMNT_NBR																									

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00
23:0	CHANNEL_ELMNT_NBR	Number of elements within a frame (unsigned) to transfer	RW	0x------

Table 7-44. Register Call Summary for Register DMA4_CENi

SDMA Functional Description • Addressing Modes: [0] [1]
SDMA Basic Programming Model • Software-Triggered (Nonsynchronized) Transfer: [2] • Hardware-Synchronized Transfer: [3] [4] [5] • 90° Clockwise Image Rotation: [6]
SDMA Use Cases and Tips • Programming Flow: [7] [8] • Programming Flow: [9] [10]
SDMA Registers Manual • SDMA Register Summary: [11]

Table 7-45. DMA4_CFNi

Address Offset	0x0000 0098 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 6098 + (i* 0x60)	**Instance**	SDMA
Description	Channel Frame Number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CHANNEL_FRAME_NBR															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	CHANNEL_FRAME_NBR	Number of frames within the block to be transferred (unsigned)	RW	0x----

Table 7-46. Register Call Summary for Register DMA4_CFNi

SDMA Functional Description • Addressing Modes: [0] [1]
SDMA Basic Programming Model • Software-Triggered (Nonsynchronized) Transfer: [2] • Hardware-Synchronized Transfer: [3] [4] • 90° Clockwise Image Rotation: [5]
SDMA Use Cases and Tips • Programming Flow: [6] [7] • Programming Flow: [8] [9]
SDMA Registers Manual • SDMA Register Summary: [10]

Table 7-47. DMA4_CSSAi

Address Offset	0x0000 009C + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 609C + (i* 0x60)	**Instance**	SDMA
Description	Channel Source Start Address		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_START_ADRS																															

Bits	Field Name	Description	Type	Reset
31:0	SRC_START_ADRS	32 bits of the source start address	RW	0x--------

Table 7-48. Register Call Summary for Register DMA4_CSSAi

SDMA Functional Description • Addressing Modes: [0]
SDMA Basic Programming Model • Software-Triggered (Nonsynchronized) Transfer: [1] • Hardware-Synchronized Transfer: [2] [3] [4] [5] • 90° Clockwise Image Rotation: [6]
SDMA Use Cases and Tips • Programming Flow: [7] [8] • Programming Flow: [9] [10]
SDMA Registers Manual • SDMA Register Summary: [11]

Table 7-49. DMA4_CDSAi

Address Offset	0x0000 00A0 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60A0 + (i* 0x60)	**Instance**	SDMA
Description	Channel Destination Start Address		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_START_ADRS																															

Bits	Field Name	Description	Type	Reset
31:0	DST_START_ADRS	32 bits of the destination start address	RW	0x--------

Table 7-50. Register Call Summary for Register DMA4_CDSAi

SDMA Functional Description • Addressing Modes: [0]
SDMA Basic Programming Model • Software-Triggered (Nonsynchronized) Transfer: [1] • Hardware-Synchronized Transfer: [2] [3] [4] [5] • 90° Clockwise Image Rotation: [6]
SDMA Use Cases and Tips • Programming Flow: [7] [8] • Programming Flow: [9] [10]
SDMA Registers Manual • SDMA Register Summary: [11]

Table 7-51. DMA4_CSEIi

Address Offset	0x0000 00A4 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60A4 + (i* 0x60)	**Instance**	SDMA
Description	Channel Source Element Index (Signed)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CHANNEL_SRC_ELMNT_INDEX															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	CHANNEL_SRC_ ELMNT_INDEX	Channel source element index	RW	0x----

Table 7-52. Register Call Summary for Register DMA4_CSEIi

SDMA Registers Manual • SDMA Register Summary: [0]

Table 7-53. DMA4_CSFIi

Address Offset	0x0000 00A8 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60A8 + (i* 0x60)	**Instance**	SDMA
Description	Channel Source Frame Index (Signed) or 16-bit Packet size		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH_SRC_FRM_INDEX_OR_																															

Bits	Field Name	Description	Type	Reset
31:0	CH_SRC_FRM_INDEX_OR_ 16BIT_PKT_ELNT_NBR	Channel source frame index value if source address is in double index mode. Or if fs=bs=1 and DMA_CCR[SEL_SRC_DST_SYNC]=1; the bit field [15:0] gives the number of element in packet. The field [31:16] is unused for the packet size.	RW	0x--------

Table 7-54. Register Call Summary for Register DMA4_CSFIi

SDMA Functional Description • Packet Synchronization: [0] [1] [2] [3] [4]
SDMA Basic Programming Model • Hardware-Synchronized Transfer: [5] [6]
SDMA Registers Manual • SDMA Register Summary: [7]

Table 7-55. DMA4_CDEIi

Address Offset	0x0000 00AC + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60AC + (i* 0x60)	**Instance**	SDMA
Description	Channel Destination Element Index (Signed)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CHANNEL_DST_ELMNT_INDEX															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	CHANNEL_DST_ELMNT_INDEX	Channel destination element index	RW	0x----

Table 7-56. Register Call Summary for Register DMA4_CDEIi

SDMA Registers Manual • SDMA Register Summary: [0]

Table 7-57. DMA4_CDFIi

Address Offset	0x0000 00B0 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60B0 + (i* 0x60)	**Instance**	SDMA
Description	Channel Destination Frame Index (Signed) or 16-bit Packet size		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH_DST_FRM_IDX_OR_16BIT_																															

Bits	Field Name	Description	Type	Reset
31:0	CH_DST_FRM_IDX_OR_ 16BIT_PKT_ELNT_NBR	Channel destination frame index value if destination address is in double index mode. Or if fs=bs=1 and DMA_CCR[SEL_SRC_DST_SYNC]=0; the bit field [15:0] gives the number of element in packet. The field [31:16] is unused for the packet size..	RW	0x--------

Table 7-58. Register Call Summary for Register DMA4_CDFIi

SDMA Functional Description • Packet Synchronization: [0] [1] [2] [3]
SDMA Basic Programming Model • Hardware-Synchronized Transfer: [4] [5]
SDMA Registers Manual • SDMA Register Summary: [6]

Table 7-59. DMA4_CSACi

Address Offset	0x0000 00B4 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60B4 + (i* 0x60)	**Instance**	SDMA
Description	Channel Source Address Value. User has to access this register only in 32-bit access. If accessed in 8-bit or 16bit data may be corrupted.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_ELMNT_ADRS																															

Bits	Field Name	Description	Type	Reset
31:0	SRC_ELMNT_ADRS	Current source address counter value	R	0x--------

Table 7-60. Register Call Summary for Register DMA4_CSACi

SDMA Registers Manual
- SDMA Register Summary: [0]

Table 7-61. DMA4_CDACi

Address Offset	0x0000 00B8 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60B8 + (i* 0x60)	**Instance**	SDMA
Description	Channel Destination Address Value. User has to access this register only in 32-bit access. If accessed in 8-bit or 16bit data may be corrupted.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DST_ELMNT_ADRS																															

Bits	Field Name	Description	Type	Reset
31:0	DST_ELMNT_ADRS	Current destination address counter value	RW	0x--------

Table 7-62. Register Call Summary for Register DMA4_CDACi

SDMA Functional Description
- Hardware Synchronization: [0]

SDMA Basic Programming Model
- Hardware-Synchronized Transfer: [1]
- Synchronized Transfer Monitoring Using CDAC: [2] [3] [4] [5] [6] [7] [8]

SDMA Registers Manual
- SDMA Register Summary: [9]

Table 7-63. DMA4_CCENi

Address Offset	0x0000 00BC + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60BC + (i* 0x60)	**Instance**	SDMA
Description	Channel Current Transferred Element Number in the current frame. User has to access this register only in 32-bit access. If accessed in 8-bit or 16bit data may be corrupted.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CURRENT_ELMNT_NBR																									

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x00
23:0	CURRENT_ELMNT_NBR	Channel current transferred element number in the current frame	R	0x------

Table 7-64. Register Call Summary for Register DMA4_CCENi

SDMA Basic Programming Model • Synchronized Transfer Monitoring Using CDAC: [0]
SDMA Registers Manual • SDMA Register Summary: [1]

Table 7-65. DMA4_CCFNi

Address Offset	0x0000 00C0 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60C0 + (i* 0x60)	**Instance**	SDMA
Description	Channel Current Transferred Frame Number in the current transfer. User has to access this register only in 32-bit access. If accessed in 8-bit or 16bit data may be corrupted.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CURRENT_FRAME_NBR															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	CURRENT_FRAME_NBR	Channel current transferred frame number in the current transfer	R	0x----

Table 7-66. Register Call Summary for Register DMA4_CCFNi

SDMA Basic Programming Model • Synchronized Transfer Monitoring Using CDAC: [0]
SDMA Registers Manual • SDMA Register Summary: [1]

Table 7-67. DMA4_COLORi

Address Offset	0x0000 00C4 + (i* 0x60)	**Index**	i = 0 to 31
Physical Address	0x4805 60C4 + (i* 0x60)	**Instance**	SDMA
Description	Channel DMA COLOR KEY /SOLID COLOR		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CH_BLT_FRGRND_COLOR_																							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Reserved. Write 0s for future compatibility. Read returns 0.	RW	0x–
23:0	CH_BLT_FRGRND_COLOR_ OR_SOLID_COLOR_PTRN	Color key or solid color pattern: The pattern is replicated according to the data type. If the data-type is 8-bit the pattern is replicated 4 times to fill the register in order to enhance processing when data is packed at the graphic module input. The same reasoning for 16-bit data-type.	RW	0x------

Table 7-68. Register Call Summary for Register DMA4_COLORi

SDMA Functional Description • Graphics Acceleration Support: [0] [1]
SDMA Basic Programming Model • Graphic Operations: [2] [3]
SDMA Registers Manual • SDMA Register Summary: [4]

Interrupt Controller (INTC)

This chapter describes in detail the MPU subsystem interrupt controller (MPU_INTC) module used in the stand-alone processor.

NOTE: This chapter gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. Ensure that interrupts of unavailable modules and features are masked in MPU subsystem.

Topic **Page**

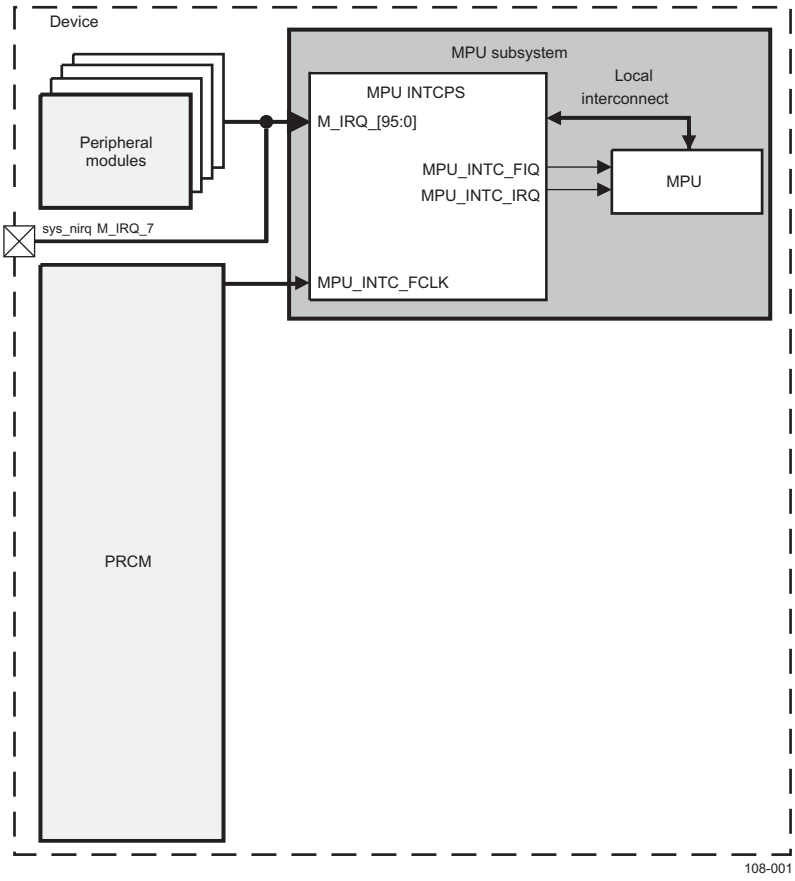
8.1 Interrupt Controller Overview

The device has one interrupt controller (INTC) module, the MPU subsystem INTC (INTCPS). This module handles all MPU-related events, using Priority Threshold and Security Features. It communicates with the ARM Cortex-A8™ processor using a private local interconnect, and runs at half the speed of the processor.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

Figure 8-1 shows the internal interrupt scheme.

Figure 8-1. Interrupt Controller Highlight



8.2 Interrupt Controller Environment

The INTC can handle two types of interrupts originating from an external device:

- sys_nirq interrupt inputs:

  The MPU INTC handles external interrupts through a dedicated sys_nirq interrupt line that connects the INTC module with a TPS65950 power IC. An interrupt can generate a system wake-up event.

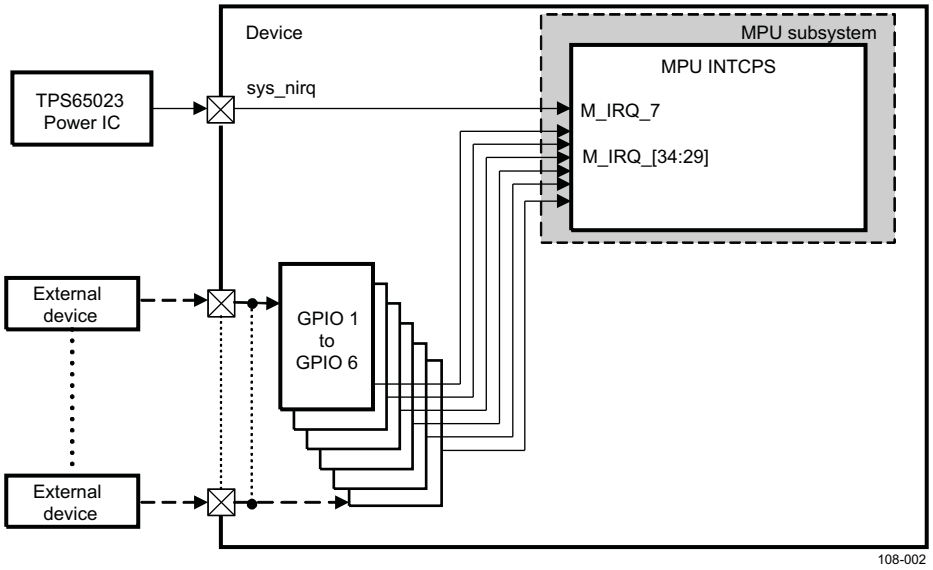
  If the system is idle and the external interrupt is masked, the interrupt cannot wake up the system. Like other interrupt lines, the external interrupt is active at low level and is acknowledged by the software according to the common programming model.

- GPIO interrupt inputs:

  External devices can also use GPIO modules to generate interrupts to the MPU. There are six dedicated interrupt lines to the MPU INTC. One interrupt line is associated with each GPIO module. Each GPIO module can generate a single interrupt whenever there is at least one event in any one of the configured 32 GPIO inputs. For more information about GPIO features, see the *General-Purpose Interface* chapter.

Figure 8-2 shows the relationship between the device and external interrupts.

Figure 8-2. Interrupts from External Devices



The features specific to INTCPS are:

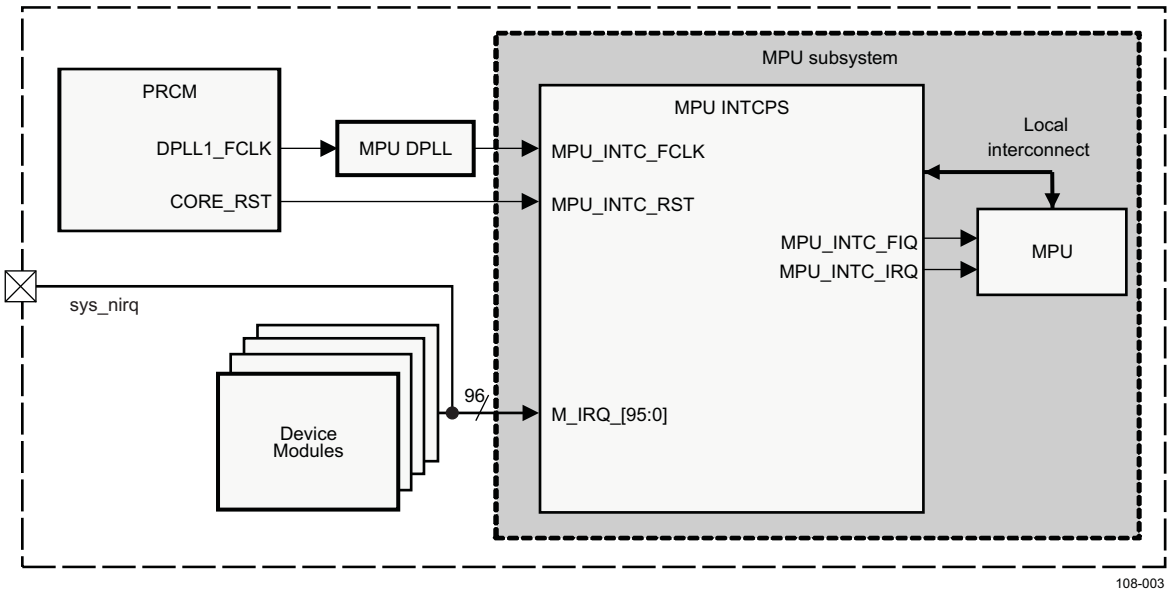
- Up to 96 level-sensitive interrupt inputs
- Individual priority (up to 64) for each interrupt input
- Interrupt lines connected to internal module interrupts
- One incoming interrupt line from an external device

8.3 MPU Subsystem INTCPS Integration

The INTCPS module is the interface between incoming interrupts and the two interrupt inputs of the MPU. It can handle up to 96 request inputs that can be configured as MPU FIQ or IRQ interrupt requests.

Figure 8-3 shows the integration of the INTCPS in the MPU subsystem.

Figure 8-3. MPU Subsystem INTCPS Integration



The MPU subsystem INTCPS is directly connected to the MPU by an MPU peripheral port. Consequently, the MPU subsystem INTCPS is accessible and visible only by the MPU.

8.3.1 Clocking, Reset, and Power Management Scheme

8.3.1.1 MPU Subsystem INTC Clocks

The MPU subsystem INTCPS runs at half the rate of the MPU functional clock (see the *MPU Subsystem* chapter).

The interface clock used for register access runs at the rate of the interconnect bus clock (equal to the rate of the MPU interface clock; see the *Power, Reset, and Clock Management* chapter).

The synchronizer clock allows external asynchronous interrupts to be resynchronized before they are masked.

Table 8-1 lists the MPU subsystem INTC clock rates.

Table 8-1. MPU Subsystem INTC Clock Rates

Clock	Frequency	Name	Comments
Functional	ARM_FCLK	MPU_INTC_FCLK	Source is the MPU DPLL.
Interface	ARM_FCLK	MPU_INTC_ICLK	Source is the PRCM module.
Synchronizer	MPU_INTC_FCLK	Synchronizer clock (module internal clock)	Source is the MPU_INTC_FCLK.

8.3.1.2 Hardware and Software Reset

Table 8-2 lists the MPU subsystem INTC resets.

Table 8-2. Hardware and Software Reset

Type	Name	Source	Activation	Domain
Hardware	CORE_RST	PRCM	Active low	CORE
Software	SOFTRESET	MPU_INTC.INTCPS_SYSCONFIG[1] SOFTRESET bit	Active at 1	MPU INTC internal

8.3.1.3 Power Management

The MPU subsystem INTC belongs to the CORE power domain. As part of CORE power domain, it is sensitive to a CORE_RST issued by the PRCM. For more information about the CORE power domain implementation and CORE_RST signal, see the *Power, Reset, and Clock Management* chapter.

The MPU INTC clocks come from the MPU DPLL. For more information about these clocks control, see the *MPU Subsystem* chapter.

8.3.2 Interrupt Request Lines

Table 8-3 lists the incoming and outgoing interrupt lines of the INTCPS.

Table 8-3. Interrupt Lines Incoming and Outgoing

Type	Number	Name	Mapping	Comments
Interrupt request inputs	Up to 96	M_IRQ_[95:0]	See Table 8-4	Inputs to INTCPS module, source from various modules.
Interrupt request outputs	2	MPU_INTC_FIQ	MPU_INTC_FIQ	Outgoing to MPU Fast Interrupt
		MPU_INTC_IRQ	MPU_INTC_IRQ	Outgoing to MPU Normal Interrupt

NOTE: Interrupt request signals are active at low level.

This section gives information about all modules and features in the high-tier device. See Chapter 1, *Device Family* section, to check availability of modules and features. Ensure that interrupts of unavailable modules and features are masked in MPU subsystem.

Table 8-4 lists interrupt mappings to the MPU subsystem.

Table 8-4. Interrupt Mapping to the MPU Subsystem(1)

IRQ	Source	Description
M_IRQ_0	EMUINT	MPU emulation(2)
M_IRQ_1	COMMTX	MPU emulation(2)
M_IRQ_2	COMMRX	MPU emulation(2)
M_IRQ_3	BENCH	MPU emulation(2)
M_IRQ_4	MCBSP2_ST_IRQ	Sidetone MCBSP2 overflow
M_IRQ_5	MCBSP3_ST_IRQ	Sidetone MCBSP3 overflow
M_IRQ_6	Reserved	Reserved
M_IRQ_7	sys_nirq	External source (active low)
M_IRQ_8	Reserved	Reserved
M_IRQ_9	SMX_DBG_IRQ	L3 Interconnect error for debug
M_IRQ_10	SMX_APP_IRQ	L3 Interconnect error for application
M_IRQ_11	PRCM_MPU_IRQ	PRCM module IRQ
M_IRQ_12	SDMA_IRQ_0	System DMA request 0
M_IRQ_13	SDMA_IRQ_1	System DMA request 1
M_IRQ_14	SDMA_IRQ_2	System DMA request 2

(1) All the IRQ signals are active at low level.
(2) These interrupts are internally generated within the MPU subsystem.

Table 8-4. Interrupt Mapping to the MPU Subsystem[(1)] (continued)

IRQ	Source	Description
M_IRQ_15	SDMA_IRQ_3	System DMA request 3
M_IRQ_16	MCBSP1_IRQ	McBSP module 1 IRQ
M_IRQ_17	MCBSP2_IRQ	McBSP module 2 IRQ
M_IRQ_18	SR1_IRQ	SmartReflex™ 1
M_IRQ_19	Reserved	Reserved
M_IRQ_20	GPMC_IRQ	General-purpose memory controller module
M_IRQ_21	SGX_IRQ	2D/3D graphics module
M_IRQ_22	MCBSP3_IRQ	McBSP module 3
M_IRQ_23	MCBSP4_IRQ	McBSP module 4
M_IRQ_24	HECC0INT	High End CAN controller interrupt line 0 (Level)
M_IRQ_25	DSS_IRQ	Display subsystem module
M_IRQ_26	RESERVED	Reserved
M_IRQ_27	MCBSP5_IRQ	McBSP module 5
M_IRQ_28	HECC1INT	High End CAN controller interrupt line 1 (Level)
M_IRQ_29	GPIO1_MPU_IRQ	GPIO module 1
M_IRQ_30	GPIO2_MPU_IRQ	GPIO module 2
M_IRQ_31	GPIO3_MPU_IRQ	GPIO module 3
M_IRQ_32	GPIO4_MPU_IRQ	GPIO module 4
M_IRQ_33	GPIO5_MPU_IRQ	GPIO module 5
M_IRQ_34	GPIO6_MPU_IRQ	GPIO module 6
M_IRQ_35	Reserved	Reserved
M_IRQ_36	WDT3_IRQ	Watchdog timer module 3 overflow
M_IRQ_37	GPT1_IRQ	General-purpose timer module 1
M_IRQ_38	GPT2_IRQ	General-purpose timer module 2
M_IRQ_39	GPT3_IRQ	General-purpose timer module 3
M_IRQ_40	GPT4_IRQ	General-purpose timer module 4
M_IRQ_41	GPT5_IRQ	General-purpose timer module 5
M_IRQ_42	GPT6_IRQ	General-purpose timer module 6
M_IRQ_43	GPT7_IRQ	General-purpose timer module 7
M_IRQ_44	GPT8_IRQ	General-purpose timer module 8
M_IRQ_45	GPT9_IRQ	General-purpose timer module 9
M_IRQ_46	GPT10_IRQ	General-purpose timer module 10
M_IRQ_47	GPT11_IRQ	General-purpose timer module 11
M_IRQ_48	SPI4_IRQ	McSPI module 4
M_IRQ_49	SHA1MD5_IRQ2	SHA1/MD5 crypto-accelerator 2
M_IRQ_50	FPKA_IRQREADY_N	PKA crypto-accelerator
M_IRQ_51	SHA2MD5_IRQ	SHA2/MD5 crypto-accelerator 1
M_IRQ_52	RNG_IRQ	RNG module
M_IRQ_53	EMIF4ERR	Active High Level Interrupt
M_IRQ_54	MCBSP4_IRQ_TX	McBSP module 4 transmit
M_IRQ_55	MCBSP4_IRQ_RX	McBSP module 4 receive
M_IRQ_56	I2C1_IRQ	I^2C module 1
M_IRQ_57	I2C2_IRQ	I^2C module 2
M_IRQ_58	HDQ_IRQ	HDQ™/One-wire™
M_IRQ_59	McBSP1_IRQ_TX	McBSP module 1 transmit
M_IRQ_60	McBSP1_IRQ_RX	McBSP module 1 receive
M_IRQ_61	I2C3_IRQ	I^2C module 3

Table 8-4. Interrupt Mapping to the MPU Subsystem[(1)] (continued)

IRQ	Source	Description
M_IRQ_62	McBSP2_IRQ_TX	McBSP module 2 transmit
M_IRQ_63	McBSP2_IRQ_RX	McBSP module 2 receive
M_IRQ_64	FPKA_IRQRERROR_N	PKA crypto-accelerator
M_IRQ_65	SPI1_IRQ	McSPI module 1
M_IRQ_66	SPI2_IRQ	McSPI module 2
M_IRQ_67	C0_RX_THRESH_PULSE	EMAC Interrupt 0 (Active high pulse)
M_IRQ_68	C0_RX_PULSE	EMAC Interrupt 1 (Active high pulse)
M_IRQ_69	C0_TX_PULSE	EMAC Interrupt 2 (Active high pulse)
M_IRQ_70	C0_MISC_PULSE	EMAC Interrupt 3 (Active high pulse)
M_IRQ_71	USBOTG_INT	USBOTG interrupt request
M_IRQ_72	UART1_IRQ	UART module 1
M_IRQ_73	UART2_IRQ	UART module 2
M_IRQ_74	UART3_IRQ	UART module 3 (also infrared)
M_IRQ_75	PBIAS_IRQ	Merged interrupt for PBIASlite1 and 2
M_IRQ_76	OHCI_IRQ	OHCI controller HSUSB MP Host Interrupt
M_IRQ_77	EHCI_IRQ	EHCI controller HSUSB MP Host Interrupt
M_IRQ_78	TLL_IRQ	HSUSB MP TLL Interrupt
M_IRQ_79	PARTHASH_IRQ	SHA2/MD5 crypto-accelerator 1
M_IRQ_80	Reserved	Reserved
M_IRQ_81	MCBSP5_IRQ_TX	McBSP module 5 transmit
M_IRQ_82	MCBSP5_IRQ_RX	McBSP module 5 receive
M_IRQ_83	MMC1_IRQ	MMC/SD module 1
M_IRQ_84	UART4	UART4 module, active low level interrupt
M_IRQ_85	Reserved	Reserved
M_IRQ_86	MMC2_IRQ	MMC/SD module 2
M_IRQ_87	MPU_ICR_IRQ	MPU ICR
M_IRQ_88	CCDC_VD0_INT	VPFE interrupt 0 active high pulse
M_IRQ_89	MCBSP3_IRQ_TX	McBSP module 3 transmit
M_IRQ_90	MCBSP3_IRQ_RX	McBSP module 3 receive
M_IRQ_91	SPI3_IRQ	McSPI module 3
M_IRQ_92	CCDC_VD1_INT	VPFE interrupt 1 active high pulse
M_IRQ_93	CCDC_VD2_INT	VPFE interrupt 2 active high pulse
M_IRQ_94	MMC3_IRQ	MMC/SD module 3
M_IRQ_95	GPT12_IRQ	General purpose timer module 12

8.4 Interrupt Controller Functional Description

The main features of the INTCPS are:

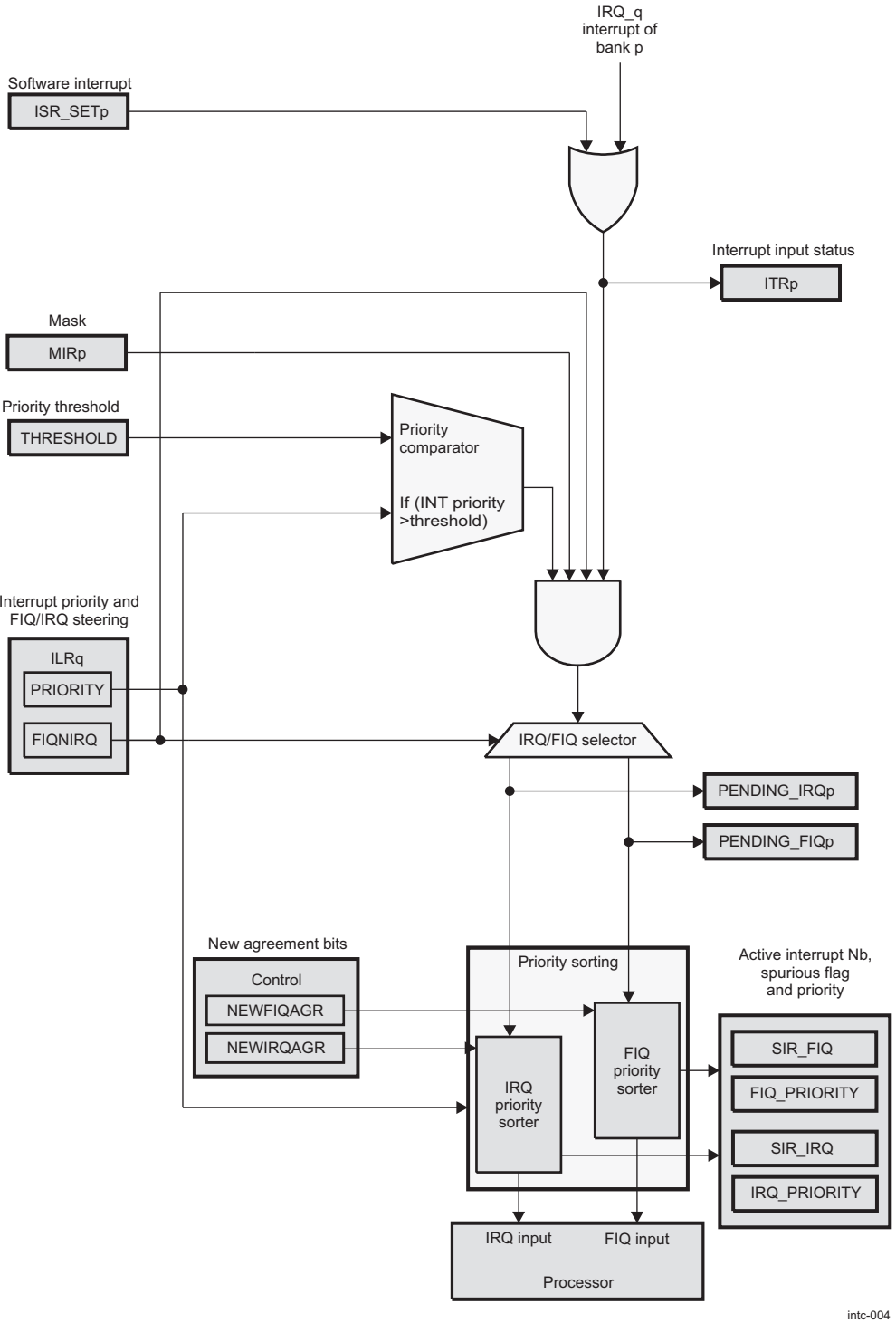
- Individual priority (up to 64 levels) for each interrupt input
- Ability for each interrupt to be steered to either FIQ or IRQ
- Independent priority sorting for FIQ and IRQ; FIQ sorting is processed concurrently with IRQ sorting.
- Priority masking: Interrupts can be masked based on the priority threshold register.
- Atomic bit set and clear capability for interrupt mask and software interrupt registers
- For high-security (HS) devices only:
  - Global interrupt mask
  - Protection for secure interrupts
- Power-management and wake-up support

- Auto-idle power-saving support

The INTCPS processes incoming interrupts by masking and priority sorting, then it generates the interrupt requests to the MPU.

Figure 8-4 shows the top-level view of the interrupt processing.

Figure 8-4. Top-Level Block Diagram



(1) Public Mask = SICR[3] PUBLICINHIBIT bit OR (SICR[2] AUTOINHIBIT bit AND SICR[0] SSMFIQSTATUS bit)

(2) The SCRn forces the interrupt priority to 0 (highest).

(3) The SCRn forces the interrupt to FIQ.

(4) The comparator output is TRUE only if the interrupt priority is higher than the threshold register priority. The highest priority is 0x0; the lowest priority is 0x3F (63).

8.4.1 Interrupt Processing

8.4.1.1 Input Selection

The INTCPS supports only level-sensitive incoming interrupt detection. A peripheral asserting an interrupt maintains it until software has handled the interrupt and instructed the peripheral to de-assert the interrupt.

A software interrupt is generated if the corresponding bit in the MPU_INTC.INTCPS_ISR_SETn register is set (register bank number: n = [0,2] for the MPU subsystem INTCPS, 96 incoming interrupt lines are supported). The software interrupt clears when the corresponding bit in the MPU_INTC.INTCPS_ISR_CLEARn register is written. Typical use of this feature is software debugging.

8.4.1.2 Masking

8.4.1.2.1 Individual Masking

Detection of interrupts on each incoming interrupt line can be enabled or disabled independently by the MPU_INTC.INTCPS_MIRn interrupt mask register. In response to an unmasked incoming interrupt, the INTCPS can generate one of two types of interrupt requests to the processor:

- IRQ: low-priority interrupt request
- FIQ: fast interrupt request

The type of interrupt request is determined by the MPU_INTC.INTCPS_ILRm[0] FIQNIRQ bit (m= [0,95]).

The current incoming interrupt status before masking is readable from the MPU_INTC.INTCPS_ITRn register. After masking and IRQ/FIQ selection, and before priority sorting is done, the interrupt status is readable from the MPU_INTC.INTCPS_PENDING_IRQn and MPU_INTC.INTCPS_PENDING_FIQn registers.

8.4.1.2.2 Global Masking (HS Devices Only)

Software may use global masking for security purposes, but this feature must be used with care. To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

8.4.1.2.3 Priority Masking

To enable faster processing of high-priority interrupts, a programmable priority masking threshold is provided (the MPU_INTC.INTCPS_THRESHOLD[7:0] PRIORITYTHRESHOLD field). This priority threshold allows preemption by higher priority interrupts; all interrupts of lower or equal priority than the threshold are masked. However, priority 0 can never be masked by this threshold; a priority threshold of 0 is treated the same way as priority 1.

PRIORITY and PRIORITYTHRESHOLD fields values can be set between 0x0 and 0x3F; 0x0 is the highest priority and 0x3F is the lowest priority.

When priority masking is not necessary, a priority threshold value of 0xFF disables the priority threshold mechanism. This value is also the reset default for backward compatibility with previous versions of the INTCPS.

8.4.1.3 Priority Sorting

A priority level (0 being the highest) is assigned to each incoming interrupt line. Both the priority level and the interrupt request type are configured by the MPU_INTC.INTCPS_ILRm register. If more than one incoming interrupt with the same priority level and interrupt request type occur simultaneously, the highest-numbered interrupt is serviced first.

When one or more unmasked incoming interrupts are detected, the INTCPS separates between IRQ and FIQ using the corresponding MPU_INTC.INTCPS_ILRm[0] FIQNIRQ bit. The result is placed in INTCPS_PENDING_IRQn or INTCPS_PENDING_FIQn

If no other interrupts are currently being processed, INTCPS asserts IRQ/FIQ and starts the priority computation. Priority sorting for IRQ and FIQ can execute in parallel.

Each IRQ/FIQ priority sorter determines the highest priority interrupt number. Each priority number is placed in the corresponding MPU_INTC.INTCPS_SIR_IRQ[6:0] ACTIVEIRQ field or MPU_INTC.INTCPS_SIR_FIQ[6:0] ACTIVEFIQ field. The value is preserved until the corresponding MPU_INTC.INTCPS_CONTROL NEWIRQAGR or NEWFIQAGR bit is set.

Once the interrupting peripheral device has been serviced and the incoming interrupt de-asserted, the user must write to the appropriate NEWIRQAGR or NEWFIQAGR bit to indicate to the INTCPS the interrupt has been handled. If there are any pending unmasked incoming interrupts for this interrupt request type, the INTCPS restarts the appropriate priority sorter; otherwise, the IRQ or FIQ interrupt line is de-asserted.

8.4.2 Secure Interrupts (HS Devices Only)

To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

8.4.3 Register Protection

If the MPU_INTC.INTCPS_PROTECTION[0] PROTECTION bit is set, access to the INTCPS registers is restricted to the supervisor mode. Access to the MPU_INTC.INTCPS_PROTECTION register is always restricted to privileged mode.

8.4.4 Module Power Saving

The INTCPS provides an auto-idle function in its three clock domains:

- Interface clock
- Functional clock
- Synchronizer clock

The interface clock auto-idle power-saving mode is enabled if the MPU_INTC.INTCPS_SYSCONFIG[0] AUTOIDLE bit is set to 1. When this mode is enabled and there is no activity on the bus interface, the interface clock is disabled internally to the module, thus reducing power consumption. When there is new activity on the bus interface, the interface clock restarts without any latency penalty. After reset, this mode is disabled, by default.

The functional clock auto-idle power-saving mode is enabled if the MPU_INTC.INTCPS_IDLE[0] FUNCIDLE bit is set to 0. When this mode is enabled and there is no active interrupt (IRQ or FIQ interrupt being processed or generated) or no pending incoming interrupt, the functional clock is disabled internally to the module, thus reducing power consumption. When a new unmasked incoming interrupt is detected, the functional clock restarts and the INTCPS processes the interrupt. If this mode is disabled, the interrupt latency is reduced by one cycle. After reset, this mode is enabled, by default.

The synchronizer clock allows external asynchronous interrupts to be resynchronized before they are masked. The synchronizer input clock has an auto-idle power-saving mode enabled if the MPU_INTC.INTCPS_IDLE[1] TURBO bit is set to 1. If the auto-idle mode is enabled, the standby power is reduced, but the IRQ or FIQ interrupt latency increases from four to six functional clock cycles. This feature can be enabled dynamically according to the requirements of the device. After reset, this mode is disabled, by default.

To ensure optimal power consumption, INTC_INIT_REGISTER1[0] INIT1 and INTC_INIT_REGISTER2[1] INIT2 bits must be set to 1 during initialization.

8.4.5 Interrupt Latency

The IRQ/FIQ interrupt generation takes four INTCPS functional clock cycles (plus or minus one cycle) if the MPU_INTC.INTCPS_IDLE[1] TURBO bit is set to 0. If the TURBO bit is set to 1, the interrupt generation takes six cycles, but power consumption is reduced while waiting for an interrupt.

These latencies can be reduced by one cycle by disabling functional clock auto-idle (MPU_INTC.INTCPS_IDLE[0] FUNCIDLE bit set to 1), but power consumption is increased, so the benefit is minimal. For information about power saving, see Section 8.4.4.

To minimize interrupt latency when an unmasked interrupt occurs, the IRQ or FIQ interrupt is generated before priority sorting completion. The priority sorting takes 10 functional clock cycles, which is less than the minimum number of cycles required for the MPU to switch to the interrupt context after reception of the IRQ or FIQ event.

Any read of the MPU_INTC.INTCPS_SIR_IRQ or MPU_INTC.INTCPS_SIR_FIQ register during the priority sorting process stalls until priority sorting is complete and the relevant register is updated. However, the delay between the interrupt request being generated and the interrupt service routine being executed is such that priority sorting always completes before the MPU_INTC.INTCPS_SIR_IRQ or MPU_INTC.INTCPS_SIR_FIQ register is read.

8.5 Interrupt Basic Programming Model

8.5.1 Initialization Sequence

1. Program the MPU_INTC.INTCPS_SYSCONFIG register: If necessary, enable the interface clock autogating by setting the AUTOIDLE bit.
2. Program the MPU_INTC.INTCPS_IDLE register: If necessary, disable functional clock autogating or enable synchronizer autogating by setting the FUNCIDLE bit or TURBO bit accordingly.
3. Program the MPU_INTC.INTCPS_ILRm register for each interrupt line: Assign a priority level and set the FIQNFIQ bit for an FIQ interrupt (by default, interrupts are mapped to IRQ and priority is 0x0 [highest]).
4. Program the MPU_INTC.INTCPS_MIRn register: Enable interrupts (by default, all interrupt lines are masked).

NOTE: To program the MPU_INTC.INTCPS_MIRn register, the MPU_INTC.INTCPS_MIR_SETn and MPU_INTC.INTCPS_MIR_CLEARn registers are provided to facilitate the masking, even if it is possible for backward-compatibility to write directly to the MPU_INTC.INTCPS_MIRn register.

8.5.2 MPU INTC Processing Sequence

After the MPU_INTC.INTCPS_MIRn and MPU_INTC.INTCPS_ILRm registers are configured to enable and assign priorities to incoming interrupts, the interrupt is processed as explained in the following subsections.

IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in **bold** characters in the text or the code below.

1. One or more unmasked incoming interrupts (M_IRQ_n signals) are received and IRQ or FIQ outputs (MPU_INTC_IRQ/**FIQ**) are not currently asserted.
2. If the MPU_INTC.INTCPS_ILRm[0] FIQNIRQ bit is set to 0, the MPU_INTC_IRQ output signal is generated. If the FIQNIRQ bit is set to 1, the MPU_INTC_FIQ output signal is generated.
3. The INTC performs the priority sorting and updates the MPU_INTC.INTCPS_SIR_IRQ[6:0] ACTIVEIRQ /**MPU_INTC.INTCPS_SIR_FIQ[6:0] ACTIVEFIQ** field with the current interrupt number.
4. During priority sorting, if the IRQ/**FIQ** is enabled at the host processor side, the host processor automatically saves the current context and executes the ISR as follows:

NOTE: The ARM host processor automatically performs the following actions in pseudo code.

```
LR = PC + 4                    /* return link                    */
SPSR = CPSR                    /* Save CPSR before execution     */
CPSR[5] = 0                    /* Execute in ARM state           */
CPSR[7] = 1                    /* Disable IRQ                    */
CPSR[8] = 1                    /* Disable Imprecise Data Aborts  */
CPSR[9] = CP15_reg1_EEbit      /* Endianness on exception entry  */
if interrupt == IRQ then
    CPSR[4:0] = 0b10010            /* Enter IRQ mode                 */
```

```
    if high vectors configured then
        PC = 0xFFFF0018
    else
        PC = 0x00000018              /* execute interrupt vector        */
else if interrupt == FIQ then
    CPSR[4:0] = 0b10001              /* Enter FIQ mode                  */
    CPSR[6] = 1                      /* Disable FIQ                     */
    if high vectors configured then
        PC = 0xFFFF001C
    else
        PC = 0x0000001C              /* execute interrupt vector        */
end if
```

5. The ISR saves the remaining context, identifies the interrupt source by reading the ACTIVEIRQ/**ACTIVEFIQ** field, and jumps to the relevant subroutine handler as follows:

> **CAUTION**
>
> The code in steps 5 and 7 is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
; INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register address
INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR .word 0x48200040/0x48200044

; ACTIVEIRQ bit field mask to get only the bit field
ACTIVEIRQ_MASK .equ 0x7F

_IRQ_ISR/_FIQ_ISR:

    ; Save the critical context
    STMFD SP!, {R0-R12, LR}             ; Save working registers and the Link register
    MRS R11, SPSR                   ; Save the SPSR into R11

    ; Get the number of the highest priority active IRQ/FIQ
    LDR R10, INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR
    LDR R10, [R10]                  ; Get the INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register
    AND R10, R10, #ACTIVEIRQ_MASK   ; Apply the mask to get the active IRQ number

    ; Jump to relevant subroutine handler
    LDR PC, [PC, R10, lsl #2]        ; PC base address points this instruction + 8
    NOP                             ; To index the table by the PC

    ; Table of handler start addresses
    .word IRQ0handler ;For IRQ0 of BANK0
    .word IRQ1handler
    .word IRQ2handler
```

6. The subroutine handler executes code specific to the peripheral generating the interrupt by handling the event and de-asserting the interrupt condition at the peripheral side.

```
; IRQ0 subroutine
IRQ0handler:

    ; Save working registers
    STMFD SP!, {R0-R1}

    ; Now read-modify-write the peripheral module status register
    ; to de-assert the M_IRQ_0 interrupt signal

    ; De-Assert the peripheral interrupt
    MOV R0, #0x7                    ; Mask for 3 flags
    LDR R1, MODULE0_STATUS_REG_ADDR ; Get the address of the module Status Register
    STR R0, [R1]                    ; Clear the 3 flags
```

```
    ; Restore working registers
    LDMFD SP!, {R0-R1}

    ; Jump to the end part of the ISR
    B IRQ_ISR_end/FIQ_ISR_end
```

7. After the return of the subroutine, the ISR sets the NEWIRQAGR/**NEWFIQAGR** bit to enable the processing of subsequent pending IRQs/**FIQs** and to restore ARM context in the following code. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/**FIQs**, a Data Synchronization Barrier is used. This operation ensure that the IRQ/**FIQ** line is de-asserted before IRQ/**FIQ** enabling. After that, the INTC processes any other pending interrupts or de-asserts the MPU_INTC_IRQ/**MPU_INTC_FIQ** signal if there is no interrupt.

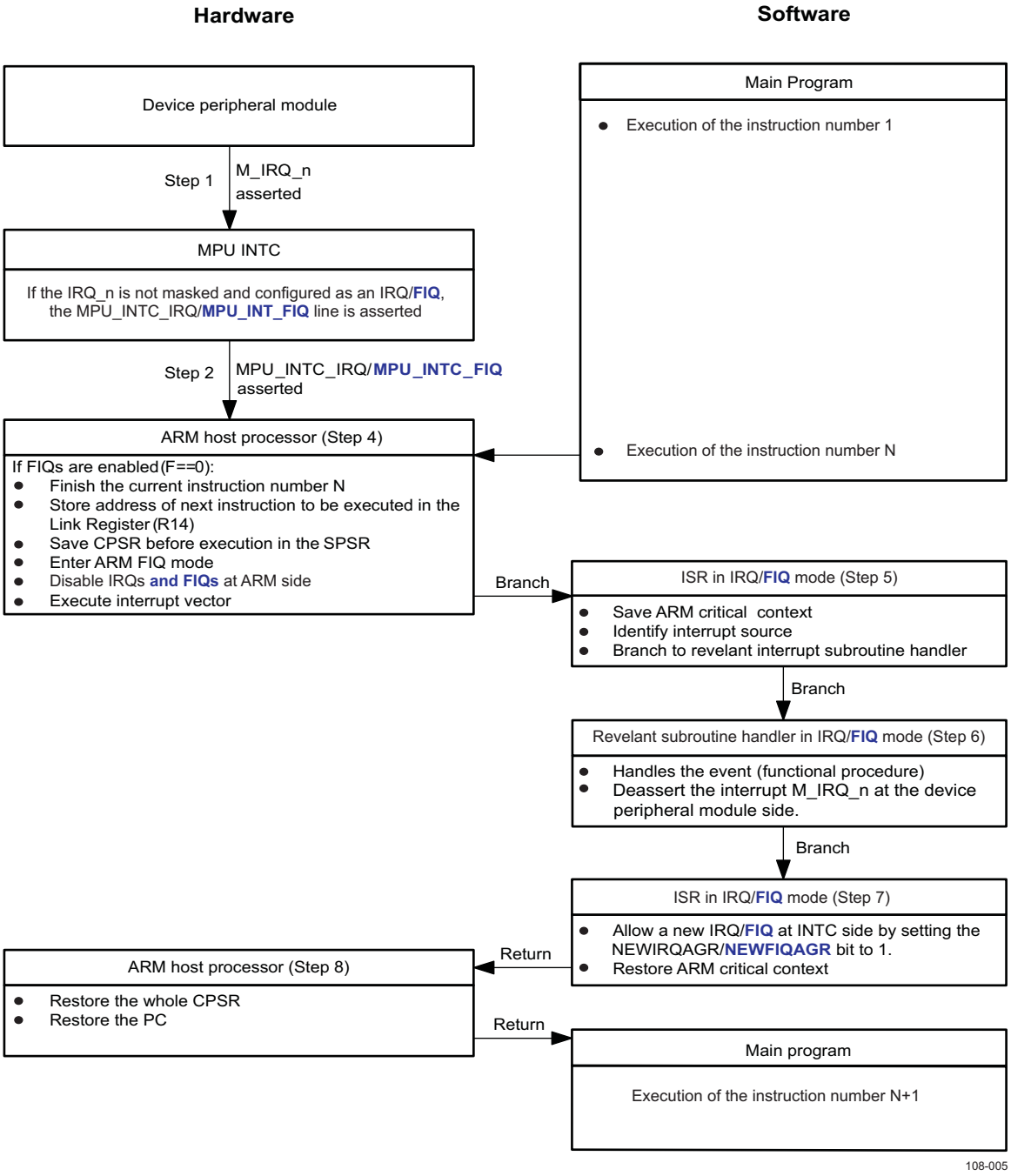
```
; INTCPS_CONTROL register address
INTCPS_CONTROL_ADDR .word 0x48200048

; NEWIRQAGR/NEWFIQAGR bit mask to set only the NEWIRQAGR/NEWFIQAGR bit
NEWIRQAGR_MASK/NEWFIQAGR_MASK .equ 0x01/0x02

IRQ_ISR_end/FIQ_ISR_end:

    ; Allow new IRQs/FIQs at INTC side
    ; The INTCPS_CONTROL register is a write only register so no need to write back others bits
    MOV R0, #NEWIRQAGR_MASK/NEWFIQAGR_MASK ; Get the NEWIRQAGR/NEWFIQAGR bit position
    LDR R1, INTCPS_CONTROL_ADDR
    STR R0, [R1]                    ; Write the NEWIRQAGR/NEWFIQAGR bit to allow new IRQs/FIQs

    ; Data Synchronization Barrier
    MOV R0, #0
    MCR P15, #0, R0, C7, C10, #4

    ; restore critical context
    MSR SPSR, R11                   ; Restore the SPSR from R11
    LDMFD SP!, {R0-R12, LR}         ; Restore working registers and Link register

    ; Return after handling the interrupt
    SUBS PC, LR, #4
```

8. After the ISR return, the ARM automatically restores its context as follows:

```
CPSR = SPSR
PC = LR
```

Figure 8-5 shows the IRQ/**FIQ** processing sequence from the originating device peripheral module to the main program interruption.

Figure 8-5. IRQ/FIQ Processing Sequence



NOTE: The differences between the IRQ and the FIQ sequence are highlighted in blue and bold characters.

The priority sorting mechanism is frozen during an interrupt processing sequence. If an interrupt condition occurs during this time, the interrupt is not lost. It is sorted when the NEWIRQAGR/**NEWFIQAGR** bit is set (priority sorting is reactivated).

8.5.3 MPU INTC Preemptive Processing Sequence

Preemptive interrupts, also called nested interrupts, can reduce the latencies for higher priority interrupts. A preemptive ISR can be suspended by a higher priority interrupt. Thus, the higher priority interrupt can be served immediately.

Nested interrupts must be used carefully to avoid using corrupted data. Programmers must save corruptible registers and enable IRQ or FIQ at ARM side.

IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in **bold** characters in the text or the code below.

To enable IRQ/**FIQ** preemption by higher priority IRQs/**FIQs**, programers can follow this procedure to write the ISR:

At the beginning of an IRQ/**FIQ** ISR:

1. Save the ARM critical context registers.
2. Save the MPU_INTC.INTCPS_THRESHOLD PRIORITYTHRESHOLD field before modifying it.
3. Read the active interrupt priority in the MPU_INTC.INTCPS_IRQ_PRIORITY IRQPRIORITY / **MPU_INTC.INTCPS_FIQ_PRIORITY FIQPRIORITY** field and write it to the PRIORITYTHRESHOLD(1) field.
4. Read the active interrupt number in the MPU_INTC.INTCPS_SIR_IRQ[6:0] ACTIVEIRQ / **MPU_INTC.INTCPS_SIR_FIQ[6:0] ACTIVEFIQ** field to identify the interrupt source.
5. Write 1 to the appropriate MPU_INTC.INTCPS_CONTROL NEWIRQAGR **and** (2) **NEWFIQAGR** bit while an interrupt is still processing to allow only higher priority interrupts to preempt.
6. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/**FIQs**, a Data Synchronization Barrier is used. This operation ensure that the IRQ line is de-asserted before IRQ/**FIQ** enabling.
7. Enable IRQ/**FIQ** at ARM side.
8. Jump to the relevant subroutine handler.

The sample code below shows the previous steps:

CAUTION

The code below is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
;  bit field mask to get only the bit field
ACTIVEPRIO_MASK .equ 0x3F

_IRQ_ISR:
    ; Step 1 : Save the critical context
    STMFD SP!, {R0-R12, LR}            ; Save working registers
    MRS R11, SPSR                      ; Save the SPSR into R11

    ; Step 2 : Save the INTCPS_THRESHOLD register into R12
    LDR R0, INTCPS_THRESHOLD_ADDR
    LDR R12, [R0]
```

(1) The priority-threshold mechanism is enabled automatically when writing a priority in the range of 0x00 to 0x3F while reading it from the IRQPRIORITY and FIQPRIORITY fields. Writing a value of 0xFF (reset default) disables the priority-threshold mechanism. Values between 0x3F and 0xFF must not be used.
When the hardware-priority threshold is in use, the priorities of interrupts selected as FIQ or IRQ become linked; otherwise, they are independent. When they are linked, all FIQ priorities must be set higher than all IRQ priorities to maintain the relative priority of FIQ over IRQ.

(2) When handling FIQs using the priority-threshold mechanism, both NEWFIQAGR and NEWIRQAGR bits must be written at the same time to ensure that the new priority threshold is applied while an IRQ sort is in progress. This IRQ will not have been seen by the ARM, as it will have been masked on entry to the FIQ ISR. However, the source of the IRQ remains active and it is finally processed when the priority threshold falls to a priority sufficiently low to allow it to be processed. The precaution of writing to New FIQ Agreement is not required during an IRQ ISR, as FIQ sorting is not affected (providing all FIQ priorities are higher than all IRQ priorities).

```
    ; Step 3 : Get the priority of the highest priority active IRQ
    LDR R1, INTCPS_IRQ_PRIORITY_ADDR/INTCPS_FIQ_PRIORITY_ADDR
    LDR R1, [R1]                        ; Get the INTCPS_IRQ_PRIORITY/INTCPS_FIQ_PRIORITY register
    AND R1, R1, #ACTIVEPRIO_MASK        ; Apply the mask to get the priority of the IRQ
    STR R1, [R0]                        ; Write it to the INTCPS_THRESHOLD register

    ; Step 4 : Get the number of the highest priority active IRQ
    LDR R10, INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR
    LDR R10, [R10]                      ; Get the INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register
    AND R10, R10, #ACTIVEIRQ_MASK       ; Apply the mask to get the active IRQ number

    ; Step 5 : Allow new IRQs and FIQs at INTC side
    MOV R0, #0x1/0x3                    ; Get the NEWIRQAGR and NEWFIQAGR bit position
    LDR R1, INTCPS_CONTROL_ADDR
    STR R0, [R1]                        ; Write the NEWIRQAGR and NEWFIQAGR bit

    ; Step 6 : Data Synchronization Barrier
    MOV R0, #0
    MCR P15, #0, R0, C7, C10, #4

    ; Step 7 : Read-modify-write the CPSR to enable IRQs/FIQs at ARM side
    MRS R0, CPSR                        ; Read the status register
    BIC R0, R0, #0x80/0x40              ; Clear the I/F bit
    MSR CPSR, R0                        ; Write it back to enable IRQs

    ; Step 8 : Jump to relevant subroutine handler
    LDR PC, [PC, R10, lsl #2]           ; PC base address points this instruction + 8
    NOP                                 ; To index the table by the PC

    ; Table of handler start addresses
    .word IRQ0handler ;IRQ0 BANK0
    .word IRQ1handler
    .word IRQ2handler
```

After the return of the relevant IRQ/**FIQ** subroutine handle :

1. Disable IRQs/**FIQs** at ARM side.
2. Restore the MPU_INTC.INTCPS_THRESHOLD PRIORITYTHRESHOLD field.
3. Restore the ARM critical context registers.

The sample code below shows the three previous steps:

CAUTION

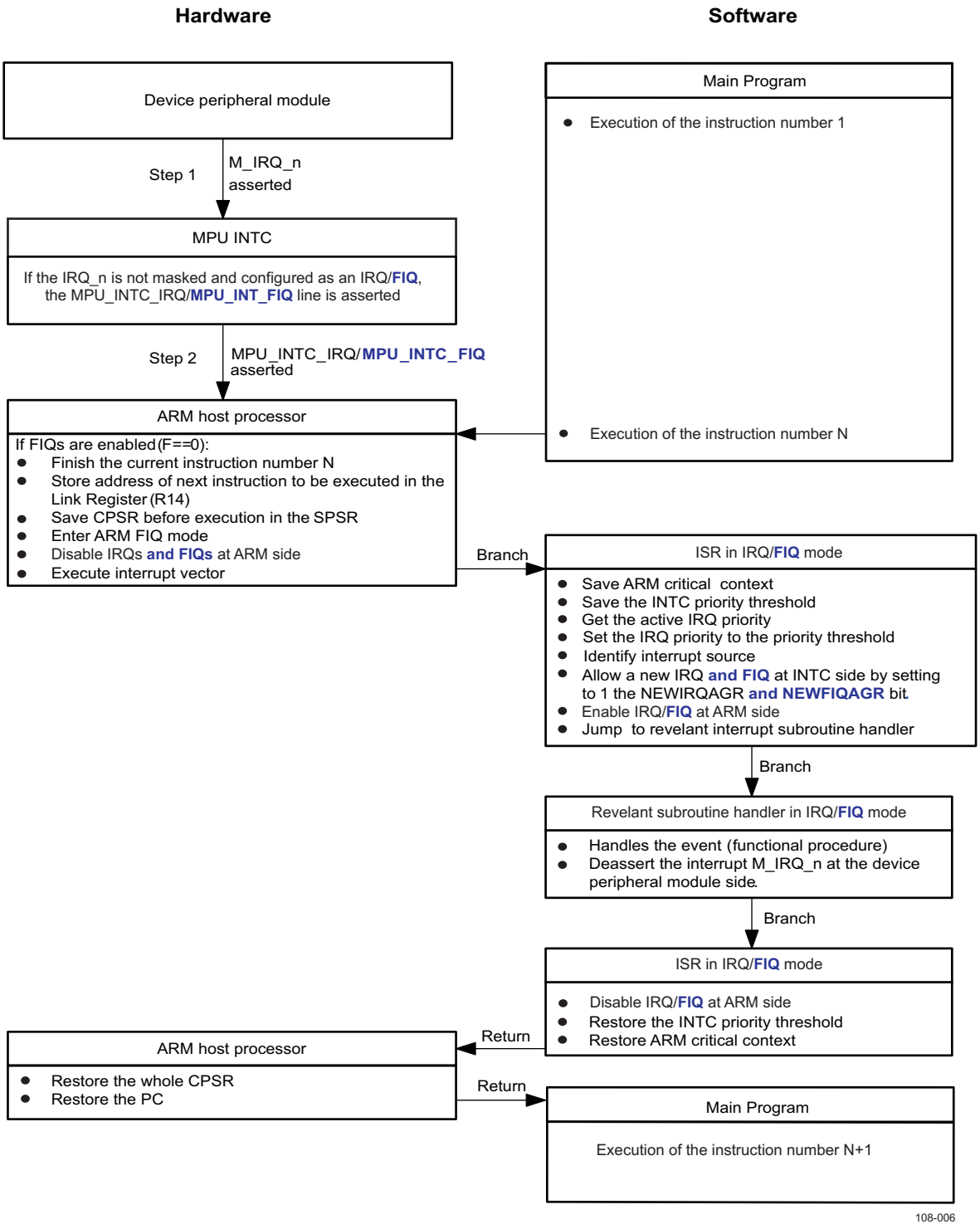
The code below is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
IRQ_ISR_end:

    ; Step 1 : Read-modify-write the CPSR to disable IRQs/FIQs at ARM side
    MRS R0, CPSR                        ; Read the CPSR
    ORR R0, R0, #0x80/0x40              ; Set the I/F bit
    MSR CPSR, R0                        ; Write it back to disable IRQs

    ; Step 2 : Restore the INTCPS_THRESHOLD register from R12
    LDR R0, INTCPS_THRESHOLD_ADDR
    STR R12, [R0]

    ; Step 3 : Restore critical context
    MSR SPSR, R11                       ; Restore the SPSR from R11
    LDMFD SP!, {R0-R12, LR}             ; Restore working registers and Link register
```

```
; Return after handling the interrupt
SUBS PC, LR, #4
```

Figure 8-6 shows the nested IRQ/**FIQ** processing sequence from the originating device peripheral module to the main program interruption.

Figure 8-6. Nested IRQ/FIQ Sequence



NOTE: The differences between the IRQ and the FIQ sequence are highlighted in blue and bold characters.

8.5.4 MPU INTC Spurious Interrupt Handling

The spurious flag indicates whether the result of the sorting (a window of 10 INTC functional clock cycles after the interrupt assertion) is invalid. The sorting is invalid if:

- The interrupt that triggered the sorting is no longer active during the sorting.
- A change in the mask has affected the result during the sorting time.

As a result, the values in the MPU_INTC.INTCPS_MIRn, MPU_INTC.INTCPS_ILRm, or MPU_INTC.INTCPS_MIR_SETn registers must not be changed while the corresponding interrupt is asserted. If these registers are changed within the 10-cycle window after the interrupt assertion, only the active interrupt input that triggered the sort can be masked before its turn in the sort. The resulting values of the following registers become invalid:

- MPU_INTC.INTCPS_SIR_IRQ
- MPU_INTC.INTCPS_SIR_FIQ
- MPU_INTC.INTCPS_IRQ_PRIORITY
- MPU_INTC. INTCPS_FIQ_PRIORITY

This condition is detected for both IRQ and FIQ, and the invalid status is flagged across the SPURIOUSIRQFLAG (see Note 1) and SPURIOUSFIQFLAG (see Note 2) bit fields in the SIR and PRIORITY registers. A 0 indicates valid and a 1 indicates invalid interrupt number and priority. The invalid indication can be tested in software as a false register value.

NOTE:

1. The MPU_INTC.INTCPS_SIR_IRQ[31:7] SPURIOUSIRQFLAG field is a copy of the MPU_INTC.INTCPS_IRQ_PRIORITY[31:7] SPURIOUSIRQFLAG field.
2. The MPU_INTC.INTCPS_SIR_FIQ[31:7] SPURIOUSFIQFLAG field is a copy of the MPU_INTC.INTCPS_FIQ_PRIORITY[31:7] SPURIOUSFIQFLAG field.

8.6 Interrupt Controller Registers

Table 8-5 lists the base address and address space for the INTC.

Table 8-5. INTC Instance Summary

Module Name	Base Address	Size
MPU INTC	0x4820 0000	4K bytes

8.6.1 Register Mapping Summary

CAUTION

MPU INTC registers are limited to 32-bit and 16-bit data accesses. 8-bit is not allowed and can corrupt register content.

In Section 8.6.2, each register from MPU_INTC.INTCPS_ITRn to MPU_INTC.INTCPS_PENDING_FIQn contains 32 bits, 1 bit for each interrupt (in ascending order: bit 0 of the MPU_INTC.INTCPS_ITR0 register applies to interrupt line 0; bit 0 of the MPU_INTC.INTCPS_ITR1 register applies to interrupt line 32).

Table 8-6. MPU INTC Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	MPU INTC Physical Address	Section
INTCPS_SYSCONFIG	RW	32	0x0000 0010	0x4820 0010	Section 8.6.2.1
INTCPS_SYSSTATUS	R	32	0x0000 0014	0x4820 0014	Section 8.6.2.2
INTCPS_SIR_IRQ	R	32	0x0000 0040	0x4820 0040	Section 8.6.2.3
INTCPS_SIR_FIQ	R	32	0x0000 0044	0x4820 0044	Section 8.6.2.4
INTCPS_CONTROL	RW	32	0x0000 0048	0x4820 0048	Section 8.6.2.5
INTCPS_PROTECTION	RW	32	0x0000 004C	0x4820 004C	Section 8.6.2.6
INTCPS_IDLE	RW	32	0x0000 0050	0x4820 0050	Section 8.6.2.7
INTCPS_IRQ_PRIORITY	RW	32	0x0000 0060	0x4820 0060	Section 8.6.2.8
INTCPS_FIQ_PRIORITY	RW	32	0x0000 0064	0x4820 0064	Section 8.6.2.9
INTCPS_THRESHOLD	RW	32	0x0000 0068	0x4820 0068	Section 8.6.2.10
INTCPS_ITRn(1)	R	32	0x0000 0080 + (0x20 * n)	0x4820 0080 + (0x20 * n)	Section 8.6.2.11
INTCPS_MIRn(1)	RW	32	0x0000 0084 + (0x20 * n)	0x4820 0084 + (0x20 * n)	Section 8.6.2.12
INTCPS_MIR_CLEARn(1)	W	32	0x0000 0088 + (0x20 * n)	0x4820 0088 + (0x20 * n)	Section 8.6.2.13
INTCPS_MIR_SETn(1)	W	32	0x0000 008C + (0x20 * n)	0x4820 008C + (0x20 * n)	Section 8.6.2.14
INTCPS_ISR_SETn(1)	RW	32	0x0000 0090 + (0x20 * n)	0x4820 0090 + (0x20 * n)	Section 8.6.2.15
INTCPS_ISR_CLEARn(1)	W	32	0x0000 0094 + (0x20 * n)	0x4820 0094 + (0x20 * n)	Section 8.6.2.16
INTCPS_PENDING_IRQn(1)	R	32	0x0000 0098 + (0x20 * n)	0x4820 0098 + (0x20 * n)	Section 8.6.2.17
INTCPS_PENDING_FIQn(1)	R	32	0x0000 009C + (0x20 * n)	0x4820 009C + (0x20 * n)	Section 8.6.2.18
INTCPS_ILRm(2)	RW	32	0x0000 0100 + (0x4 * m)	0x4820 0100 + (0x4 * m)	Section 8.6.2.19

(1) n = 0 to 2
(2) m = 0 to 95

Table 8-7. Device INTC Initialization Register Summary

Register Name	Type	Register Width (Bits)	Physical Address	Section
INTC_INIT_REGISTER1	RW	32	0x480C 7010	Section 8.6.3.1
INTC_INIT_REGISTER2	RW	32	0x480C 7050	Section 8.6.3.2

8.6.2 MPU INTC Register Descriptions

Table 8-8 through Table 8-26 describe the MPU INTC registers.

8.6.2.1 INTCPS_SYSCONFIG

Table 8-8. INTCPS_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4820 0010	**Instance**	MPU INTC
Description	This register controls various parameters of the module interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. Read returns 0. Write 0x0: No functional effect Write 0x1: The module is reset.	RW	0
0	AUTOIDLE	Internal interface clock gating strategy 0x0: Interface clock is free-running. 0x1: Automatic interface clock gating strategy is applied, based on the interface bus activity.	RW	0

8.6.2.2 INTCPS_SYSSTATUS

Table 8-9. INTCPS_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4820 0014	**Instance**	MPU INTC
Description	This register provides status information about the module.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns reset value.	R	0x00000000
0	RESETDONE	Internal reset monitoring Read 0x0: Internal module reset is ongoing. Read 0x1: Reset complete	R	-

8.6.2.3 INTCPS_SIR_IRQ

Table 8-10. INTCPS_SIR_IRQ

Address Offset	0x040		
Physical Address	0x4820 0040	**Instance**	MPU INTC
Description	This register supplies the currently active IRQ interrupt number.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSIRQFLAG																								ACTIVEIRQ							

Bits	Field Name	Description	Type	Reset
31:7	SPURIOUSIRQFLAG	Spurious IRQ flag	R	0x1FFFFFF
6:0	ACTIVEIRQ	Active IRQ number	R	0x00

8.6.2.4 INTCPS_SIR_FIQ

Table 8-11. INTCPS_SIR_FIQ

Address Offset	0x044		
Physical Address	0x4820 0044	**Instance**	MPU INTC
Description	This register supplies the currently active FIQ interrupt number.		
Type	R		

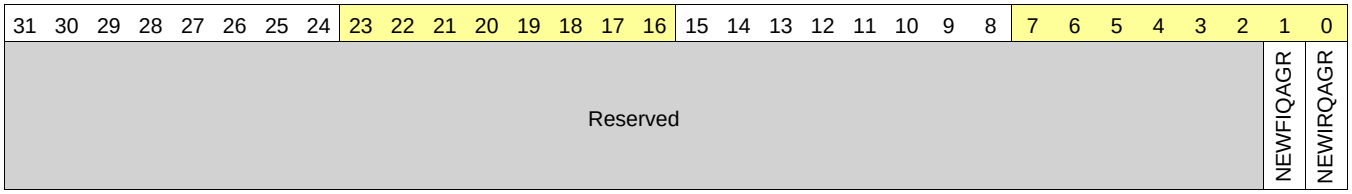
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSFIQFLAG																								ACTIVEFIQ							

Bits	Field Name	Description	Type	Reset
31:7	SPURIOUSFIQFLAG	Spurious FIQ flag	R	0x1FFFFFF
6:0	ACTIVEFIQ	Active FIQ number	R	0x00

8.6.2.5 INTCPS_CONTROL

Table 8-12. INTCPS_CONTROL

Address Offset	0x048		
Physical Address	0x4820 0048	**Instance**	MPU INTC
Description	This register contains the new interrupt agreement bits.		
Type	RW		

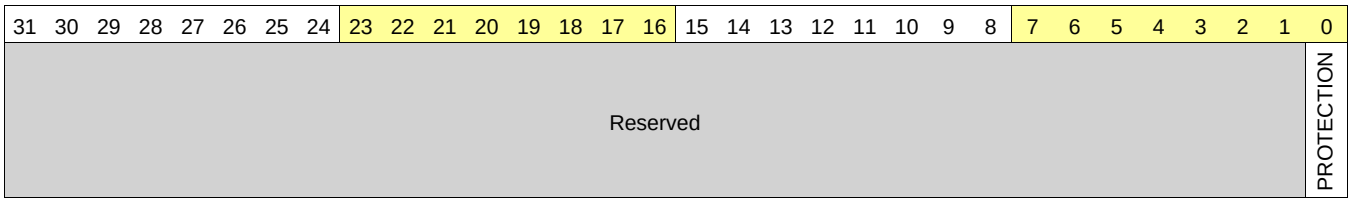


Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	NEWFIQAGR	Reset FIQ output and enable new FIQ generation.	W	-
		Write 0x0: No functional effect		
		Write 0x1: Reset FIQ output and enable new FIQ generation.		
0	NEWIRQAGR	New IRQ generation	W	-
		Write 0x0: No functional effect		
		Write 0x1: Reset IRQ output and enable new IRQ generation.		

8.6.2.6 INTCPS_PROTECTION

Table 8-13. INTCPS_PROTECTION

Address Offset	0x04C		
Physical Address	0x4820 004C	**Instance**	MPU INTC
Description	This register controls protection of the other registers. It can be accessed only in supervisor mode, regardless of the current value of the protection bit.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
0	PROTECTION	Protection mode	RW	0
		0x0: Protection mode is disabled (default).		
		0x1: Protection mode is enabled. When enabled, all the MPU INTC registers are accessible only in privileged mode.		

8.6.2.7 INTCPS_IDLE

Table 8-14. INTCPS_IDLE

Address Offset	0x050		
Physical Address	0x4820 0050	**Instance**	MPU INTC
Description	This register controls the functional clock auto-idle and the synchronizer clock auto-gating.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														TURBO	FUNCIDLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	TURBO	Input synchronizer clock auto-gating	RW	0
		0x0: Input synchronizer clock is free-running (default).		
		0x1: Input synchronizer clock is auto-gated based on interrupt input activity.		
0	FUNCIDLE	Functional clock idle mode	RW	0
		0x0: Functional clock gating strategy is applied (default).		
		0x1: Functional clock is free-running.		

8.6.2.8 INTCPS_IRQ_PRIORITY

Table 8-15. INTCPS_IRQ_PRIORITY

Address Offset	0x060		
Physical Address	0x4820 0060	**Instance**	MPU INTC
Description	This register supplies the currently active IRQ priority level.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSIRQFLAG																								IRQPRIORITY							

Bits	Field Name	Description	Type	Reset
31:6	SPURIOUSIRQFLAG	Spurious IRQ flag	R	0x3FFFFFF
5:0	IRQPRIORITY	Current IRQ priority	R	0x00

8.6.2.9 INTCPS_FIQ_PRIORITY

Table 8-16. INTCPS_FIQ_PRIORITY

Address Offset	0x064		
Physical Address	0x4820 0064	**Instance**	MPU INTC
Description	This register supplies the currently active FIQ priority level.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSFIQFLAG																								FIQPRIORITY							

Bits	Field Name	Description	Type	Reset
31:6	SPURIOUSFIQFLAG	Spurious FIQ flag	R	0x3FFFFFF
5:0	FIQPRIORITY	Current FIQ priority	R	0x00

8.6.2.10 INTCPS_THRESHOLD

Table 8-17. INTCPS_THRESHOLD

Address Offset	0x068		
Physical Address	0x4820 0068	**Instance**	MPU INTC
Description	This register sets the priority threshold.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PRIORITYTHRESHOLD							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x000000
7:0	PRIORITYTHRESHOLD	Priority threshold	RW	0xFF
		Write 0xFF: Priority threshold disabled		
		Write 0x0 to 0x3F: Priority threshold enabled		

8.6.2.11 INTCPS_ITRn

Table 8-18. INTCPS_ITRn

Address Offset	0x080 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0080 + (0x20 * n)	**Instance**	MPU INTC
Description	This register shows the raw interrupt input status before masking.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITR																															

Bits	Field Name	Description	Type	Reset
31:0	ITR	Interrupt status before masking	R	Depends on interrupt inputs

8.6.2.12 INTCPS_MIRn

Table 8-19. INTCPS_MIRn

Address Offset	0x084 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0084 + (0x20 * n)	**Instance**	MPU INTC
Description	This register contains the interrupt mask.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIR																															

Bits	Field Name	Description	Type	Reset
31:0	MIR	Interrupt mask 0x1: The interrupt is masked 0x0: The interrupt is unmasked	RW	0xFFFFFFFF

8.6.2.13 INTCPS_MIR_CLEARn

Table 8-20. INTCPS_MIR_CLEARn

Address Offset	0x088 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0088 + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to clear the interrupt mask bits.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIRCLEAR																															

Bits	Field Name	Description	Type	Reset
31:0	MIRCLEAR	Clear the interrupt mask bits. Read returns 0. Write 0x1: Clears the MIR mask bit to 0 Write 0x0: No functional effect	W	0x00000000

8.6.2.14 INTCPS_MIR_SETn

Table 8-21. INTCPS_MIR_SETn

Address Offset	0x08C + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 008C + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to set the interrupt mask bits.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIRSET																															

Bits	Field Name	Description	Type	Reset
31:0	MIRSET	Mask the interrupt bits. Read returns 0. Write 0x0: No functional effect Write 0x1: Sets the MIR mask bit to 1.	W	0x00000000

8.6.2.15 INTCPS_ISR_SETn

Table 8-22. INTCPS_ISR_SETn

Address Offset	0x090 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0090 + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISRSET																															

Bits	Field Name	Description	Type	Reset
31:0	ISRSET	Set the software interrupt bits. Read returns the currently active software interrupts. Write 0x0: No functional effect Write 0x1: Sets the software interrupt bits to 1.	RW	0x00000000

8.6.2.16 INTCPS_ISR_CLEARn

Table 8-23. INTCPS_ISR_CLEARn

Address Offset	0x094 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0094 + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to clear the software interrupt bits.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISRCLEAR																															

Bits	Field Name	Description	Type	Reset
31:0	ISRCLEAR	Clear the software interrupt bits. Read returns 0. Write 0x0: No functional effect Write 0x1: Clears the software interrupt bits to 0.	W	0x00000000

8.6.2.17 INTCPS_PENDING_IRQn

Table 8-24. INTCPS_PENDING_IRQn

Address Offset	0x098 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0098 + (0x20 * n)	**Instance**	MPU INTC
Description	This register contains the IRQ status after masking.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PENDINGIRQ																															

Bits	Field Name	Description	Type	Reset
31:0	PENDINGIRQ	IRQ status after masking.	R	0x00000000

8.6.2.18 INTCPS_PENDING_FIQn

Table 8-25. INTCPS_PENDING_FIQn

Address Offset	0x09C + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 009C + (0x20 * n)	**Instance**	MPU INTC
Description	This register contains the FIQ status after masking.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PENDINGFIQ																															

Bits	Field Name	Description	Type	Reset
31:0	PENDINGFIQ	FIQ status after masking.	R	0x00000000

8.6.2.19 INTCPS_ILRm

Table 8-26. INTCPS_ILRm

Address Offset	0x100 + (0x4 * m)	**Index**	m = 0 to 95
Physical Address	0x4820 0100 + (0x4 * m)	**Instance**	MPU INTC
Description	These registers contain the priority for the interrupts and the FIQ/IRQ steering.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PRIORITY						Reserved	FIQNIRQ

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x000000
7:2	PRIORITY	Interrupt priority	RW	0x00
1	Reserved	Write 0 for future compatibility. Read returns reset value.	R	0
0	FIQNIRQ	Interrupt IRQ FIQ mapping. Read returns reset value. Write 0x0: Interrupt is routed to IRQ. Write 0x1: Interrupt is routed to FIQ.	RW	0

8.6.3 Device INTC Initialization Register Descriptions

Table 8-27 and Table 8-28 describe device INTC registers that need to be programmed during initialization to ensure optimal power savings.

8.6.3.1 INTC_INIT_REGISTER1

Table 8-27. INTC_INIT_REGISTER1

Physical Address	0x480C 7010	**Instance**	Device INTC Initialization
Description	This register enables power optimizations.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															INIT1

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
0	INIT1	To ensure lowest power configuration, this bit must be set to 1 during initialization.	RW	0

8.6.3.2 INTC_INIT_REGISTER2

Table 8-28. INTC_INIT_REGISTER2

Physical Address	0x480C 7050	**Instance**	Device INTC Initialization
Description	This register enables power optimizations.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														INIT2	Reserved

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	INIT2	For optimal power consumption, this bit must be set to 1 during initialization.	RW	0
0	Reserved	For optimal power consumption keep default value of 0 for this bit.	R	0

Memory Subsystem

This chapter describes the memory subsystem.

Topic **Page**

9.1 General-Purpose Memory Controller (GPMC)

9.1.1 General-Purpose Memory Controller Overview

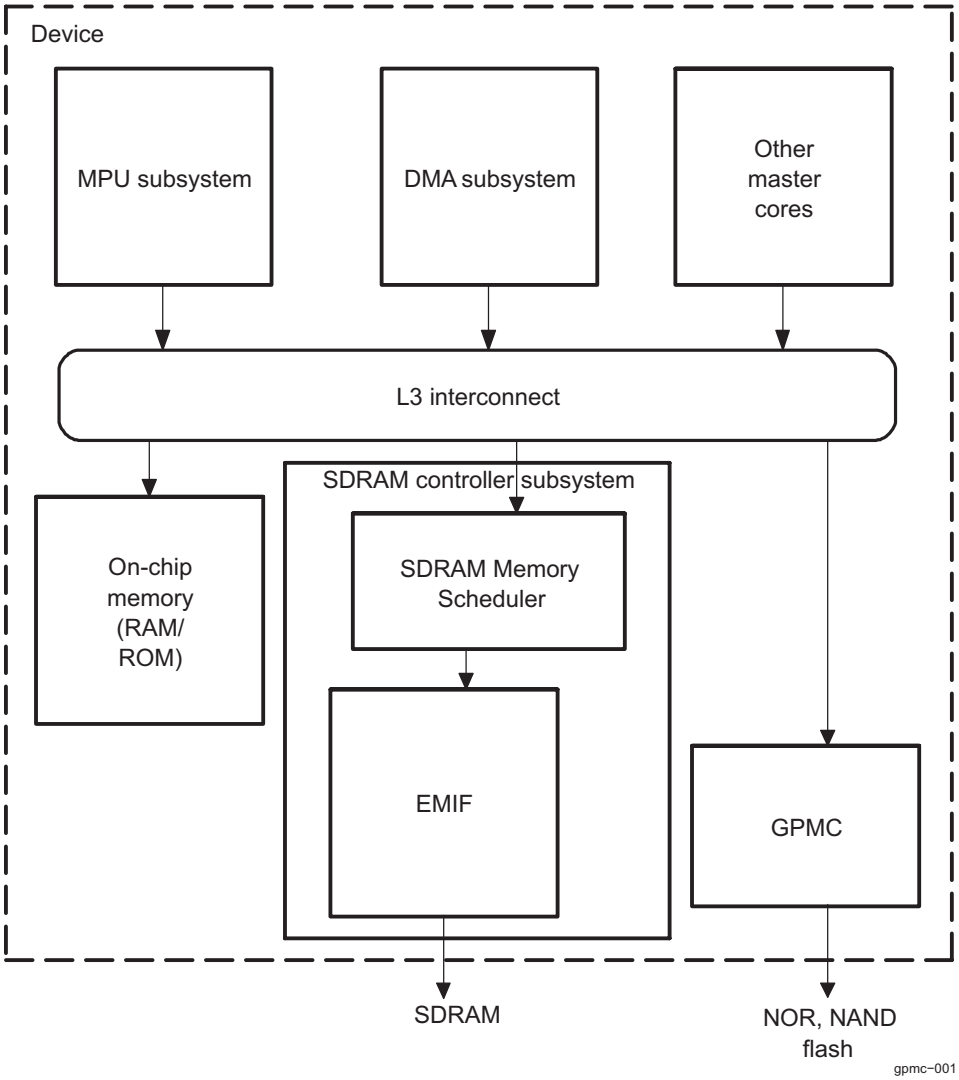
The general-purpose memory controller (GPMC) is dedicated to interfacing external memory devices:

- Asynchronous SRAM-like memories and application-specific integrated circuit (ASIC) devices
- Asynchronous, synchronous, and page mode (only available in non-muxed mode) burst NOR flash devices
- NAND flash
- Pseudo-SRAM devices

NOTE: Page mode is only available in non-muxed mode. The non-muxed mode is described in this chapter even though its use is very limited (address space limited to 2 KBytes).

Figure 9-1 shows the environment of the GPMC.

Figure 9-1. GPMC Environment



9.1.1.1 GPMC Features

The GPMC is a 16-bit external memory controller. The GPMC data access engine provides a flexible programming model for communication with all standard memories. The GPMC supports various accesses:

- Asynchronous read/write access
- Asynchronous read page access (4, 8, 16 Word16)
- Synchronous read/write access
- Synchronous read/write burst access without wrap capability (4, 8, 16 Word16)
- Synchronous read/write burst access with wrap capability (4, 8, 16 Word16)
- Address/data-multiplexed access
- Little- and big-endian access

The GPMC can communicate with a wide range of external devices:

- External asynchronous or synchronous 8-bit wide memory or device
- External asynchronous or synchronous 16-bit wide memory or device
- External 16-bit nonmultiplexed device with limited address range (2 Kbytes)
- External 16-bit address/data-multiplexed NOR flash device
- External 8-bit and 16-bit NAND flash device
- External 16-bit pseudo SRAM (pSRAM) device

The GPMC supports up to eight chip-select regions of programmable size, and programmable base addresses in a total address space of 1 Gbyte.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

9.1.2 GPMC Environment

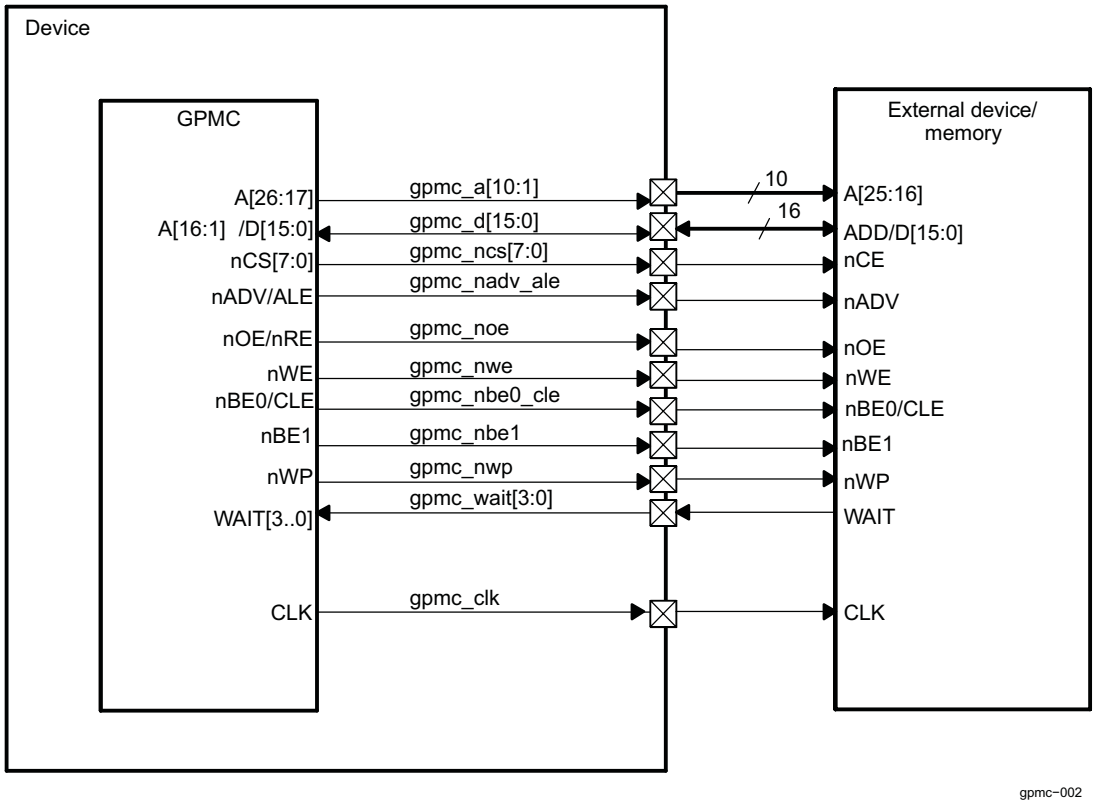
Figure 9-2 and Figure 9-3 show two GPMC external connection options:

- GPMC to 16-bit address/data-multiplexed memory

  Figure 9-2 shows a connection between the GPMC and a 16-bit synchronous address/data-multiplexed external memory device.
- GPMC to 16-bit NAND device

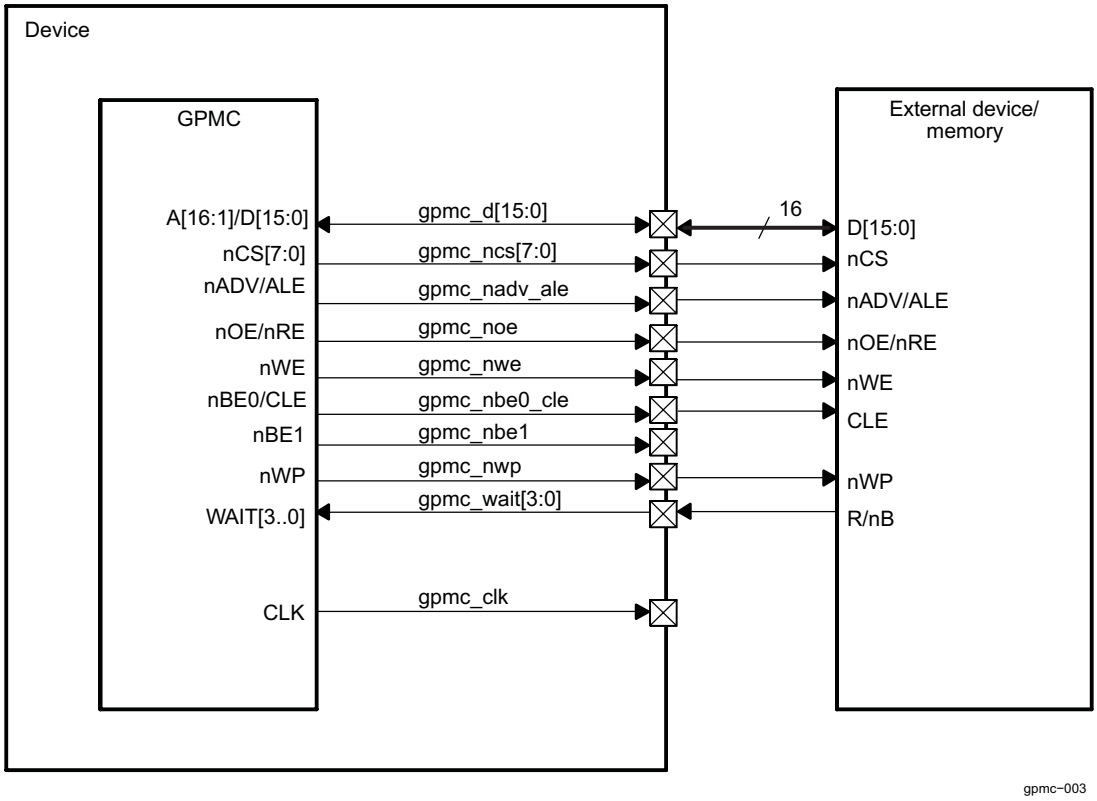
  Figure 9-3 shows a connection between the GPMC and a 16-bit NAND device.

Figure 9-2. GPMC to 16-Bit Address/Data-Multiplexed Memory



NOTE: The device does not provide the A0 byte address line required for random-byte addressable 8-bit wide device interfacing (for multiplexed and nonmultiplexed protocol). Hence, an 8-bit device must be connected to the D[7:0] / gpmc_d[7:0] data bus (rather than D[15:8] / gpmc_d[15:8]) of the GPMC controller. This limits the use of 8-bit wide device interfacing to byte-alias access.

Figure 9-3. GPMC to 16-Bit NAND Device



NOTE: An 8-bit NAND must be connected to the D[7:0] data bus of the GPMC.

Table 9-1 lists the GPMC subsystem I/O pins.

Table 9-1. GPMC I/O Description

Pin Name	I/O	Description
gpmc_a[10:1]	O	Address
gpmc_d[15:0]	I/O	Data
gpmc_ncs[7:0]	O	Chip-selects (active low)
gpmc_clk	I/O	Clock(1)
gpmc_nadv_ale	O	Address valid (active low). Also used as address latch enable (active high) for NAND protocol memories.
gpmc_noe_nre	O	Output enable (active low). Also used as read enable (active low) for NAND protocol memories.
gpmc_nwe	O	Write enable (active low)
gpmc_nbe0_cle	O	Lower-byte enable (active low). Also used as command latch enable for NAND protocol memories.
gpmc_nbe1	O	Byte 1 enable (active low)
gpmc_nwp	O	Write protect (active low)
gpmc_wait[3:0]	I	External wait signal for NOR and NAND protocol memories
gpmc_io_dir	O	gpmc_d[15:0] signal direction control: Low during transmit (for write access: data OUT from GPMC to memory), High during receive (for read access: data IN from memory to GPMC)

(1) This output signal is also used as re-timing input.

Table 9-2 shows the use of address and data GPMC controller pins based on the type of external device.

Table 9-2. GPMC Pin Multiplexing Options

GPMC Pin	Multiplexed Address Data 16-Bit Device	Nonmultiplexed Address Data 16-Bit Device With LIMITED-ADDRESS Bit Enabled	16-Bit NAND Device	8-Bit NAND Device
gpmc_a[10]	A26	A10	Not used	Not used
gpmc_a[9]	A25	A9	Not used	Not used
gpmc_a[8]	A24	A8	Not used	Not used
gpmc_a[7]	A23	A7	Not used	Not used
gpmc_a[6]	A22	A6	Not used	Not used
gpmc_a[5]	A21	A5	Not used	Not used
gpmc_a[4]	A20	A4	Not used	Not used
gpmc_a[3]	A19	A3	Not used	Not used
gpmc_a[2]	A18	A2	Not used	Not used
gpmc_a[1]	A17	A1	Not used	Not used
gpmc_d[15]	A16/D15	D15	D15	Not used
gpmc_d[14]	A15/D14	D14	D14	Not used
gpmc_d[13]	A14/D13	D13	D13	Not used
gpmc_d[12]	A13/D12	D12	D12	Not used
gpmc_d[11]	A12/D11	D11	D11	Not used
gpmc_d[10]	A11/D10	D10	D10	Not used
gpmc_d[9]	A10/D9	D9	D9	Not used
gpmc_d[8]	A9/D8	D8	D8	Not used
gpmc_d[7]	A8/D7	D7	D7	D7
gpmc_d[6]	A7/D6	D6	D6	D6
gpmc_d[5]	A6/D5	D5	D5	D5
gpmc_d[4}	A5/D4	D4	D4	D4
gpmc_d[3]	A4/D3	D3	D3	D3
gpmc_d[2]	A3/D2	D2	D2	D2
gpmc_d[1]	A2/D1	D1	D1	D1
gpmc_d[0]	A1/D0	D0	D0	D0

Enabling the GPMC.GPMC_CONFIG[1] LIMITEDADDRESS bit forces A[26:11] to 1 on the GPMC I/O side. Thus, only devices with 2 Kbytes of addressing space can be accessed using gpmc_a[10:1].

With all device types, the GPMC does not drive unnecessary address lines. They stay at their reset value of 0x00.

Address mapping supports address/data-multiplexed 16-bit wide devices:

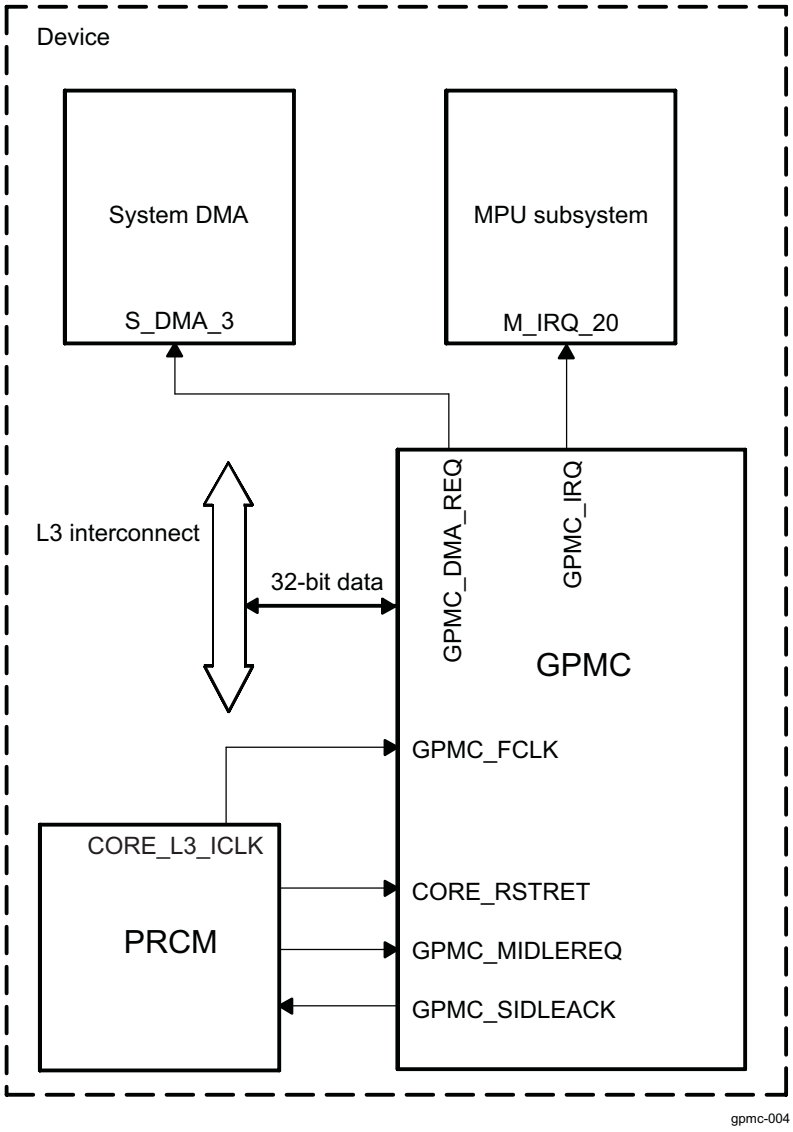
- To minimize the number of IC pins required for the external memory connection, the NOR flash memory controller supports multiplexed address and data memory devices without adding logic externally.
- Multiplexing mode can be selected through the GPMC.GPMC_CONFIG1_i[9] MUXADDDATA bit (I = 0 to 7).
- Asynchronous page mode is not supported for multiplexed address and data devices.

9.1.3 GPMC Integration

9.1.3.1 Description

Figure 9-4 shows how the GPMC interacts with other modules in the processor.

Figure 9-4. GPMC Integration in the Processor



9.1.3.2 Clocking, Reset, and Power Management Scheme

9.1.3.2.1 Clocking

The GPMC use a single clock, GPMC_FCLK, which comes internally from the power, reset, and clock-management (PRCM) module and runs at the L3 interconnect frequency. Its source is the PRCM module, CORE_L3_ICLK output. CORE_L3_ICLK belongs to the L3 interconnect clock domain.

For details, see the *Power, Reset, and Clock Management* chapter.

GPMC_CLK is the external clock provided to the attached synchronous memory or device. The GPMC_CLK clock frequency is the GPMC_FCLK clock frequency divided by 1, 2, 3, or 4, depending on the GPMC.GPMC_CONFIG1_i[1:0] GPMCFCLKDIVIDER bit field (where I = 0 to 7).

NOTE: When the GPMC is configured for synchronous mode, the GPMC_CLK signal (which is an output) must also be set as an input (CONTROL.CONTROL_PADCONF_GPMC_NCS7[24] INPUTENABLE1 = 1). GPMC_CLK is looped back through the output and input buffers of the corresponding GPMC_CLK pad at the device boundary. The looped-back clock is used to synchronize the sampling of the memory signals.

9.1.3.2.2 Hardware Reset

A global reset of the GPMC occurs through activation of the CORE_RSTRET signal (CORE power domain) controlled by the Power, Reset and Clock Management module (see the *Power, Reset, and Clock Management* chapter).
The CORE_RSTRET signal is activated during IC global power-on and global warm reset, and it resets the controller state machine and configuration registers.

9.1.3.2.3 Software Reset

GPMC modules can be reset under software control through the GPMC.GPMC_SYSCONFIG[1] SOFTRESET bit. When software reset bit is set, all registers and the finite state-machine (FSM) are reset immediately and unconditionally. The GPMC_SYSSTATUS[0] RESETDONE bit can be polled to check reset status.

9.1.3.2.4 Power Domain, Power Saving, and Reset Management

GPMC power is supplied by the CORE power domain, and GPMC power management complies with system power-management guidelines.

The GPMC reduces power consumption through auto-idle mode and the idle request/acknowledge process, both of which are configurable:

- Dynamic auto-idle (configurable through the GPMC.GPMC_SYSCONFIG[0] AUTOIDLE bit): To reduce power consumption, the GPMC internally disables the functional clock when no requests are pending and no accesses are ongoing.
- Idle request/acknowledge (one of three idle modes configurable through the GPMC.GPMC_SYSCONFIG[4:3] IDLEMODE field):
  - Force-idle: Immediately on receiving an idle request from the PRCM module, the GPMC sends an idle request/acknowledge to let the PRCM module correctly cut the GPMC source clock.
  - No-idle: The GPMC never goes to idle mode.
  - Smart-idle (strongly recommended): The GPMC goes to idle mode when all ongoing transactions are complete.

For detailed information about power management, see the *Power, Reset, and Clock Management* chapter.

9.1.3.2.5 Hardware Requests

The GPMC uses two hardware requests as shown in Figure 9-4 :

- One interrupt request goes from GPMC (GPMC_IRQ) to the microprocessor unit (MPU) subsystem : M_IRQ_20.
- One DMA request goes from GPMC (GPMC_DMA_REQ) to the system DMA (sDMA) : S_DMA_3.

9.1.3.3 GPMC Address and Data Bus

The current application supports GPMC connection to address/data-multiplexed memory and a NAND device. Connection to a nonmultiplexed address/data memory is supported with an address range of only 2 Kbytes.

Depending on the GPMC configuration on each chip-select, address and data-bus lines that are not required for a particular access protocol are not updated (changed from current value) and are not sampled when input (input data bus).

The current application supports GPMC connection to address/data-multiplexed memory, address/data-nonmultiplexed memory with limited address (2 Kbytes), and a NAND device:

- When the GPMC.GPMC_CONFIG[1] LIMITEDADDRESS bit is set to 1, only gpmc_a[10:1] address lines are used. This limits the memory support to 2K-byte addressable memories.
- For address/data-multiplexed NOR devices, the address is multiplexed on the data bus.
- 8-bit wide NOR devices do not use GPMC I/O: gpmc_d[15:8] for data (they are used for address if needed).
- 16-bit wide NAND devices do not use GPMC I/O: gpmc_a[10:1].
- 8-bit wide NAND devices do not use GPMC I/O: gpmc_a[10:1] and GPMC I/O: gpmc_d[15:8].

CAUTION

Before trying to access a chip-select configured with a nonmultiplexed protocol, set the LIMITEDADDRESS bit control.

9.1.3.3.1 GPMC I/O Configuration Setting (In Default Pinout Mode 0)

NOTE: In this section, the I in GPMC_CONFIG1_i stands for the GPMC chip-select I where I = 0 to 7.

The address/data-nonmultiplexed device, which is limited to a 2K-byte address range, is selected by programming the following register fields:

- GPMC.GPMC_CONFIG1_i[11:10] DEVICETYPE field = 0x00
- GPMC.GPMC_CONFIG1_i[9] MUXADDDATA bit = 0
- GPMC.GPMC_CONFIG[1] LIMITEDADDRESS bit = 1

NOTE: The LIMITEDADDRESS field applies only to address/data-nonmultiplexed devices; it has no effect on other device types (address/data-multiplexed, NAND).

To select the address/data-multiplexed device, program the following register fields:

- GPMC.GPMC_CONFIG1_i[11:10] DEVICETYPE field = 0b00
- GPMC.GPMC_CONFIG1_i[9] MUXADDDATA bit = 1

To select the NAND device, program the following register field:

- GPMC.GPMC_CONFIG1_i[11:10] DEVICETYPE field = 0b10
- GPMC.GPMC_CONFIG1_i[9] MUXADDDATA bit = 0

9.1.3.3.2 GPMC CS0 Default Configuration at IC Reset

To ensure a correct external boot with a GPMC access from IC reset time on CS0, several external pins are sampled:

- The sys_boot[4:0] pins (device boundary) define the sequence of interfaces and devices to use for booting.
- The sys_boot[5] pin defines which group of booting sequences is preferred: memory booting (sys_boot[5] = 0) or peripheral booting (sys_boot[5] = 1).
- Three additional pins are used to configure reset values in the GPMC.GPMC_CONFIG1_i register (where I = 0):
  - The bootwaiten input pin (GPMC boundary) enables the monitoring on chip-select 0 of the WAIT pin at IC reset release time for read accesses. The input pin is used to configure the GPMC.GPMC_CONFIG1_i[22] WAITREADMONITORING bit (where I = 0). Its value comes from the BOOT_WAIT_ENABLE signal generated by the system control module (SCM). When sys_boot[5:0] = 0b111111, the BOOT_WAIT_ENABLE signal is activated, causing the wait pin to be monitored for read access.
  - The bootdevicesize input pin (GPMC boundary) defines the size of the attached device on chip-select 0 and is used to configure the GPMC.GPMC_CONFIG1_i[13:12] DEVICESIZE bits (where I = 0). A BOOT_DEVICE_SIZE signal is propagated from the SCM. Its value is fixed at 0x1 at IC reset, causing a 16-bit wide external memory to be used.
  - The cs0muxdevice input pin (GPMC boundary) selects whether the attached device to chip-select 0 is a multiplexed address and data device or not. The input pin is used to configure the GPMC.GPMC_CONFIG1_i[9] MUXADDDATA bit (where I = 0). A CS0_MUX_DEVICE signal is propagated from the SCM. Its value is fixed at 0x1 at IC reset, causing the attached device to be address/data-multiplexed.
  - The waitselectpin input pin selects the WAIT signal at IC reset release time between WAIT0 input pin or WAIT1 input pin. At IC reset release time, these two pins have different polarity.

CAUTION

Using the internal boot code, the entire CS0 configuration can be modified before the first CS0 access. This modification of internal boot code is necessary for two external devices:

- NAND device attached to CS0
- Nonmultiplexed 2 Kbyte address range device attached to CS0

At reset time, the IC may boot from the internal ROM or from the memory attached to the GPMC chip-select 0. This selection is made outside the GPMC.

Reset values of the timing control parameters are defined to cope with direct boot on address and data multiplexed NOR Flash device, on non-multiplexed NOR Flash device or on any asynchronous device with large timing margins assuming a low GPMC_FCLK frequency (for example, 19.2Mhz) at boot time.

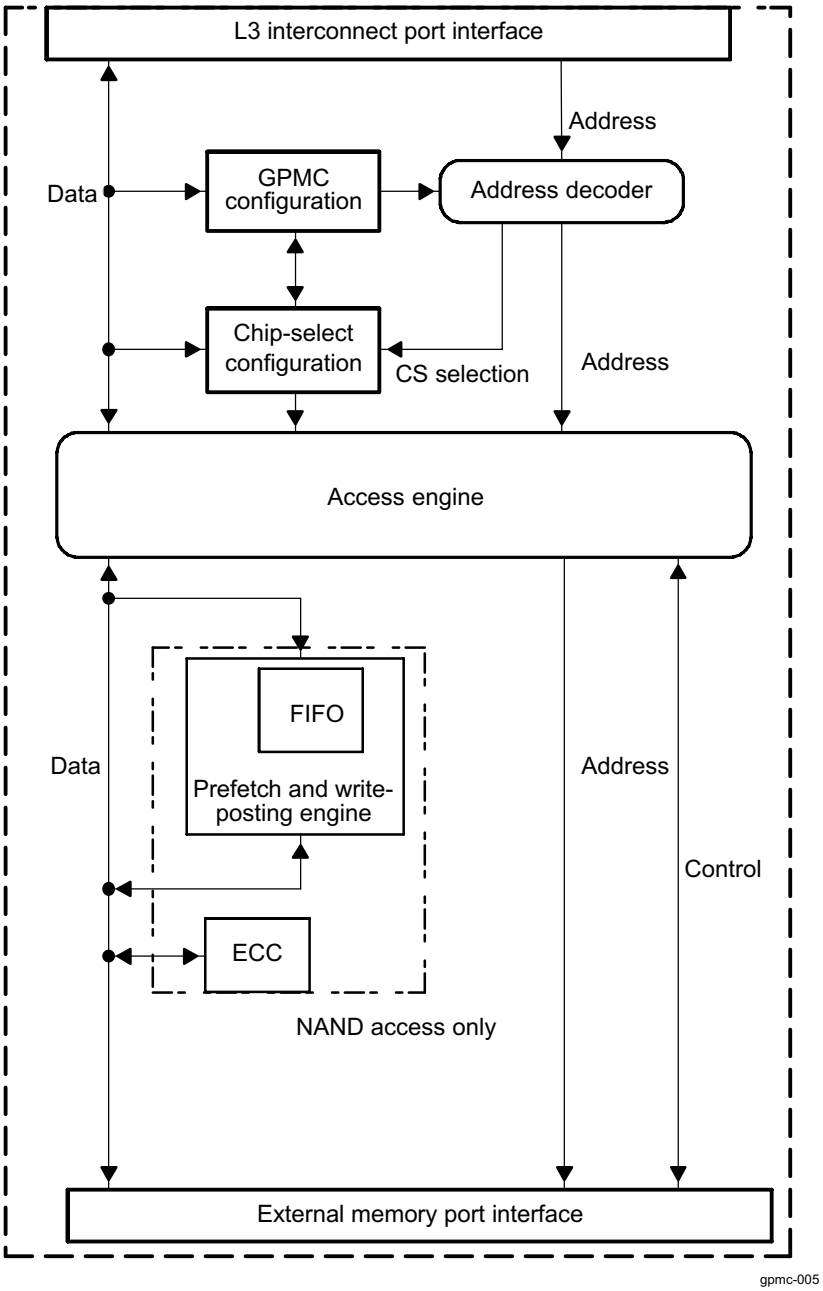
9.1.4 GPMC Functional Description

9.1.4.1 Description

As Figure 9-5 shows, the GPMC consists of six blocks:

- L3 interconnect port interface
- Address decoder, GPMC configuration, and chip-select configuration register file
- Access engine
- Prefetch and write-posting engine
- Error correction code engine (ECC)
- External device/memory port interface

Figure 9-5. GPMC Functional Diagram



The GPMC can access various external devices through the L3 Interconnect. The flexible programming model allows a wide range of attached device types and access schemes.

Based on the programmed configuration bit fields stored in the GPMC registers, the GPMC is able to generate all control signals timing depending on the attached device and access type.

Given the chip-select decoding and its associated configuration registers, the GPMC selects the appropriate device type control signals timing.

9.1.4.2 L3 Interconnect Interface

The GPMC L3 interconnect interface is a pipelined interface including an 8 × 32-bit word write buffer.

Any device system host can issue external access requests through the GPMC.

The device system can issue the following requests through this interface:

- One 8-bit / 16-bit / 32-bit interconnect access (read/write)
- Two incrementing 32-bit interconnect accesses (read/write)
- Two wrapped 32-bit interconnect accesses (read/write)
- Four incrementing 32-bit interconnect accesses (read/write)
- Four wrapped 32-bit interconnect accesses (read/write)
- Eight incrementing 32-bit interconnect accesses (read/write)
- Eight wrapped 32-bit interconnect accesses (read/write)

Only linear burst transactions are supported; interleaved burst transactions are not supported. Only power-of-two-length precise bursts 2 × 32, 4 × 32, or 8 × 32 with the burst base address aligned on the total burst size are supported (this limitation applies to incrementing bursts only).

This interface also provides one interrupt and one DMA request line, for specific event control.

It is recommended to program the ATTACHEDDEVICEPAGELENGTH field (GPMC_CONFIG1_i[24:23]) according to the effective attached device page length and to enable WRAPBURST bit (GPMC_CONFIG1_i[31]) if the attached device supports wrapping burst.

However, it is possible to emulate wrapping burst on a non-wrapping memory by providing relevant addresses within the page or splitting transactions. Bursts larger than the memory page length are chopped into multiple bursts transactions. Due to the alignment requirements, a page boundary is never crossed.

9.1.4.3 Address Decoder, GPMC Configuration, and Chip-Select Configuration Register File

Address-decoding logic selects for chip-selects according to the address request and the content of the chip-select base address register file, which includes a set of global GPMC configuration registers and eight sets of chip-select configuration registers.

The GPMC configuration register file is memory-mapped and can be read or written with byte, 16-bit word, or 32-bit word accesses. The register file should be configured as a noncacheable, nonbufferable region to prevent any desynchronization between host execution (write request) and the completion of register configuration (write completed with register updated). Section 9.1.7 of this chapter provides the GPMC register locations. For the map of GPMC memory locations, see the *Memory Mapping* chapter.

After the chip-select is configured, the access engine accesses the external device, drives the external interface control signals, and applies the interface protocol based on user-defined timing parameters and settings.

9.1.4.4 Error Correction Code Engine (ECC)

The GPMC includes an ECC calculation engine that allows ECC calculation during data read or data program (write) operations.Two ECC algorithms are available depending on GPMC_ECC_CONFIG[16] ECCALGORITHM settings: Hamming code or BCH code (Bose-ChaudhurI-Hocquenghem).

The GPMC does not directly handle the error code correction itself. During writes, the GPMC computes parity bits. During reads, the GPMC provides enough information for the processor to correct errors without reading the data buffer all over again.

The Hamming code ECC is based on a 2-dimensional (row and column) bit parity accumulation. This parity accumulation is either accomplished on the programmed number of bytes or Word16s read from the memory device, or written to the memory device in stream mode.

Because the ECC engine includes only one accumulation context, it can be allocated to only one chip-select at a time through the GPMC.GPMC_ECC_CONFIG[3:1] ECCCS bit field.

Refer to Section 9.1.5.14.3, for more information on ECC Calculation.

9.1.4.5 Prefetch and Write-Posting Engine

The prefetch and write-posting engine is a simplified embedded-access requester that presents requests to the access engine on a user-defined chip-select target. The access engine interleaves these requests with any request coming from the L3 interface; as a default the prefetch and write-posting engine has the lowest priority.

The prefetch and write-posting engine is dedicated to data-stream access (as opposed to random data access); thus, it is primarily dedicated to NAND support. The engine does not include an address generator; the request is limited to chip-select target identification. It includes a 64-byte FIFO associated with a DMA request synchronization line, for optimal DMA-based use.

For more information about prefetch and write-posting engine programming, see Section 9.1.5.14.4, *Prefetch and Write-Posting Engine*.

9.1.4.6 External Device/Memory Port Interface

The external port interface controls all address, data, and control signals required for communication with GPMC-supported devices and memories.

9.1.5 GPMC Basic Programming Model

The GPMC basic programming model offers maximum flexibility to support various access protocols for each of the eight configurable chip-selects. Use optimal chip-select settings, based on the characteristics of the external device:

- Different protocols can be selected to support generic asynchronous or synchronous random-access devices (NOR flash, SRAM) or to support specific NAND devices.
- The address and the data bus can be multiplexed on the same external bus.
- Read and write access can be independently defined as asynchronous or synchronous.
- System requests (byte, Word16, burst) are performed through single or multiple accesses. External access profiles (single, multiple with optimized burst length, native- or emulated-wrap) are based on external device characteristics (supported protocol, bus width, data buffer size, native-wrap support).
- System burst read or write requests are synchronous-burst (multiple-read or multiple-write). When neither burst nor page mode is supported by external memory or ASIC devices, system burst read or write requests are translated to successive single synchronous or asynchronous accesses (single reads or single writes). 8-bit wide devices are supported only in single-synchronous or asynchronous read or write mode.
- To simulate a programmable internal-wait state, an external wait pin can be monitored to dynamically control external access at the beginning (initial access time) of and during a burst access.

Each control signal is controlled independently for each chip-select. The internal functional clock of the GPMC (GPMC_FCLK) is used as a time reference to specify the following:

- Read- and write-access duration
- Most GPMC external interface control-signal assertion and deassertion times
- Data-capture time during read access
- External wait-pin monitoring time
- Duration of idle time between accesses, when required

9.1.5.1 Chip-Select Base Address and Region Size Configuration

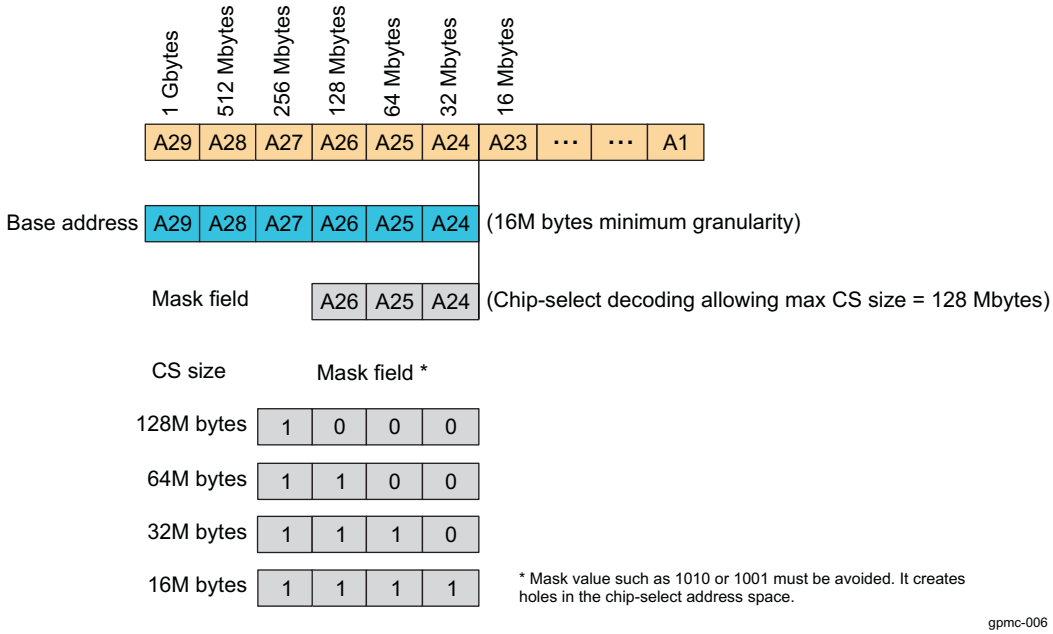
Any external memory or ASIC device attached to the GPMC external interface can be accessed by any device system host within the GPMC 1 Gbyte contiguous address space. For details, see the *Memory Mapping* chapter.

The GPMC 1 Gbyte address space can be divided into a maximum of eight chip-select regions with programmable base address and programmable CS size. The CS size is programmable from 16 MBytes to 256 MBytes (must be a power-of-2) and is defined by the mask field. Attached memory smaller than the programmed CS region size is accessed through the entire CS region (aliasing).

Each chip-select has a 6-bit base address encoding and a 4-bit decoding mask, which must be programmed according to the following rules:

- The programmed chip-select region base address must be aligned on the chip-select region size address boundary and is limited to a power-of-2 address value. During access decoding, the register base address value is used for address comparison with the address-bit line mapping as described in Figure 9-6 (with A0 as the device system byte-address line). Base address is programmed through the GPMC_CONFIG7_i[5:0] BASEADDRESS bit field
- The register mask is used to exclude some address lines from the decoding. A register mask bit field set to 0 suppresses the associated address line from the address comparison (incoming address bit line is don't care). The register mask value must be limited to the subsequent value, based on the desired chip-select region size. Any other value has an undefined result. When multiple chip-select regions with overlapping addresses are enabled concurrently, access to these chip-select regions is cancelled and a GPMC access error is posted. The mask field is programmed through the GPMC_CONFIG7_i[11:8] MASKADDRESS bit field.

Figure 9-6. Chip-Select Address Mapping and Decoding Mask



Chip-select configuration (base and mask address or any protocol and timing settings) must be performed while the associated chip-select is disabled through the GPMC.GPMC_CONFIG7_i[6] CSVALID bit (where I stands for the GPMC chip-select value, I = 0 to 7). In addition, a chip-select configuration can only be disabled if there is no ongoing access to that chip-select. This requires activity monitoring of the prefetch or write-posting engine if the engine is active on the chip-select. Also, the write buffer state must be monitored to wait for any posted write completion to the chip-select.

Conversely, before trying to access a chip-select, software must ensure that the chip-select is enabled. To account for prefetch engine effects, after the chip-select-enable instruction, an NOP instruction (equivalent to 64 bits) must be executed before the chip-select is accessed.

Any access attempted to a nonvalid GPMC address region (CSVALID disabled or address decoding outside a valid chip-select region) is not propagated to the external interface and a GPMC access error is posted. In case of chip-selects overlapping, an error is generated and no access will occur on either chip-select.

Chip-select 0 is the only chip-select region enabled after either a power-up or a GPMC reset.

CAUTION

Although the GPMC interface can drive up to 8 chip-selects, the frequency specified for this interface is for a specific load. If this load is exceeded, the maximum frequency cannot be reached. One solution is to implement a board with buffers, to allow the slowest device to maintain the total load on the lines at the value specified in your device-specific data manual. To have access to the device-specific data manual, please contact your TI representative.

9.1.5.2 Access Protocol Configuration

9.1.5.2.1 Supported Devices

The access protocol of each chip-select can be independently specified through the GPMC.GPMC_CONFIG1_i[11:10] DEVICETYPE parameter (where I =0 to 7) for:

- Random-access synchronous or asynchronous memory like NOR flash, SRAM
- NAND flash asynchronous devices

NOTE: NAND flash interfacing requires the parameter settings of generic chip-select 0. For more information about the NAND flash GPMC basic programming model and NAND support, see Section 9.1.5.14, *NAND Device Basic Programming Model*, and Section 9.1.5.14.1, *NAND Memory Device in Byte or Word 16 Stream Mode*.

9.1.5.2.2 Access Size Adaptation and Device Width

Each chip-select can be independently configured through the GPMC.GPMC_CONFIG1_i[13:12] DEVICESIZE field (I = 0 to 7) to interface with a 16-bit wide device or an 8-bit wide device. System requests with data width greater than the external device data bus width are split into successive accesses according to both the external device data-bus width and little-endian data organization.

NOTE: The processor does not provide the A0 byte address line required for random-byte addressable 8-bit wide device interfacing (for both multiplexed and nonmultiplexed protocol). It limits the use of 8-bit wide device interfacing to byte-alias accesses. This limitation is not applicable to NAND device interfacing (8-bit wide or 16-bit wide devices).

9.1.5.2.3 Address/Data-Multiplexing Interface

For random synchronous or asynchronous memory interfacing (DEVICETYPE = 0b00), an address- and data-multiplexing protocol can be selected through the GPMC.GPMC_CONFIG1_i[9] MUXADDDATA bit (I = 0 to 7). The nADV signal must be used as the external device address latch control signal. For the associated chip-select configuration, nADV assertion and deassertion time and nOE assertion time must be set to the appropriate value to meet the address latch setup/hold time requirements of the external device. See Section 9.1.3, *GPMC Integration*.

NOTE: This address/data-multiplexing interface is not applicable to NAND device interfacing. NAND devices require a specific address, command, and data multiplexing protocol. See Section 9.1.5.14, *NAND Device Basic Programming Model*.

9.1.5.2.4 Address and Data Bus

See Section 9.1.3.3, *GPMC Address and Data Bus*.

9.1.5.2.5 Asynchronous and Synchronous Access

For each chip-select configuration, the read access can be specified as either asynchronous or synchronous access through the GPMC.GPMC_CONFIG1_i[29] READTYPE bit (I = 0 to 7). For each chip-select configuration, the write access can be specified as either synchronous or asynchronous access through the GPMC.GPMC_CONFIG1_i[27] WRITETYPE bit (I = 0 to 7).

Asynchronous and synchronous read (write) access time and related control signals are controlled through timing parameters that refer to GPMC_FCLK. The primary difference of synchronous mode is the availability of a configurable clock interface (GPMC_CLK) to control the external device. Synchronous mode also affects data-capture and wait-pin monitoring schemes in read access.

For details about asynchronous and synchronous access, see the descriptions of GPMC_CLK, RdAccessTime, WrAccessTime, and wait-pin monitoring.

For more information about timing-parameter settings, see the sample timing diagrams in this chapter.

NOTE: The address bus and nBE[1:0] are fixed for the duration of a synchronous burst read access, but they are updated for each beat of an asynchronous page-read access.

9.1.5.2.6 Page and Burst Support

Each chip-select can be configured to process system single or burst requests into successive single accesses or asynchronous page/synchronous burst accesses, with appropriate access size adaptation.

Depending on the external device page or burst capability, read and write accesses can be independently configured through the GPMC. The GPMC_CONFIG1_i[30] READMULTIPLE and GPMC.GPMC_CONFIG1_i[28] WRITMULTIPLE bits (I = 0 to 7) are associated with the READTYPE and WRITETYPE parameters.

NOTE:

- Asynchronous write page mode is not supported.
- 8-bit wide device support is limited to nonburstable devices (READMULTIPLE and WRITEMULTIPLE are don't care).
- Not applicable to NAND device interfacing.

9.1.5.2.7 System Burst Versus External Device Burst Support

The device system can issue the following requests to the GPMC:

- Byte, Word16, Word32 requests (byte enable controlled). This is always a single request from the interconnect point of view.
- Incrementing fixed-length bursts of two words, four words, and eight words
- Wrapped (critical word access first) fixed-length burst of two, four, or eight words

To process a system request with the optimal protocol, the READMULTIPLE (and READTYPE) and WRITEMULTIPLE (and WRITETYPE) parameters must be set according to the burstable capability (synchronous or asynchronous) of the attached device.

The GPMC access engine issues only fixed-length burst. The maximum length that can be issued is defined per CS by the GPMC.GPMC_CONFIG1_i[24:23] ATTACHEDDEVICEPAGELENGTH field (I = 0 to 7). When the ATTACHEDDEVICEPAGELENGTH value is less than the system burst request length (including the appropriate access size adaptation according to the device width), the GPMC splits the system burst request into multiple burst beats. Within the specified 4-, 8-, or 16-word value, the ATTACHEDDEVICEPAGELENGTH field value must correspond to the maximum-length burst supported by the memory device configured in fixed-length burst mode (as opposed to continuous burst mode).

To get optimal performance from memory devices that natively support 16 Word16-length-wrapping burst capability (critical word access first), the ATTACHEDDEVICEPAGELENGTH parameter must be set to 16 words and the GPMC.GPMC_CONFIG1_i[31] WRAPBURST bit (I = 0 to 7) must be set to 1. Similarly DEVICEPAGELENGTH is set to 4 and 8 for memories supporting respectively 4 and 8 Word16-length-wrapping burst.

When the memory device does not offer (or is not configured to offer) native 16 Word16-length-wrapping burst, the WRAPBURST parameter must be cleared, and the GPMC access engine emulates the wrapping burst by issuing the appropriate burst sequences according to the ATTACHEDDEVICEPAGELENGTH value.

When the memory device does not support native-wrapping burst, there is usually no difference in behavior between a fixed burst length mode and a continuous burst mode configuration (except for a potential power increase from a memory-speculative data prefetch in a continuous burst read). However, even though continuous burst mode is compatible with GPCM behavior, because the GPMC access engine issues only fixed-length burst and does not benefit from continuous burst mode, it is best to configure the memory device in fixed-length burst mode.

The memory device maximum-length burst (configured in fixed-length burst wrap or nonwrap mode) usually corresponds to the memory device data buffer size. Memory devices with a minimum of 16 half-word buffers are the most appropriate (especially with wrap support), but memory devices with smaller buffer size (4 or 8) are also supported, assuming that the GPMC.GPMC_CONFIG1_i[24:23] ATTACHEDDEVICEPAGELENGTH field is set accordingly to 4 or 8 words.

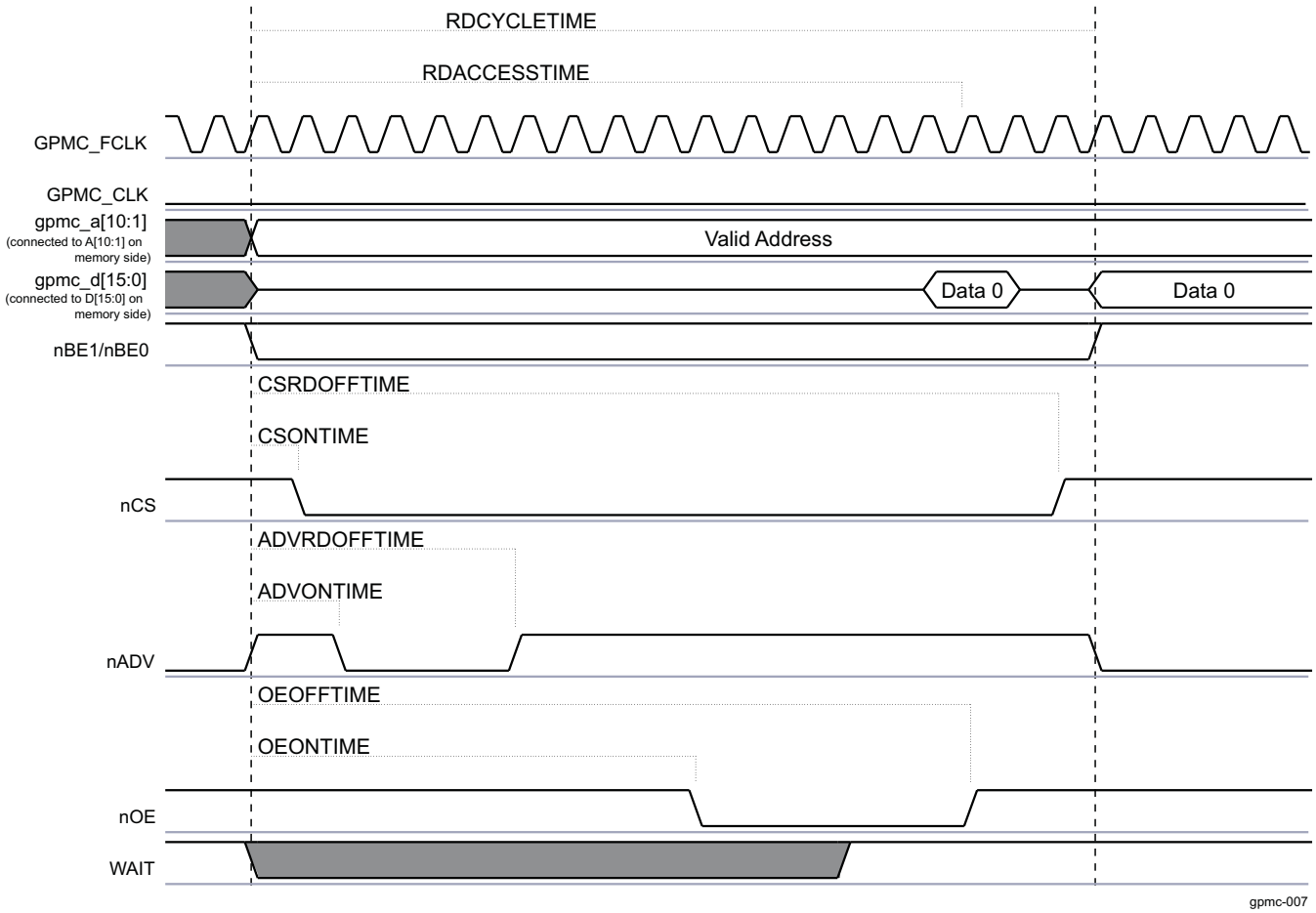
The device system issues only requests with addresses or starting addresses for nonwrapping burst requests; that is, the request size boundary is aligned. In case of an eight-word-wrapping burst, the wrapping address always occurs on the eight-words boundary. As a consequence, all words requested must be available from the memory data buffer when the buffer size is equal to or greater than the ATTACHEDDEVICEPAGELENGTH value. This usually means that data can be read from or written to the buffer at a constant rate (number of cycles between data) without wait states between data accesses. If the memory does not behave this way (nonzero wait state burstable memory), wait-pin monitoring must be enabled to dynamically control data-access completion within the burst beat.

NOTE: When the system burst request length is less than the ATTACHEDDEVICEPAGELENGTH value, the GPMC proceeds with the required accesses.

9.1.5.3 Timing Setting

The GPMC is a signal generator that offers the maximum flexibility to support various access protocols. Most of the timing parameters of the protocol access used by the GPMC to communicate with attached memories or devices are programmable on a chip-select basis. Assertion and deassertion times of control signals are defined to match the attached memory or device timing specifications and to get maximum performance during accesses. For example, the timing diagram in Figure 9-7 shows an asynchronous single-read access performed on an asynchronous device. For more information on GPMC_CLK and GPMC_FCLK refer to Section 9.1.5.3.6.

Figure 9-7. Asynchronous Single Read on a Nonmultiplexed Address/Data Device



9.1.5.3.1 Read Cycle Time and Write Cycle Time (RDCYCLETIME / WRCYCLETIME)

The GPMC.GPMC_CONFIG5_i[4:0] RDCYCLETIME and GPMC.GPMC_CONFIG5_i[12:8] WRCYCLETIME fields (I = 0 to 7) define the address bus and byte enables valid times for read and write accesses. To ensure a correct duty cycle of GPMC_CLK between accesses, RDCYCLETIME and WRCYCLETIME are expressed in GPMC_FCLK cycles and must be multiples of the GPMC_CLK cycle.

When either RDCYCLETIME or WRCYCLETIME completes, if they are not already deasserted, all control signals (NCS, nADV/ALE, nOE/RE, nWE, and BE0/CLE) are deasserted to their reset values, regardless of their deassertion time parameters.

An exception to this forced deassertion occurs when a pipelined request to the same chip-select or to a different chip-select is pending. In such a case, it is not necessary to deassert a control signal with deassertion time parameters equal to the cycle-time parameter. This exception to forced deassertion prevents any unnecessary glitchy transition. This requirement also applies to BE signals, thus avoiding an unnecessary BE glitch transition when pipelining requests.

If no inactive cycles are required between successive accesses to the same or to a different chip-select (GPMC.GPMC_CONFIG6_i[7] CYCLE2CYCLESAMECSEN = 0 or GPMC.GPMC_CONFIG6_i[6] CYCLE2CYCLEDIFFCSEN = 0, where I = 0 to 7), and if assertion-time parameters associated with the pipelined access are equal to 0, asserted control signals (nCS, nADV/ALE, nBE0/CLE, nWE, and nOE/RE) are kept asserted. This applies to any read/write to read/write access combination.

If inactive cycles are inserted between successive accesses, that is, CYCLE2CYCLESAMECSEN = 1 or CYCLE2CYCLEDIFFCSEN = 1, the control signals are forced to their respective default reset values for the number of GPMC_FCLK cycles defined in CYCLE2CYCLEDELAY:

- The RDCYCLETIME and WRCYCLETIME bit fields are programmable in the GPMC.GPMC_CONFIG5_i register, I = 0 to 7.
- The RDCYCLETIME and WRCYCLETIME bit fields can be set from 0 to 31 GPMC_FCLK cycles with a granularity of 1 for GPMC.GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY set to 0.
- The RDCYCLETIME and WRCYCLETIME bit fields can be set from 0 to 62 GPMC._FCLK cycles with a granularity of 2 for GPMC.GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY set to 1.

9.1.5.3.2 nCS: Chip-Select Signal Control Assertion/Deassertion Time (CSONTIME / CSRDOFFTIME / CSWROFFTIME / CSEXTRADELAY)

The GPMC.GPMC_CONFIG2_i[3:0] CSONTIME field (where I = 0 to 7) defines the nCS signal-assertion time relative to the start access time. It is common for read and write accesses.

For a read access, the GPMC.GPMC_CONFIG2_i[12:8] CSRDOFFTIME field defines the nCS signal deassertion time relative to start access time.

For a write access, the GPMC.GPMC_CONFIG2_i[20:16] CSWROFFTIME field defines the nCS signal deassertion time relative to start access time.\

CSONTIME, CSRDOFFTIME and CSWROFFTIME parameters are applicable to synchronous and asynchronous modes. CSONTIME can be used to control an address and byte enable setup time before chip-select assertion. CSRDOFFTIME and CSWROFFTIME can be used to control an address and byte enable hold time after chip-select deassertion.

nCS signal transitions as controlled through CSONTIME, CSRDOFFTIME, and CSWROFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC.GPMC_CONFIG2_i[7] CSEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on the nCS assertion and deassertion time to guarantee proper setup and hold time relative to GPMC_CLK. CSEXTRADELAY is especially useful in configurations where GPMC_CLK and GPMC_FCLK have the same frequency, but can be used for all GPMC configurations. If asserted, CSEXTRADELAY applies to all parameters controlling nCS transitions.

The CSEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than the nCS signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

9.1.5.3.3 nADV/ALE: Address Valid/Address Latch Enable Signal Control Assertion/Deassertion Time (ADVONTIME / ADVRDOFFTIME / ADVWROFFTIME / ADVEXTRADELAY)

The GPMC.GPMC_CONFIG3_i[3:0] ADVONTIME field (where I = 0 to 7) defines the nADV/ALE signal-assertion time relative to start access time. It is common to read and write accesses.

For a read access, the GPMC.GPMC_CONFIG3_i[12:8] ADVRDOFFTIME field defines the nADV/ALE signal-deassertion time relative to start access time.

For a write access, the GPMC.GPMC_CONFIG3_i[20:16] ADVWROFFTIME field defines the nADV/ALE signal-deassertion time relative to start access time.

ADVONTIME can be used to control an address and byte enable valid setup time control before nADV/ALE assertion. ADVRDOFFTIME and ADVWROFFTIME can be used to control an address and byte enable valid hold time control after nADV/ALE de-assertion. ADVRDOFFTIME and ADVWROFFTIME are applicable to both synchronous and asynchronous modes.

nADV/ALE signal transitions as controlled through ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC.GPMC_CONFIG3_i[7] ADVEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on nADV/ALE assertion and deassertion time to guarantee proper setup and hold time relative to GPMC_CLK. The ADVEXTRADELAY configuration parameter is especially useful in configurations where GPMC_CLK and GPMC_FCLK have the same frequency, but can be used for all GPMC configurations. If asserted, ADVEXTRADELAY applies to all parameters controlling nADV/ALE transitions.

ADVEXTRADELAY must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than nADV/ALE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

Refer to Section 9.1.5.14 for more details on ADVONTIME, ADVRDOFFTIME and ADVWROFFTIME usage for CLE and ALE (Command / Address Latch Enable) usage for a NAND Flash interface.

9.1.5.3.4 nOE/nRE: Output Enable / Read Enable Signal Control Assertion / Deassertion Time (OEONTIME / OEOFFTIME / OEEXTRADELAY)

The GPMC.GPMC_CONFIG4_i[3:0] OEONTIME field (where I = 0 to 7) defines the nOE/nRE signal assertion time relative to start access time. It is applicable only to read accesses.

The GPMC.GPMC_CONFIG4_i[12:8] OEOFFTIME field defines the nOE/nRE signal deassertion time relative to start access time. It is applicable only to read accesses.

OEONTIME and OEOFFTIME parameters are applicable to synchronous and asynchronous modes. OEONTIME can be used to control an address and byte enable valid setup time control before nOE/nRE assertion. OEOFFTIME can be used to control an address and byte enable valid hold time control after nOE/nRE assertion.

The nOE/RE signal transitions as controlled through OEONTIME, and OEOFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC.GPMC_CONFIG4_i[7] OEEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on nOE/RE assertion and deassertion time to guaranty proper setup and hold time relative to GPMC_CLK. If asserted, OEEXTRADELAY applies to all parameters controlling nOE/nRE transitions.

OEEXTRADELAY must be used carefully, to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program RDCYCLETIME and WRCYCLETIME to be greater than nOE/RE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

nOE/nRE is not asserted during a write cycle.

NOTE: When the GPMC generates a read access to an address-/data-multiplexed device, it drives the address bus until nOE assertion time.

9.1.5.3.5 nWE: Write Enable Signal Control Assertion / Deassertion Time (WEONTIME / WEOFFTIME /

WEEXTRADELAY)

The GPMC.GPMC_CONFIG4_i[19:16] WEONTIME field (where I = 0 to 7) defines the nWE signal-assertion time relative to start access time. It applies only to write accesses.

The GPMC.GPMC_CONFIG4_i[28:24] WEOFFTIME field defines the nWE signal-deassertion time relative to start access time. It applies only to write accesses.

WEONTIME can be used to control an address and byte enable valid setup time control before nWE assertion. WEOFFTIME can be used to control an address and byte enable valid hold time control after nWE assertion.

nWE signal transitions as controlled through WEONTIME, and WEOFFTIME can be delayed by half a GPMC_FCLK period by enabling the GPMC.GPMC_CONFIG4_i[23] WEEXTRADELAY bit. This half of a GPMC_FCLK period provides more granularity on nWE assertion and deassertion time to guaranty proper setup and hold time relative to GPMC_CLK. If asserted, WEEXTRADELAY applies to all parameters controlling nWE transitions.

The WEEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the WRCYCLETIME bit field to be greater than the nWE signal-deassertion time, including the extra half-GPMC_FCLK-period delay.

nWE is not asserted during a read cycle.

9.1.5.3.6 GPMC_CLK

GPMC_CLK is the external clock provided to the attached synchronous memory or device.

- The GPMC_CLK clock frequency is the GPMC_FCLK functional clock frequency divided by 1, 2, 3, or 4, depending on the GPMC.GPMC_CONFIG1_i[1:0] GPMCFCLKDIVIDER bit field (where I = 0 to 7), with a guaranteed 50-percent duty cycle.
- The GPMC_CLK clock is only activated when the access in progress is defined as synchronous (read or write access).
- The GPMC.GPMC_CONFIG1_i[26:25] CLKACTIVATIONTIME field (I = 0 to 7) defines the number of GPMC_FCLK cycles from start access time to GPMC_CLK activation.
- The GPMC_CLK clock is stopped when cycle time completes and is asserted low between accesses.
- The GPMC_CLK clock is kept low when access is defined as asynchronous.
- When cycle time completes, the GPMC_CLK may be high because of the GPMCFCLKDIVIDER bit field. To ensure correct stoppage of the GPMC_CLK clock within the 50-percent required duty cycle, it is the user's responsibility to extend the RDCYCLETIME or WRCYCLETIME value.
- When the GPMC is configured for synchronous mode, the GPMC_CLK signal (which is an output) must also be set as an input (CONTROL.CONTROL_PADCONF_GPMC_NCS7[24] INPUTENABLE1 = 1). GPMC_CLK is looped back through the output and input buffers of the corresponding GPMC_CLK pad at the device boundary. The looped-back clock is used to synchronize the sampling of the memory signals.

NOTE: To ensure a correct external clock cycle, the following rules must be applied:

- (RDCYCLETIME CLKACTIVATIONTIME) must be a multiple of (GPMCFCLKDIVIDER + 1).
- The PAGEBURSTACCESSTIME value must be a multiple of (GPMCFCLKDIVIDER + 1).

9.1.5.3.7 GPMC_CLK and Control Signals Setup and Hold

Control-signal transition (assertion and deassertion) setup and hold values with respect to the GPMC_CLK edge can be controlled in the following ways:

- For the GPMC_CLK signal, the GPMC.GPMC_CONFIG1_i[26:25] CLKACTIVATIONTIME field (I = 0 to 7) allows setup and hold control of control-signal assertion time.
- The use of a divided GPMC_CLK allows setup and hold control of control-signal assertion and deassertion times.

- When GPMC_CLK runs at the GPMC_FCLK frequency so that GPMC_CLK edge and control-signal transitions refer to the same GPMC_FCLK edge, the control-signal transitions can be delayed by half of a GPMC_FCLK period to provide minimum setup and hold times. This half-GPMC_FCLK delay is enabled with the CSEXTRADELAY, ADVEXTRADELAY, OEEXTRADELAY, or WEEXTRADELAY parameter. This delay must be used carefully to prevent control-signal overlap between successive accesses to different chip-selects. This implies that the RDCYCLETIME and WRCYCLETIME are greater than the last control-signal deassertion time, including the extra half-GPMC_FCLK cycle.

9.1.5.3.8 Access Time (RDACCESSTIME / WRACCESSTIME)

The read access time and write access time durations can be programmed independently allowing nOE and GPMC data capture timing parameters to be independent of nWE and memory device data capture timing parameters.
RDACCESSTIME is programmed in the GPMC.GPMC_CONFIG5_i[20:16] bit field (I = 0 to 7).
WRACCESSTIME is programmed in the GPMC.GPMC_CONFIG6_i[28:24] bit field (I = 0 to 7).
RDACCESSTIME and WRACCESSTIME can be set from 0 to 31 GPMC_FCLK cycles with a granularity of one (GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY = 0).
RDACCESSTIME and WRACCESSTIME can be set from 0 to 62 GPMC_FCLK cycles with a granularity of two (GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY = 1).

9.1.5.3.8.1 Access Time on Read Access

In asynchronous read mode, for single and paged accesses, RDACCESSTIME field (I = 0 to 7) defines the number of GPMC_FCLK cycles from start access time to the GPMC_FCLK rising edge used for the first data capture. RDACCESSTIME must be programmed to the rounded greater GPMC_FCLK cycle value of the read access time of the attached memory device.

In synchronous read mode, for single or burst accesses, RDACCESSTIME defines the number of GPMC_FCLK cycles from start access time to the GPMC_FCLK rising edge corresponding to the GPMC_CLK rising edge used for the first data capture.

GPMC_CLK which is sent to the memory device for synchronization with the GPMC controller, is internally retimed to correctly latch the returned data. RDCYCLETIME must be greater than RDACCESSTIME in order to let the GPMC latch the last return data using the internally retimed GPMC_CLK.

The external WAIT signal can be used in conjunction with RDACCESSTIME to control the effective GPMC data-capture GPMC_FCLK edge on read access in both asynchronous mode and synchronous mode. For details about wait monitoring, see Section 9.1.5.4.

9.1.5.3.8.2 Access Time on Write Access

In asynchronous write mode, the GPMC_CONFIG6_i[28:24] WRACCESSTIME timing parameter is not used to define the effective write access time. Instead, it is used as a WAIT invalid timing window, and must be set to a correct value so that the gpmc_wait pin is at a valid state two GPMC_CLK cycles before WRACCESSTIME completes. For details about wait monitoring, see Section 9.1.5.4.

In synchronous write mode , for single or burst accesses, WRACCESSTIME defines the number of GPMC_FCLK cycles from start access time to the GPMC_CLK rising edge used by the memory device for the first data capture.

The external WAIT signal can be used in conjunction with WRACCESSTIME to control the effective memory device data capture GPMC_CLK edge for a synchronous write access. For details about wait monitoring, see Section 9.1.5.4.

9.1.5.3.9 Page Burst Access Time (PAGEBURSTACCESSTIME)

PAGEBURSTACCESSTIME is programmed in the GPMC.GPMC_CONFIG5_i[27:24] bit field (I = 0 to 7).
PAGEBURSTACCESSTIME can be set from 0 to 15 GPMC_FCLK cycles with a granularity of one (GPMC.GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY set to 0), or from 0 to 30 GPMC_FCLK cycles with a granularity of two (TIMEPARAGRANULARITY set to 1).

9.1.5.3.9.1 Page Burst Access Time on Read Access

In asynchronous page read mode, the delay between successive word captures in a page is controlled through the PAGEBURSTACCESSTIME bit field. The PAGEBURSTACCESSTIME parameter must be programmed to the rounded greater GPMC_FCLK cycle value of the read access time of the attached device.

In synchronous burst read mode, the delay between successive word captures in a burst is controlled through the PAGEBURSTACCESSTIME field.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective GPMC data capture GPMC_FCLK edge on read access. For details about wait monitoring, see Section 9.1.5.4.

9.1.5.3.9.2 Page Burst Access Time on Write Access

Asynchronous page write mode is not supported. PAGEBURSTACCESSTIME is irrelevant in this case.

In synchronous burst write mode, PAGEBURSTACCESSTIME controls the delay between successive memory device word captures in a burst.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective memory-device data capture GPMC_CLK edge in synchronous write mode. For details about wait monitoring, see Section 9.1.5.4.

9.1.5.3.10 Bus Keeping Support

At the end-cycle time of a read access, if no other access is pending, the GPMC drives the bus with the last data read after RDCYCLETIME completion time to prevent bus floating and reduce power consumption.

After a write access, if no other access is pending, the GPMC keeps driving the data bus after WRCYCLETIME completes with the same data to prevent bus floating and power consumption.

9.1.5.4 WAIT Pin Monitoring Control

GPMC access time can be dynamically controlled using an external gpmc_wait pin when the external device access time is not deterministic and cannot be defined and controlled only using the GPMC internal RDACCESSTIME, WRACCESSTIME and PAGEBURSTACCESSTIME wait state generator.

The GPMC four input wait pins: gpmc_wait3, gpmc_wait2, gpmc_wait1, and gpmc_wait0. These four pins allow direct plugin and control of external devices with different wait-pin polarity. They also allow the overlap of wait-pin assertion from different devices without affecting access to devices for which the wait pin is not asserted.

- The GPMC.GPMC_CONFIG1_i[17:16] WAITPINSELECT field (I = 0 to 7) selects which input gpmc_wait pin is used for the device attached to the corresponding chip-select.
- The polarity of the wait pin is defined through the WAITxPINPOLARITY bit of the GPMC.GPMC_CONFIG register. A wait pin configured to be active low means that low level on the WAIT signal indicates that the data is not ready and that the data bus is invalid. When WAIT is inactive, data is valid.

The GPMC access engine can be configured by CS to monitor the wait pin of the external memory device or not, based on the access type: read or write.

- The GPMC.GPMC_CONFIG1_i[22] WAITREADMONITORING bit defines whether the wait pin should be monitored during read accesses or not.
- The GPMC.GPMC_CONFIG1_i[21] WAITWRITEMONITORING bit defines whether the wait pin should be monitored during write accesses or not.

The GPMC access engine can be configured to monitor the wait pin of the external memory device asynchronously or synchronously with the GPMC_CLK clock, depending on the access type: synchronous or asynchronous (the GPMC.GPMC_CONFIG1_i[29] READTYPE and GPMC.GPMC_CONFIG1_i[27] WRITETYPE bits).

9.1.5.4.1 Wait Monitoring During an Asynchronous Read Access

When wait-pin monitoring is enabled for read accesses (WAITREADMONITORING), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state.

During asynchronous read accesses with wait-pin monitoring enabled, the wait pin must be at a valid level (asserted or deasserted) for at least two GPMC clock cycles before RDACCESSTIME completes, to ensure correct dynamic access-time control through wait-pin monitoring. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

In this context, RDACCESSTIME is used as a WAIT invalid timing window and is set to such a value that the wait pin is at a valid state two GPMC clock cycles before RDACCESSTIME completes.

Similarly, during a multiple-access cycle (for example, asynchronous read page mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-deasserted state. Wait-monitoring pipelining is also applicable to multiple accesses (access within a page).

- WAIT monitored as active freezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as asserted extends the current access time in the page. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as inactive completes the current access time and starts the next access phase in the page. The data bus is considered valid, and data are captured during this clock cycle. In case of a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their related control timing value and according to the CYCLETIME counter status.

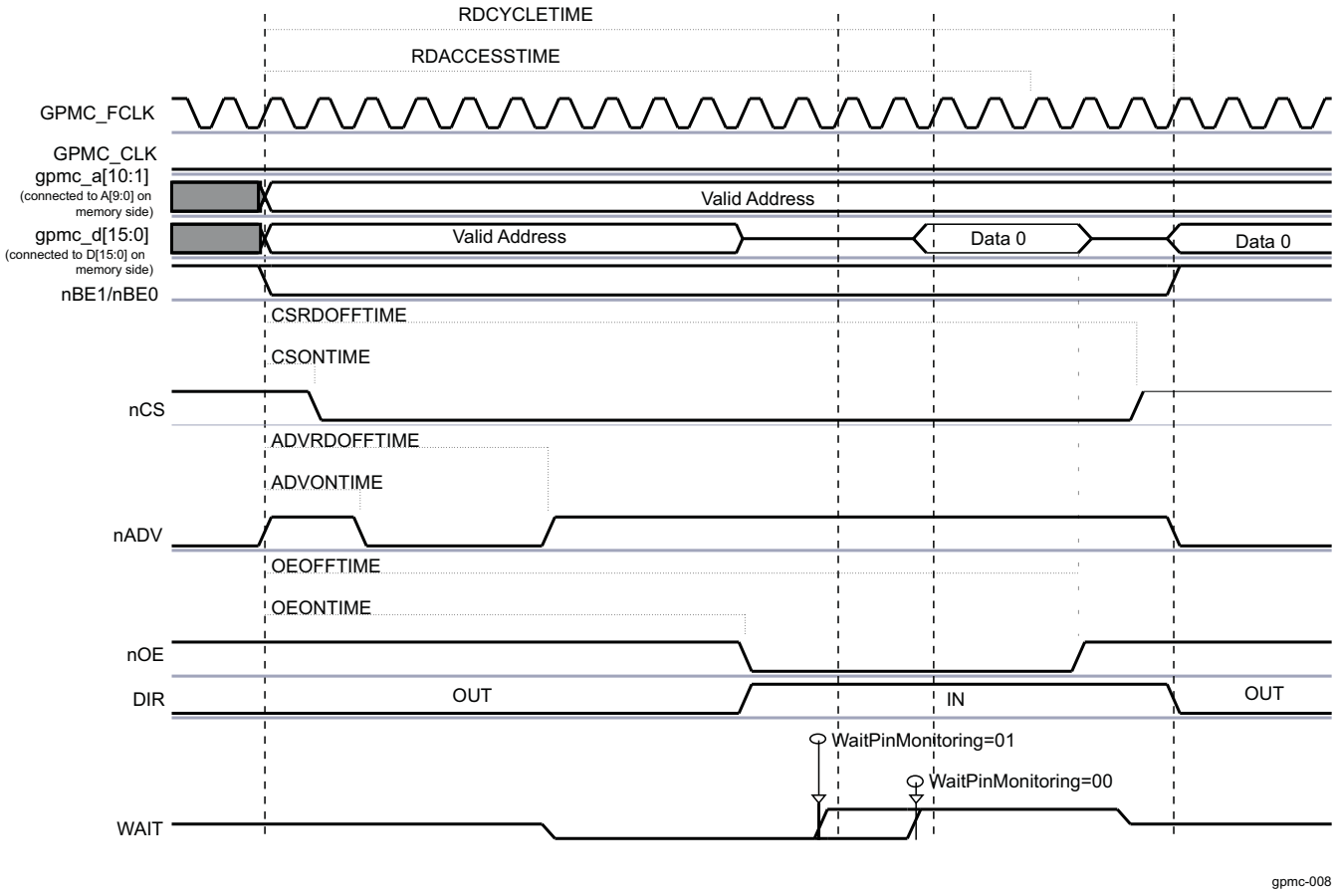
When a delay larger than two GPMC clocks must be observed between wait-pin deactivation time and data valid time (including the required GPMC and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data-capture time and the effective unlock of the CYCLETIME counter. This extra delay can be programmed in the GPMC.GPMC_CONFIG1_i[19:18] WAITMONITORINGTIME field (I = 0 to 7).

NOTE:

- The WAITMONITORINGTIME parameter does not delay the wait-pin active or inactive detection, nor does it modify the two GPMC clocks pipelined detection delay.
- This extra delay is expressed as a number of GPMC_CLK clock cycles, even though the access is defined as asynchronous, and no GPMC_CLK clock is provided to the external device. Still, GPMCFCLKDIVIDER is used as a divider for the GPMC clock, so it must be programmed to define the correct WAITMONITORINGTIME delay.

Figure 9-8 shows wait behavior during an asynchronous single read access.

Figure 9-8. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDivider = 1)



NOTE: The WAIT signal is active low. WAITMONITORINGTIME = 00, 01.

9.1.5.4.2 Wait Monitoring During an Asynchronous Write Access

When wait-pin monitoring is enabled for write accesses (GPMC.GPMC_CONFIG1_i[21] WAITWRITEMONITORING bit = 0x1), the WAIT-invalid timing window is defined by the WRACCESSTIME field. WRACCESSTIME must be set so that the wait pin is at a valid state two GPMC clock cycles before WRACCESSTIME completes. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

- WAIT monitored as active freezes the CYCLETIME counter. This informs the GPMC that the data bus is not captured by the external device. The control signals are kept in their current state. The data bus still drives the data.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. This informs that the data bus is correctly captured by the external device. All signals, including the data bus, are controlled according to their related control timing value and to the CYCLETIME counter status.

When a delay larger than two GPMC clock cycles must be observed between wait-pin deassertion time and the effective data write into the external device (including the required GPMC data setup time and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data write time into the external device and the effective unfreezing of the CYCLETIME counter. This extra delay can be programmed in the GPMC.GPMC_CONFIG1_i[19:18] WAITMONITORINGTIME fields (I = 0 to 7).

NOTE:

- The WAITMONITORINGTIME parameter does not delay the wait-pin assertion or deassertion detection, nor does it modify the two GPMC clock cycles pipelined detection delay.
- This extra delay is expressed as a number of GPMC_CLK clock cycles, even though the access is defined as synchronous, and even though no clock is provided to the external device. Still, GPMC_CONFIG1_i[1:0] GPMCFCLKDIVIDER is used as a divider for the GPMC clock and so it must be programmed to define the correct WAITMONITORINGTIME delay.

9.1.5.4.3 Wait Monitoring During a Synchronous Read Access

During synchronous accesses with wait-pin monitoring enabled, the wait pin is captured synchronously with GPMC_CLK, using the rising edge of this clock.

The WAIT signal can be programmed to apply to the same clock cycle it is captured in. Alternatively, it can be sampled one or two GPMC_CLK cycles ahead of the clock cycle it applies to. This pipelining is applicable to the entire burst access, and to all data phase in the burst access. This WAIT pipelining depth is programmed in the GPMC.GPMC_CONFIG1_i[19:18] WAITMONITORINGTIME field (where I = 0 to 7), and is expressed as a number of GPMC_CLK clock cycles.

In synchronous mode, when wait-pin monitoring is enabled (GPMC.GPMC_CONFIG1_i[22] WAITREADMONITORING bit), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the WAIT deasserted-state detection.

Depending on the programmed WAITMONITORINGTIME value, the wait pin should be at a valid level, either asserted or deasserted:

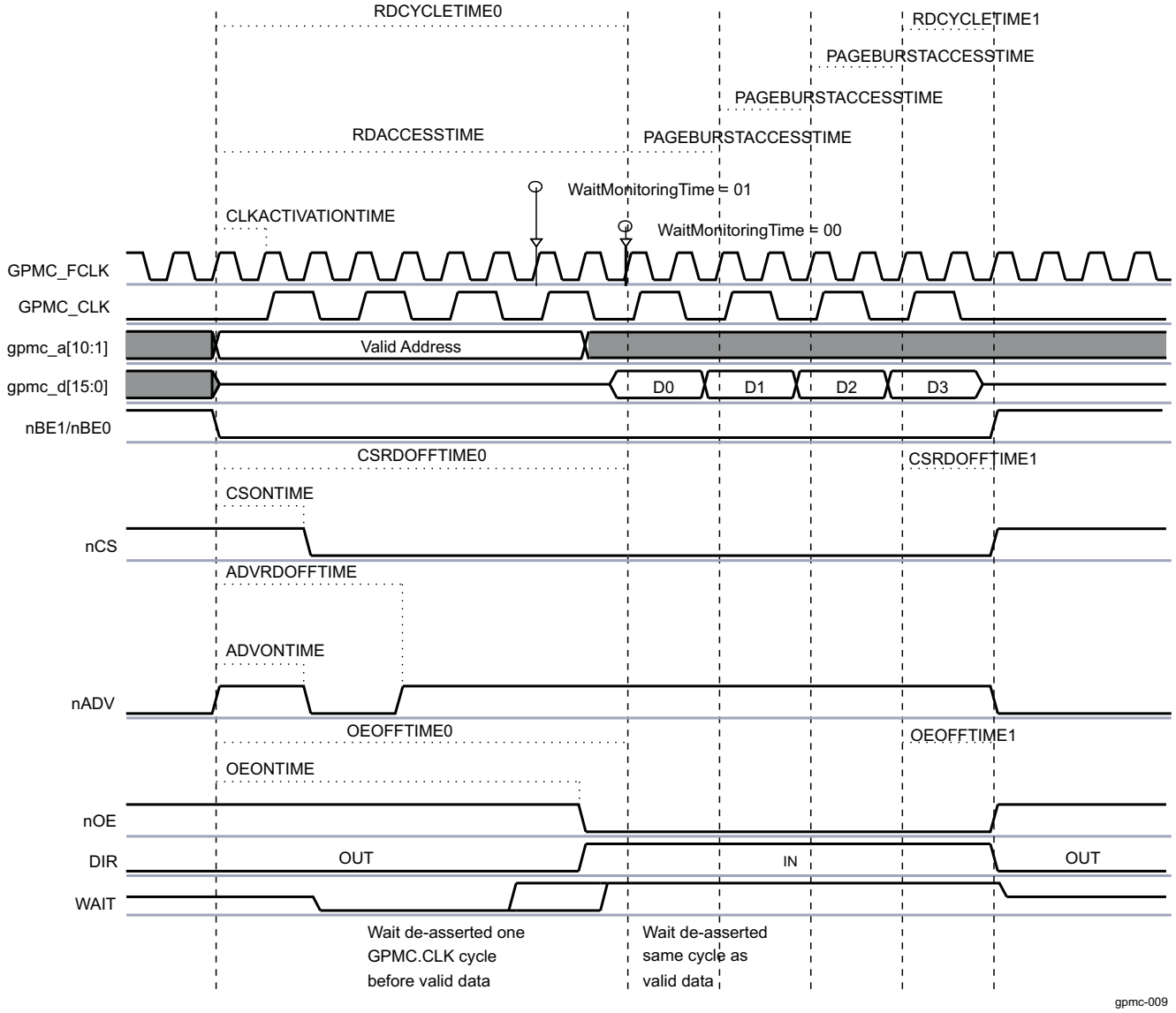
- In the same clock cycle the data is valid if WAITMONITORINGTIME = 0 (at RDACCESSTIME completion)
- In the WAITMONITORINGTIME × (GPMCFCLKDIVIDER + 1) GPMC_FCLK clock cycles before RDACCESSTIME completion if WAITMONITORINGTIME ≠ 0

Similarly, during a multiple-access cycle (burst mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-inactive state. The Wait pipelining depth programming applies to the whole burst access.

- WAIT monitored as active freezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in a lock state), WAIT monitored as active extends the current access time in the burst. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in lock state), WAIT monitored as inactive completes the current access time and starts the next access phase in the burst. The data bus is considered valid, and data are captured during this clock cycle. In a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their relative control timing value and the CYCLETIME counter status.

Figure 9-9 shows wait behavior during a synchronous read burst access.

Figure 9-9. Wait Behavior During a Synchronous Read Burst Access



NOTE: The WAIT signal is active low. WAITMONITORINGTIME = 00, 01.

9.1.5.4.4 Wait Monitoring During a Synchronous Write Access

During synchronous accesses with wait-pin monitoring enabled (the WAITWRITEMONITORING bit), the wait pin is captured synchronously with GPMC_CLK, using the rising edge of this clock.

If enabled, external wait-pin monitoring can be used in combination with WRACCESSTIME to control the effective memory device GPMC_CLK capture edge.

Wait-monitoring pipelining depth is similar to synchronous read access:

- At WRACCESSTIME completion if WAITMONITORINGTIME = 0
- The WAITMONITORINGTIME × (GPMCFCLKDIVIDER + 1) GPMC_FCLK cycles before WRACCESSTIME completion if WAITMONITORINGTIME ≠ 0.

Wait-monitoring pipelining definition applies to whole burst accesses:

- WAIT monitored as active freezes the CYCLETIME counter. For accesses within a burst, when the

CYCLETIME counter is by definition in a lock state, WAIT monitored as active indicates that the data bus is not being captured by the external device. Control signals are kept in their current state. The data bus is kept in its current state.

- WAIT monitored as inactive unfreezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as inactive indicates the effective data capture of the bus by the external device and starts the next access of the burst. In case of a single access or if this was the last access in a multiple access cycle, all signals, including the data bus, are controlled according to their related control timing value and the CYCLETIME counter status.

NOTE: Wait monitoring is supported for all configurations except for GPMC_CONFIG1_i[19:18] WAITMONITORINGTIME = 0x 0 (where I = 0 to 7) for write bursts with a clock divider of 1 or 2 (GPMC_CONFIG1_i[1:0] GPMCFCLKDIVIDER field equal to 0x0 or 0x1 respectively).

9.1.5.4.5 WAIT with NAND Device

For details about the use of the wait pin for communication with a NAND flash external device, see Section 9.1.5.14.2, *NAND Device-Ready Pin*.

9.1.5.4.6 Idle Cycle Control between Successive Accesses

9.1.5.4.6.1 Bus Turnaround (BUSTURNAROUND)

To prevent data-bus contention, an access that follows a read access to a slow memory/device (that is, control the nCS/nOE de-assertion to data bus in high-impedance delay) must be delayed.

The bus turnaround is a time-out counter starting after nCS or nOE de-assertion time (whichever occurs first) and delays the next access start-cycle time. It is programmed through the GPMC.GPMC_CONFIG6_i[3:0] BUSTURNAROUND bit field (where I = 0 to 7).

After a read access to a chip-select with a non zero BUSTURNAROUND, the next access is delayed until the BUSTURNAROUND delay completes, if the next access is one of the following:

- A write access to any chip-select (same or different from the chip-select data was read from)
- A read access to a different chip-select from the chip-select data was read access from
- A read or write access to a chip-select associated with an address/data-multiplexed device

Another way to prevent bus contention is to define an earlier nCS or nOE deassertion time for slow devices or to extend the value of RDCYCLETIME. Doing this prevents bus contention, but affects all accesses of this specific chip-select.

9.1.5.4.6.2 Idle Cycles Between Accesses to Same Chip-Select (CYCLE2CYCLESAMECSEN, CYCLE2CYCLEDELAY)

Some devices require a minimum chip-select signal inactive time between accesses. The GPMC.GPMC_CONFIG6_i[7] CYCLE2CYCLESAMECSEN bit (I = 0 to 7) enables insertion of a minimum number of GPMC_FCLK cycles, defined by the GPMC.GPMC_CONFIG6_i[11:8] CYCLE2CYCLEDELAY field, between successive accesses of any type (read or write) to the same chip-select.

If CYCLE2CYCLESAMECSEN is enabled, any subsequent access to the same chip-select is delayed until its CYCLE2CYCLEDELAY completes. The CYCLE2CYCLEDELAY counter starts when CSRDOFFTIME/CSWROFFTIME completes.

The same applies to successive accesses occurring during Word32 or burst accesses split into successive single accesses when the single-access mode is used (GPMC_CONFIG1_i[30] READMULTIPLE = 0 or GPMC_CONFIG1_i[28] WRITEMULTIPLE = 0).

All control signals are kept in their default states during these idle GPMC_FCLK cycles. This prevents back-to-back accesses to the same chip-select without idle cycles between accesses.

9.1.5.4.6.3 Idle Cycles Between Accesses to Different Chip-Select (CYCLE2CYCLEDIFFCSEN,

CYCLE2CYCLEDELAY)

Because of the pipelined behavior of the system, successive accesses to different chip-selects can occur back-to-back with no idle cycles between accesses. Depending on the control signals (nCS, nADV/ALE, nBE0/CLE, nOE/RE, nWE) assertion and de-assertion timing parameters and on the IC timing parameters, some control signals assertion times may overlap between the successive accesses to different CS. Similarly, some control signals (WE, OE/RE) may not respect required transition times.

To work around the overlapping and to observe the required control-signal transitions, a minimum of CYCLE2CYCLEDELAY inactive cycles is inserted between the access being initiated to this chip-select and the previous access ending for a different chip-select. This applies to any type of access (read or write).

If GPMC_CONFIG6_i[6] CYCLE2CYCLEDIFFCSEN is enabled, the chip-select access is delayed until CYCLE2CYCLEDELAY cycles have expired since the end of a previous access to a different chip-select. CYCLE2CYCLEDELAY count starts at CSRDOFFTIME/CSWROFFTIME completion. All control signals are kept inactive during the idle GPMC_FCLK cycles.

NOTE: CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDIFFCSEN should be set in GPMC_CONFIG6_i registers to respectively get idle cycles inserted between accesses on this chip-select and after accesses to a different chip-select.

The CYCLE2CYCLEDELAY delay runs in parallel with the BUSTURNAROUND delay. It should be noted that BUSTURNAROUND is a timing parameter defined for the ending chip-select access, whereas CYCLE2CYCLEDELAY is a timing parameter defined for the starting chip-select access. The effective minimum delay between successive accesses is based on the larger delay timing parameter and on access type combination, since bus turnaround does not apply to all access types. See Section 9.1.5.4.6.1 for more details on bus turnaround.

Table 9-3 describes the configuration required for idle cycle insertion.

Table 9-3. Idle Cycle Insertion Configuration

1st Access Type	BUSTURN AROUND Timing Parameter	Second Access Type	Chip-Select	Add/Data Multiplexed	CYCLE2 CYCLE SAMECSEN Parameter	CYCLE2 CYCLE DIFFCSEN Parameter	Idle Cycle Insertion Between the Two Accesses
R/W	= 0	R/W	Any	Any	0	x	No idle cycles are inserted if the two accesses are well pipelined.
R	> 0	R	Same	Nonmuxed	x	0	No idle cycles are inserted if the two accesses are well pipelined.
R	> 0	R	Different	Nonmuxed	0	0	BTA cycles are inserted.
R	> 0	R/W	Any	Muxed	0	0	BTA cycles are inserted.
R	> 0	W	Any	Any	0	0	BTA cycles are inserted.
W	> 0	R/W	Any	Any	0	0	No idle cycles are inserted if the two accesses are well pipelined.
R/W	= 0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted.
R/W	= 0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted.
R/W	> 0	R/W	Same	Any	1	x	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is max (BUSTURNAROUND, CYCLE2CYCLEDELAY).

Table 9-3. Idle Cycle Insertion Configuration (continued)

1st Access Type	BUSTURN AROUND Timing Parameter	Second Access Type	Chip-Select	Add/Data Multiplexed	CYCLE2 CYCLE SAMECSEN Parameter	CYCLE2 CYCLE DIFFCSEN Parameter	Idle Cycle Insertion Between the Two Accesses
R/W	> 0	R/W	Different	Any	x	1	CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is maximum (BUSTURNAROUND, CYCLE2CYCLEDELAY).

9.1.5.4.7 Slow Device Support (TIMEPARAGRANULARITY Parameter)

All access-timing parameters can be multiplied by 2 by setting the GPMC.GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY bit (where I stands for the GPMC chip-select value, I = 0 to 7). Increasing all access timing parameters allows support of slow devices.

9.1.5.5 gpmc_io_dir Pin

The gpmc_io_dir pin is used to control I/O direction on the GPMC data bus gpmc_d[15:0]. Depending on top-level pad multiplexing, this signal can be output and used externally to the processor, if required.

The gpmc_io_dir pin is low during transmit (OUT) and high during receive (IN).

For write accesses, the gpmc_io_dir pin stays OUT from start-cycle time to end-cycle time.

For read accesses, the gpmc_io_dir pin goes from OUT to IN at nOE assertion time and stays IN until:

- BUSTURNAROUND is enabled:
  - The gpmc_io_dir pin goes from IN to OUT at end-cycle time plus programmable bus turnaround time.
- BUSTURNAROUND is disabled:
  - After an asynchronous read access, the gpmc_io_dir pin goes from IN to OUT at RDACCESSTIME + 1 GPMC_FCLK cycle or when RDCYCLETIME completes, whichever occurs last.
  - After a synchronous read access, the gpmc_io_dir pin goes from IN to OUT at RDACCESSTIME + 2 GPMC_FCLK cycles or when RDCYCLETIME completes, whichever occurs last.

Because of the bus-keeping feature of the GPMC, after a read or write access and with no other accesses pending, the default value of the gpmc_io_dir pin is OUT (see Section 9.1.5.3.10, *Bus Keeping Support*).

To prevent unnecessary toggling, the gpmc_io_dir pin stays IN between two successive read accesses to a nonmultiplexed device (address mapping supports nonmultiplexed 16-bit wide devices with limited address (2 Kbytes)).

Figure 9-10 shows address mapping in nonmultiplexed mode with a limited address range (A[10:1]).

9.1.5.6 Reset

No reset signal is sent to the external memory device by the GPMC. The PRCM specifications provide more information about external-device reset.

The PRCM module provides an input pin, global_rst_n, to the GPMC:

- The global_rst_n pin is activated during processor's warm reset and cold reset.
- The global_rst_n pin initializes the internal state-machine and the internal configuration registers.

9.1.5.7 Write Protect (nWP)

When connected to the attached memory device, the WRITE PROTECT signal can enable or disable the lockdown function of the attached memory.

The gpmc_nwp output pin value is controlled through the GPMC.GPMC_CONFIG[4] WRITEPROTECT bit, which is common to all CS.

9.1.5.8 Byte Enable (nBE1/nBE0)

BYTE ENABLE signals (nBE1/nBE0) are:

- Valid (asserted or nonasserted according to the incoming system request) from access start to access completion for asynchronous and synchronous single accesses
- Asserted low from access start to access completion for asynchronous and synchronous multiple read accesses
- Valid (asserted or nonasserted, according to the incoming system request) synchronously to each written data for synchronous multiple write accesses

9.1.5.9 Asynchronous Access Description

In asynchronous operations:

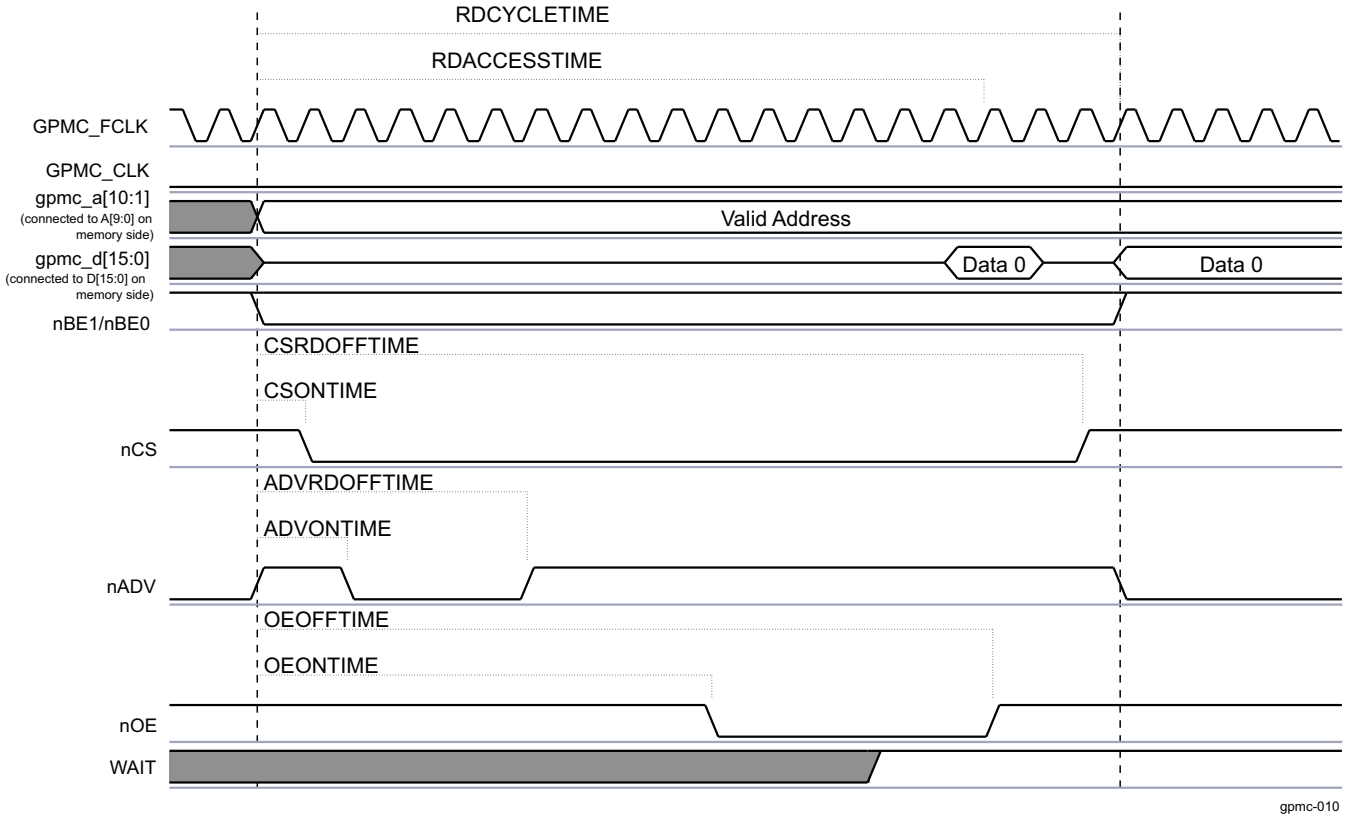
- GPMC_CLK is not provided outside the GPMC.
- GPMC_CLK is kept low.

9.1.5.9.1 Asynchronous Single Read

9.1.5.9.1.1 Asynchronous Single Read Operation on a Nonmultiplexed Device

Figure 9-10 shows an asynchronous single read operation on a nonmultiplexed device.

Figure 9-10. Asynchronous Single Read on an Address/Data-Nonmultiplexed Device



In the following section I stands for the chip-select number, I = 0 to 7.

- GPMC.GPMC_CONFIG[1] LIMITEDADDRESS set to 1 (A26-A11 are not modified during an external memory access)
- GPMC.GPMC_CONFIG1_i register settings:
  - GPMC_CONFIG1_i[30] READMULTIPLE bit at 0 (read single access)
  - GPMC_CONFIG1_i[29] READTYPE bit at 0 (asynchronous read)
  - GPMC_CONFIG1_i[9] MUXADDDATA bit at 0 (non multiplexed device)
- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC_CONFIG2_i[3:0] CSONTIME field. It controls the address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC_CONFIG2_i[12:8] CSRDOFFTIME field. It controls the address hold time from nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC_CONFIG3_i[3:0] ADVONTIME field.

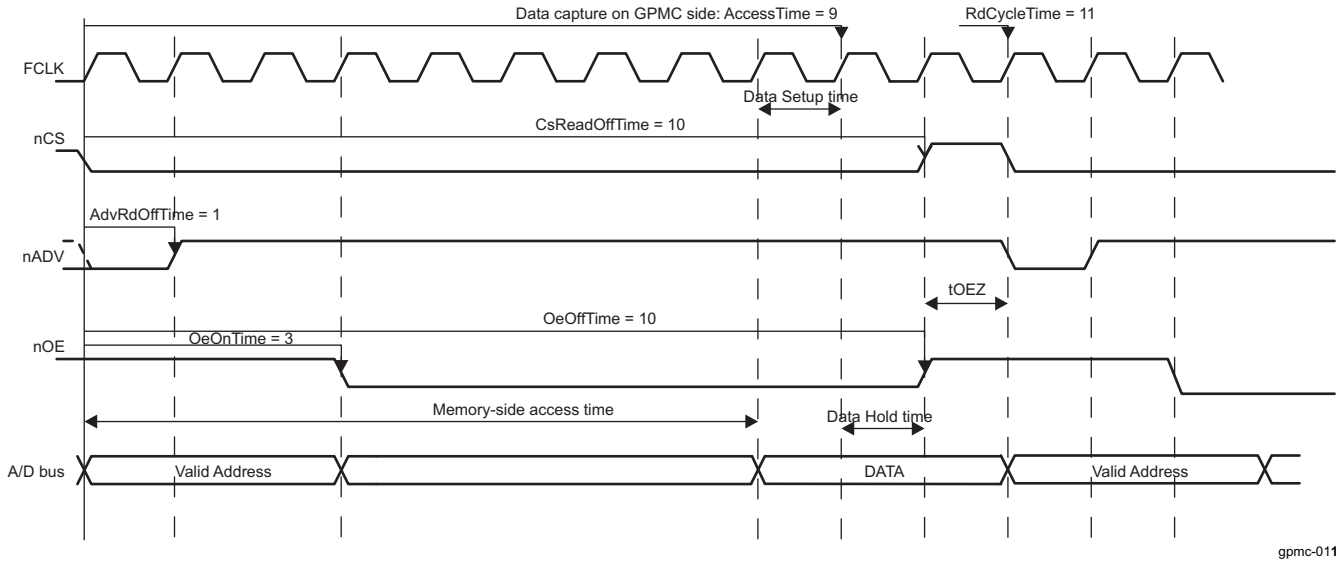
  - nADV deassertion time is controlled by the GPMC_CONFIG3_i[12:8] ADVRDOFFTIME field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC_CONFIG4_i[3:0] OEONTIME field.
  - nOE deassertion time is controlled by the GPMC_CONFIG4_i[12:8] OEOFFTIME field.
- Read data is latched when RDACCESSTIME completes. Access time is defined in the GPMC.GPMC_CONFIG5_i[20:16] RDACCESSTIME field.
- The end of the access is defined by the RDCYCLETIME parameter. The read cycle time is defined in the GPMC.GPMC_CONFIG5_i[4:0] RDCYCLETIME field.
- Direction signal DIR: DIR goes from OUT to IN at the same time that nOE is asserted.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 9.1.5.3.10, *Bus Keeping Support* for more details.

9.1.5.9.1.2 Asynchronous Single-Read Operation on an Address/Data Multiplexed Device

Figure 9-11 shows an asynchronous single read operation on an address/data-multiplexed device.

Figure 9-11. Asynchronous Single Read on an Address/Data-Multiplexed Device



When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until nOE assertion time. For details, see Section 9.1.5.2.3, *Address/Data-Multiplexing Interface*.

GPMC.GPMC_CONFIG1_i register settings (I = 0 to 7):

- READMULTIPLE bit at 0 (read single access)
- READTYPE bit at 0 (read asynchronous)
- MUXADDDATA bit at 1 (address/data-multiplexed device)

Address bits ([16:1] from a GPMC perspective, [15:0] from an external device perspective) are placed on the address/data bus, and the remaining address bits [25:16] are placed on the address bus. The address phase ends at nOE assertion, when the DIR signal goes from OUT to IN.

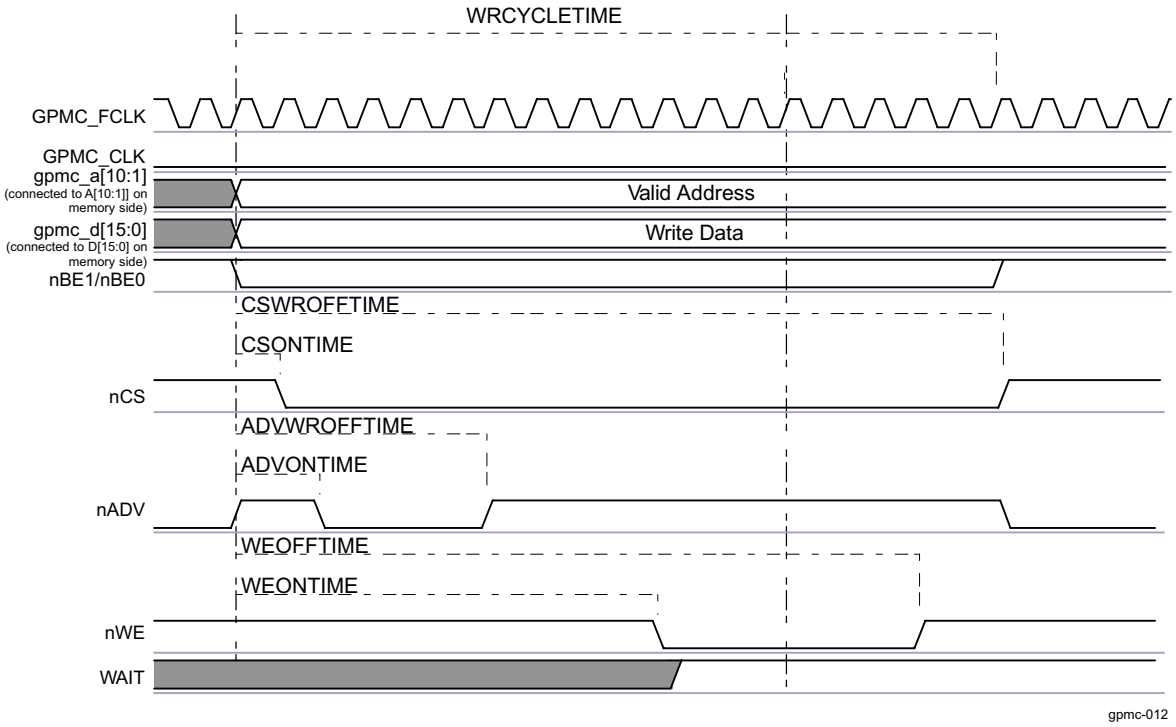
The nCS, nADV, nOE, and DIR signals are controlled in the same way as nonmultiplexed accesses.

9.1.5.9.2 Asynchronous Single Write

9.1.5.9.2.1 Asynchronous Single Write Operation on a Nonmultiplexed Device

Figure 9-12 shows an asynchronous single write operation on a nonmultiplexed device.

Figure 9-12. Asynchronous Single Write on an Address/Data-Nonmultiplexed Device



In the following section I stands for the chip-select number, I = 0 to 7.

- GPMC.GPMC_CONFIG[1] LIMITEDADDRESS set to 1 (A26-A11 are not modified during an external memory access)
- GPMC.GPMC_CONFIG1_i register settings:
  - WRITEMULTIPLE bit at 0 (write single access)
  - WRITETYPE bit at 0 (write asynchronous)
  - MUXADDDATA bit at 0 (nonaddress/data-multiplexed device)
- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC.GPMC_CONFIG2_i[3:0] CSONTIME field and ensures address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC.GPMC_CONFIG2_i[20:16] CSWROFFTIME field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC.GPMC_CONFIG3_i[3:0] ADVONTIME field.
  - nADV deassertion time is controlled by the GPMC.GPMC_CONFIG3_i[20:16] ADVWROFFTIME field.

Address and data are driven on their corresponding buses at start-of-cycle time.

- Write enable signal nWE:
  - nWE assertion indicates a write cycle.
  - nWE assertion time is controlled by the GPMC.GPMC_CONFIG4_i[19:16] WEONTIME field.
  - nWE deassertion time is controlled by the GPMC.GPMC_CONFIG4_i[28:24] WEOFFTIME field.
- Direction signal DIR:

DIR signal is OUT during the entire access.

- The end of the access is defined by the WRCYCLETIME parameter.

  This write-cycle time is defined in the GPMC.GPMC_CONFIG5_i[12:8] WRCYCLETIME field.

After a write operation, if no other access (read or write) is pending, the data bus keeps its previous value. See Section 9.1.5.3.10, *Bus Keeping Support*.

In the GPMC, when a 16-bit wide device is attached to the controller, a Word32 write access is split into two Word16 write accesses. For more information about GPMC access size and type adaptation, see Section 9.1.5.2.7,*System Burst Versus External Device Burst Support*.

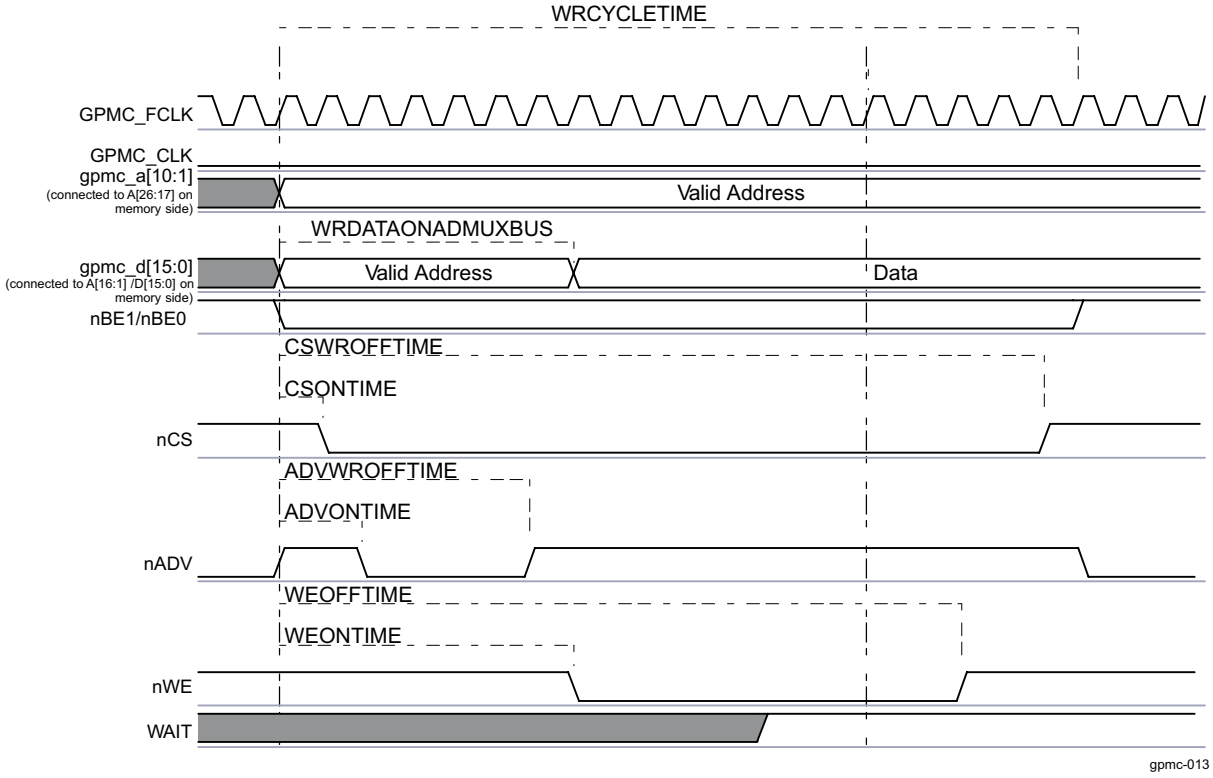
Between two successive accesses, if an nCS pulse is needed:

- The GPMC.GPMC_CONFIG6_i[11:8] CYCLE2CYCLEDELAY field can be programmed with GPMC.GPMC_CONFIG6_i[7] CYCLE2CYCLESAMECSEN enabled.
- The CSWROFFTIME and CSONTIME parameters also allow a chip-select pulse, but this affects all other types of access.

9.1.5.9.2.2 Asynchronous Single Write Operation on an Address/Data-Multiplexed Device

Figure 9-13 shows an asynchronous single write operation on an address/data-multiplexed device.

Figure 9-13. Asynchronous Single Write on an Address/Data-Multiplexed Device



When the GPMC generates a write access to an address/data-multiplexed device, it drives the address on the address/data muxed bus until WRDATAONADMUXBUS time, and then it drives the data. For more information, see Section 9.1.5.2.3, *Address/Data-Multiplexing Interface*.

GPMC.GPMC_CONFIG1_i register settings (I = 0 to 7):

- WRITEMULTIPLE bit at 0 (write single access)
- WRITETYPE bit at 0 (write asynchronous)
- MUXADDDATA bit at 1 (address/data-multiplexed device)

Address bits [16:1] are placed on the address/data bus at the start of cycle time, and the remaining address bits [26:17] are placed on the address bus.

The nCS, nADV, and nWE signals are controlled in the same way as nonmultiplexed accesses.

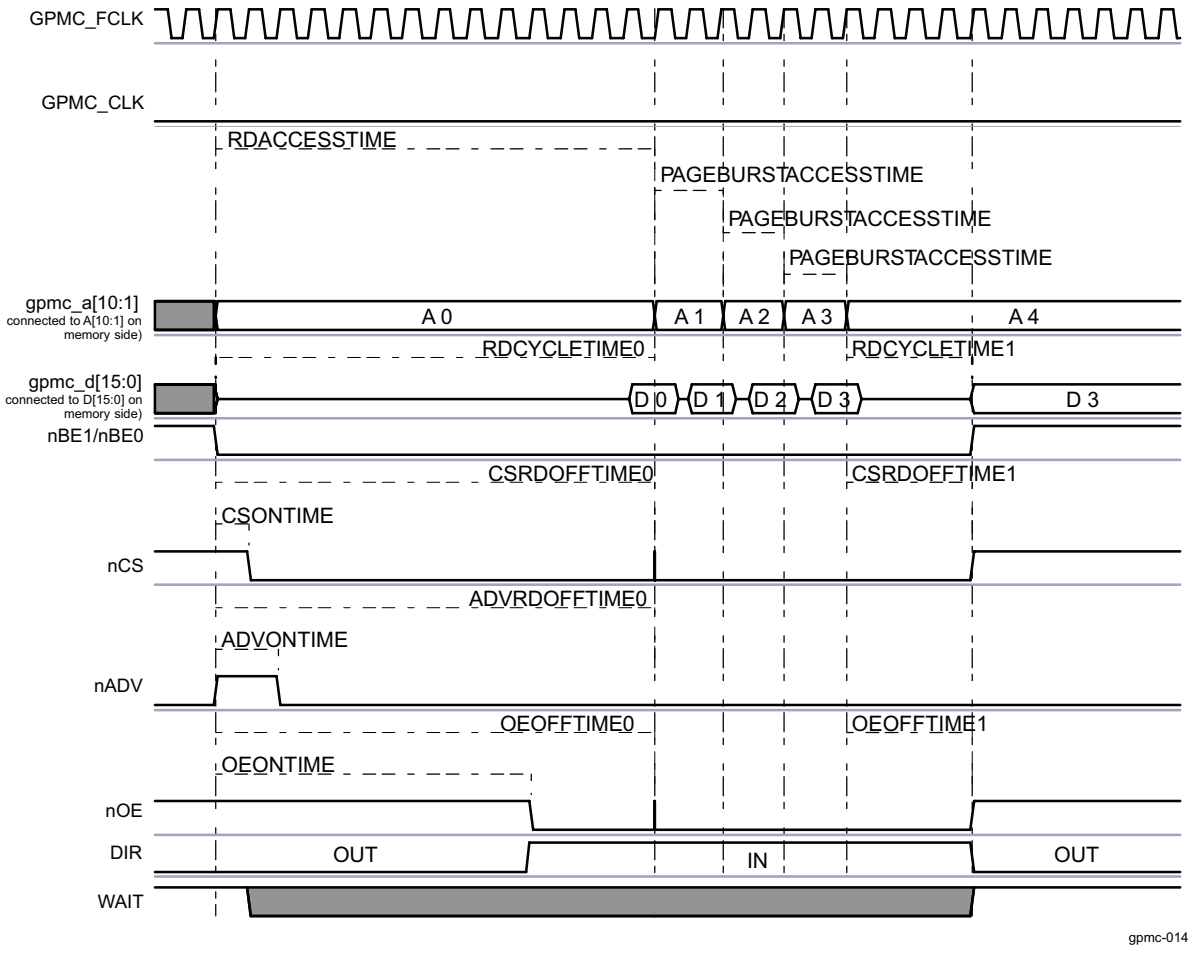
Data is driven on the address/data bus at a GPMC_CONFIG6_i[19:16] WRDATAONADMUXBUS time.

NOTE: Write multiple access in asynchronous mode is not supported. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

9.1.5.9.3 Asynchronous Multiple (Page Mode) Read

Figure 9-14 shows an asynchronous multiple read operation.

Figure 9-14. Asynchronous Multiple (Page Mode) Read



NOTE: The WAIT signal is active low.

In the following section I stands for the chip-select number, I = 0 to 7.

For read access with GPMC.GPMC_CONFIG1_i register settings:

- READMULTIPLE bit at 1 (read multiple access)
- READTYPE bit at 0 (read asynchronous)
- MUXADDDATA bit at 0 (non-address/data-multiplexed device). The page mode is not supported by address/data-multiplexed devices.

In Figure 9-14, two Word32 read host accesses on the GPMC configured with READMULTIPLE = 1, READTYPE = 0, and MUXADDDATA = 0 are merged into one multiple-read access (page mode of four Word16) on the attached device.

When RDACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC.GPMC_CONFIG2_i[3:0] CSONTIME field and ensures the address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC.GPMC_CONFIG2_i[12:8] CSRDOFFTIME field and ensures the address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC.GPMC_CONFIG3_i[3:0] ADVONTIME field.
  - nADV deassertion time is controlled by the GPMC.GPMC_CONFIG3_i[12:8] ADVRDOFFTIME field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC.GPMC_CONFIG4_i[3:0] OEONTIME field.
  - nOE deassertion time is controlled by the GPMC.GPMC_CONFIG4_i[12:8] OEOFFTIME field.
- Initial latency for the first read data is controlled by the RDACCESSTIME parameter.

  The access time is defined in the GPMC.GPMC_CONFIG5_i[20:16] RDACCESSTIME field.

During consecutive accesses, the GPMC increments the address after each data read completes.

- Delay between successive read data in the page is controlled by the PAGEBURSTACCESSTIME parameter:
  - This timing is defined in the GPMC.GPMC_CONFIG5_i[27:24] PAGEBURSTACCESSTIME field.
  - Depending on the device page length, the GPMC can control device page crossing during a burst request and insert initial RDACCESSTIME latency. Note that page crossing is only possible with a new burst access, meaning a new initial access phase is initiated.
- Total access time (RDCYCLETIME) corresponds to RDACCESSTIME plus the address hold time starting from the nCS deassertion plus the time from RDACCESSTIME to CSRDOFFTIME.
  - The read cycle time is defined in the GPMC.GPMC_CONFIG5_i[4:0] RDCYCLETIME field.
  - In Figure 9-14, the RDCYCLETIME programmed value equals RDCYCLETIME0 (before paged accesses) + RDCYCLETIME1 (after paged accesses).
- Direction signal DIR:

  DIR goes from OUT to IN at the same time as nOE assertion time.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 9.1.5.3.10, *Bus Keeping Support*.

9.1.5.10 Synchronous Access

In synchronous operations:

- The GPMC_CLK clock is provided outside the GPMC when accessing the memory device.
- The GPMC_CLK clock is derived from the GPMC_FCLK clock using the GPMC.GPMC_CONFIG1_i[1:0] GPMCFCLKDIVIDER field (where I = 0 to 7).
- The GPMC.GPMC_CONFIG1_i[26:25] CLKACTIVATIONTIME field specifies that the GPMC_CLK is provided outside the GPMC 0, 1, or 2 GPMC_FCLK cycles after start access time until CycleTime completes.
- When the GPMC is configured for synchronous mode, the GPMC_CLK signal (which is an output) must also be set as an input (CONTROL.CONTROL_PADCONF_GPMC_NCS7[24] INPUTENABLE1 = 1). GPMC_CLK is looped back through the output and input buffers of the corresponding GPMC_CLK pad at the device boundary. The looped-back clock is used to synchronize the sampling of the memory signals.

9.1.5.10.1 Synchronous Single Read

Figure 9-15 and Figure 9-16 show a synchronous single-read operation with GPMCFCLKDIVIDER equal to 0 and 1, respectively.

Figure 9-15. Synchronous Single Read (GPMCFCLKDIVIDER = 0)

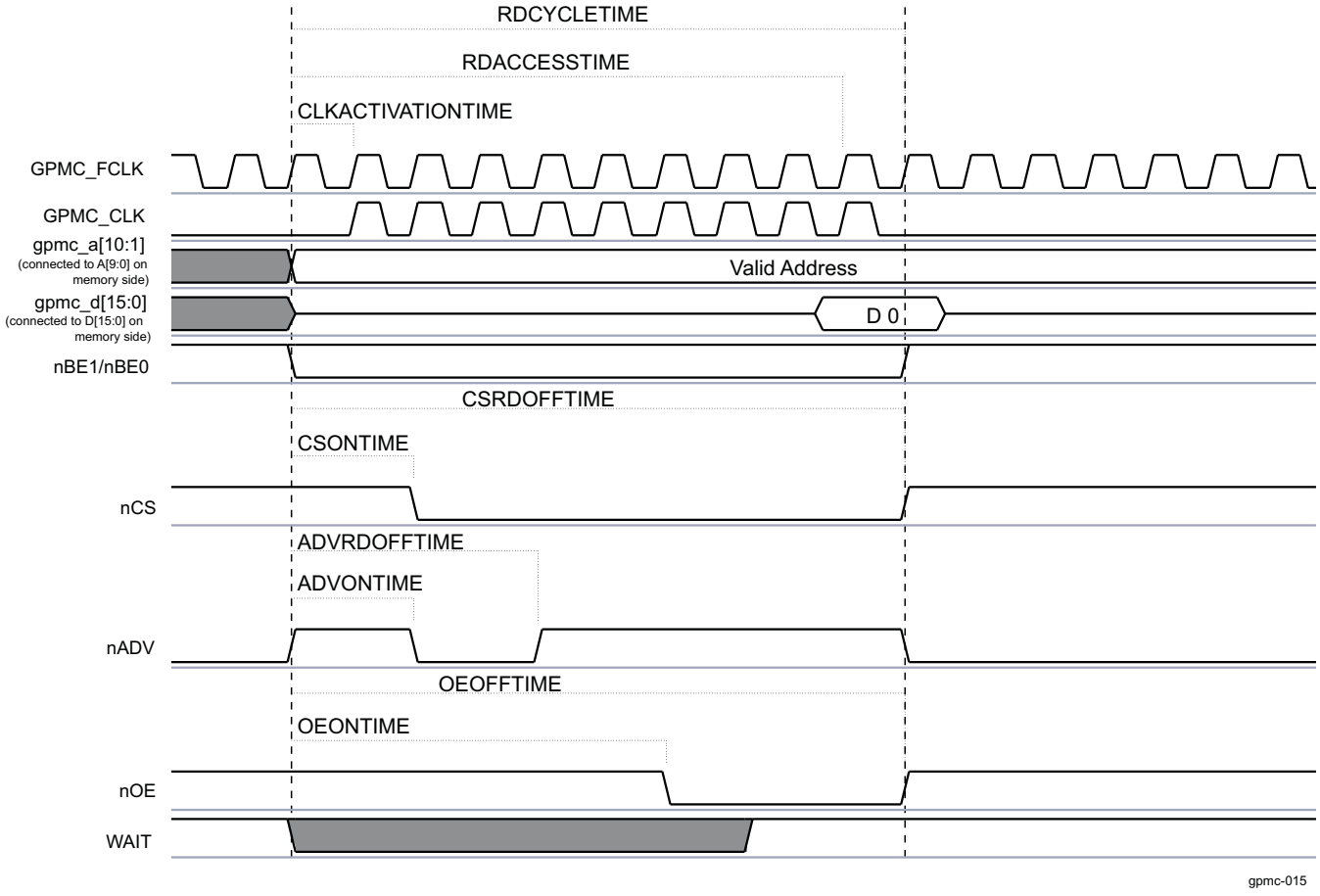
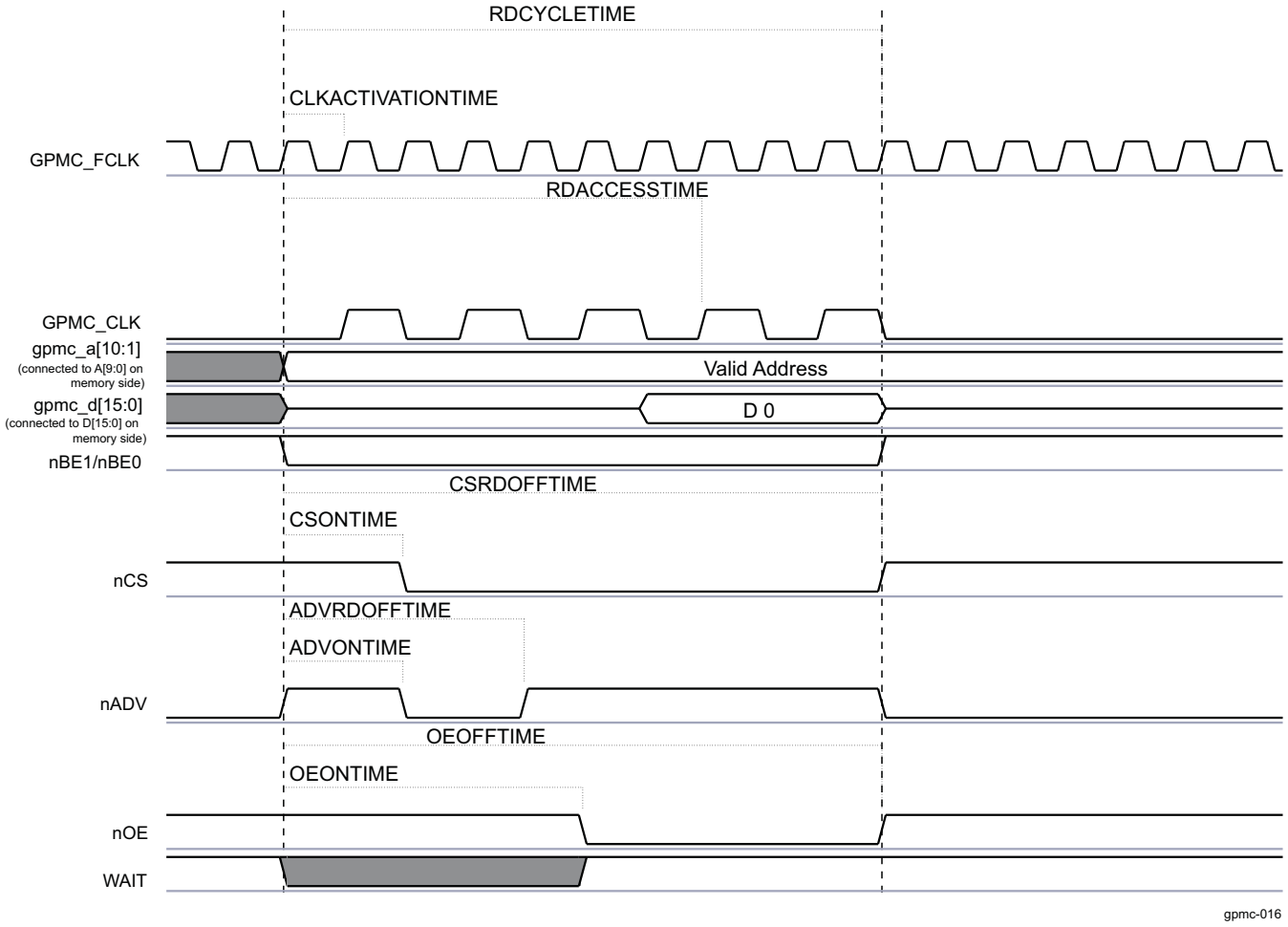


Figure 9-16. Synchronous Single Read (GPMCFCLKDIVIDER = 1)



In the following section I stands for the chip-select number, I = 0 to 7.

- GPMC.GPMC_CONFIG1_i register settings:
  - READMULTIPLE bit at 0 (read single access)
  - READTYPE bit at 1 (read synchronous)
  - MUXADDDATA bit at 0 (non-address/data-multiplexed device)
- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC.GPMC_CONFIG2_i[3:0] CSONTIME field and ensures address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC.GPMC_CONFIG2_i[12:8] CSRDOFFTIME field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC.GPMC_CONFIG3_i[3:0] ADVONTIME field.
  - nADV deassertion time is controlled by the GPMC.GPMC_CONFIG3_i[12:8] ADVRDOFFTIME field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC.GPMC_CONFIG4_i[3:0] OEONTIME field.
  - nOE deassertion time is controlled by the GPMC.GPMC_CONFIG4_i[12:8] OEOFFTIME field.

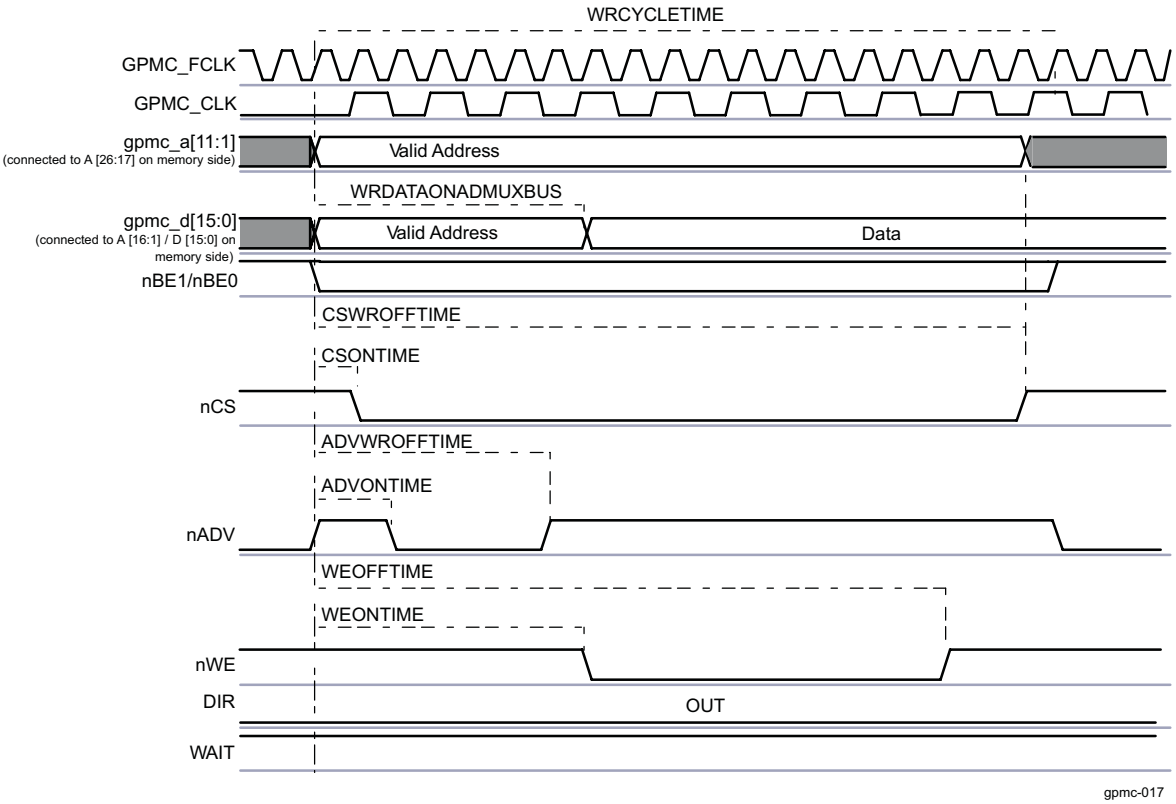
- Initial latency for the first read data is controlled by GPMC.GPMC_CONFIG5_i[20:16] RDACCESSTIME or by monitoring the WAIT signal.
- Total access time (GPMC.GPMC_CONFIG5_i[4:0] RDCYCLETIME) corresponds to RDACCESSTIME plus the address hold time from nCS deassertion, plus time from RDACCESSTIME to CSWROFFTIME.
- Direction signal DIR:

  DIR goes from OUT to IN at the same time as nOE assertion.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 9.1.5.3.10, *Bus Keeping Support*.

9.1.5.10.2 Synchronous Single Write

Figure 9-17. Synchronous Single Write on an Address/Data-Multiplexed Device



NOTE: The WAIT signal is active low.

When the GPMC generates a write access to an address/data-multiplexed device, it drives the data bus until WRDATAONADMUXBUS time (GPMC_CONFIG6_i[19:16]).

The GPMC.GPMC_CONFIG1_i register settings (I = 0 to 7) are as follows:

- WRITEMULTIPLE bit at 0 (write single access)
- WRITETYPE bit at 1 (write synchronous)
- MUXADDDATA bit at 1 (address/data-multiplexed device)

Address bits [16:1] are placed on the address/data bus at cycle-start time, and the remaining address bits [26:17] are placed on the address bus.

The address phase ends at WRDATAONADMUXBUS.

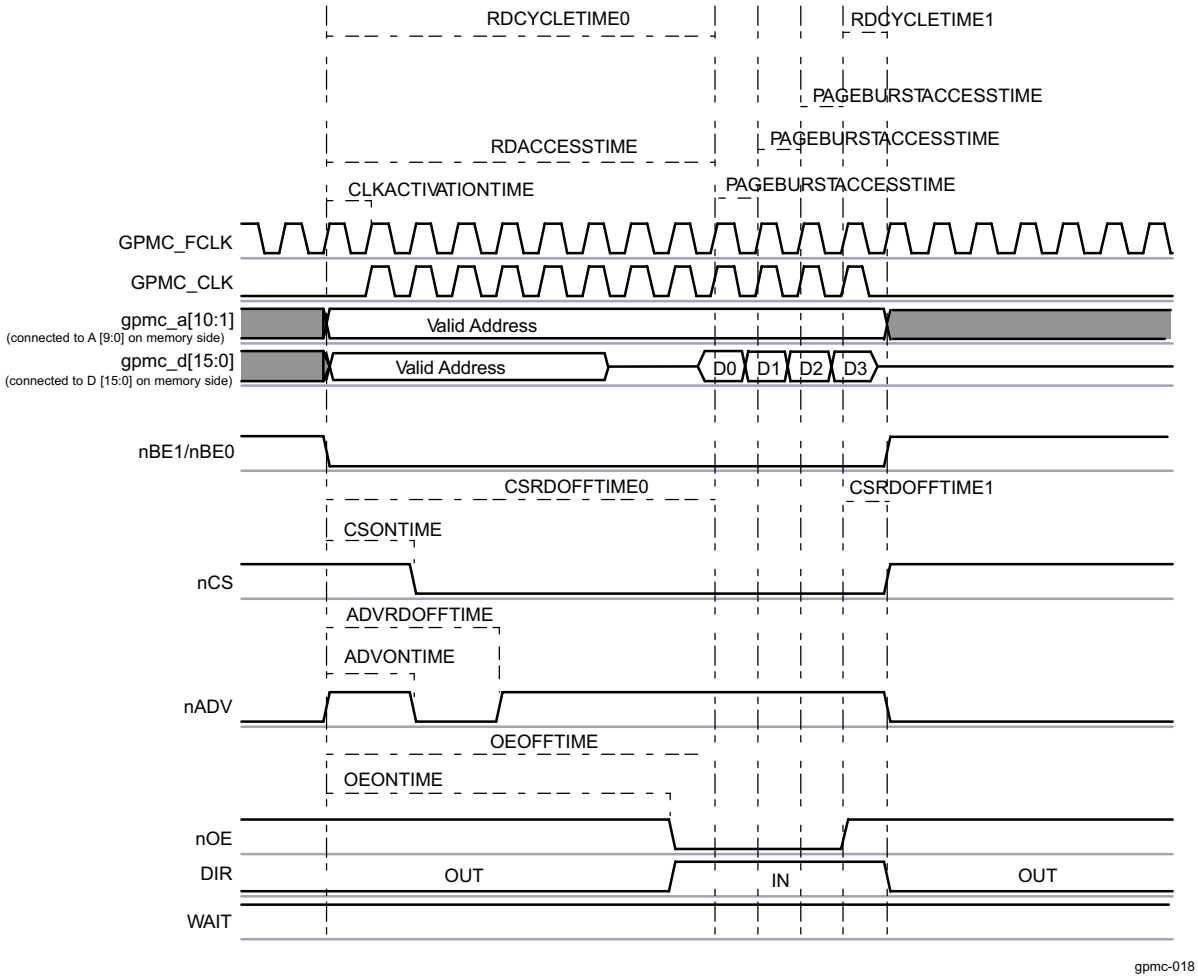
The nCS, nADV, and nWE signals are controlled in the same way as nonmultiplexed accesses.

First data of the burst is driven on the address/data bus at WRDATAONADMUXBUS time.

9.1.5.10.3 Synchronous Multiple (Burst) Read (4-, 8-, 16-Word16 Burst with Wraparound Capability)

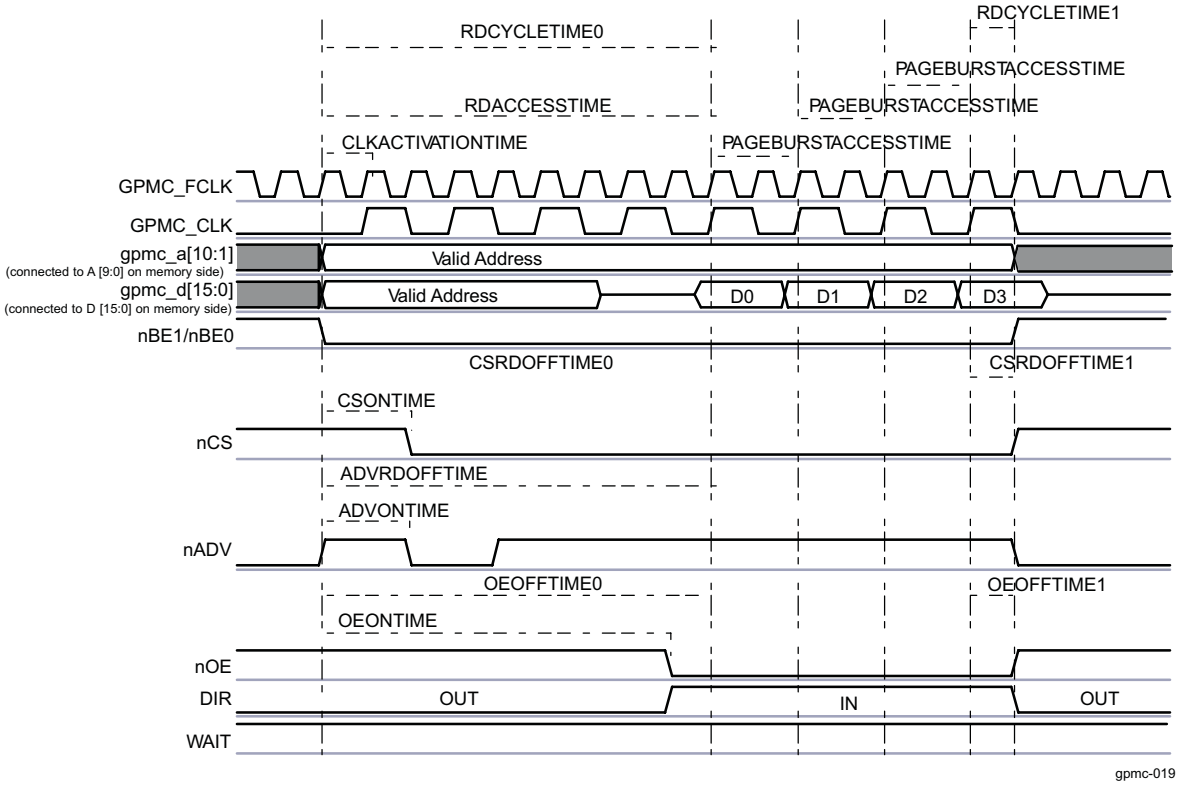
Figure 9-18 and Figure 9-19 show a synchronous multiple read operation with GPMCFCLKDivider equal to 0 and 1, respectively.

Figure 9-18. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0)



NOTE: The WAIT signal is active low.

Figure 9-19. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1)



NOTE: The WAIT signal is active low.

In the following section I stands for the chip-select number, I = 0 to 7.

- GPMC.GPMC_CONFIG1_i register settings:
  - READMULTIPLE bit at 1 (read multiple access)
  - READTYPE bit at 1 (read synchronous)
  - MUXADDDATA bit at 0 (nonaddress/data-multiplexed device)

When RDACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC.GPMC_CONFIG2_i[3:0] CSONTIME field and ensures address setup time to nCS assertion.
  - nCS deassertion time is controlled by the GPMC.GPMC_CONFIG2_i[12:8] CSRDOFFTIME field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC.GPMC_CONFIG3_i[3:0] ADVONTIME field.
  - nADV deassertion time is controlled by the GPMC.GPMC_CONFIG3_i[12:8] ADVRDOFFTIME field.
- Output enable signal nOE:
  - nOE assertion indicates a read cycle.
  - nOE assertion time is controlled by the GPMC.GPMC_CONFIG4_i[3:0] OEONTIME field.
  - nOE deassertion time is controlled by the GPMC.GPMC_CONFIG4_i[12:8] OEOFFTIME field.
- Initial latency for the first read data is controlled by GPMC.GPMC_CONFIG5_i[20:16] RDACCESSTIME or by monitoring the WAIT signal.
- Successive read data are provided by the memory device each one or two GPMC_CLK cycles. The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMCFCLKDIVIDER and the

memory-device internal configuration.

Depending on the device page length, the GPMC can control device page crossing during a new burst request and purposely insert initial latency.

- Total access time (RDCYCLETIME) corresponds to RDACCESSTIME plus the address hold time from nCS deassertion, plus the time from RDACCESSTIME to CSRDOFFTIME.
  - RDCYCLETIME is defined in the GPMC.GPMC_CONFIG5_i register.
  - In Figure 9-19, the RDCYCLETIME programmed value equals RDCYCLETIME0 + RDCYCLETIME1.
- Direction signal DIR:

  DIR goes from OUT to IN at the same time as nOE assertion.

After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See Section 9.1.5.3.10, *Bus Keeping Support*.

- Burst wraparound

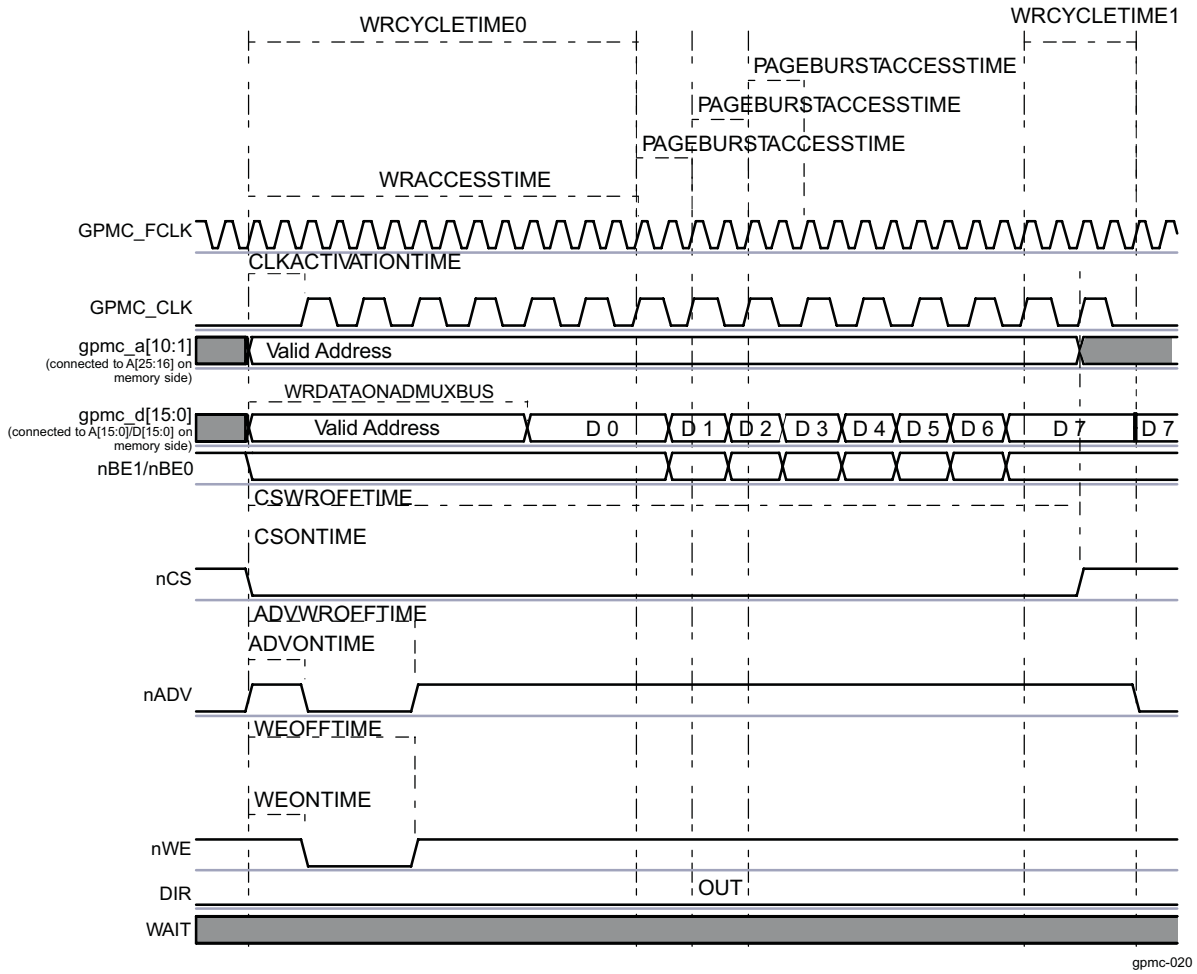
  The GPMC.GPMC_CONFIG1_i[31] WRAPBURST bit allows a 4-, 8-, or 16-Word16 linear burst access to wrap within its burst-length boundary.

9.1.5.10.4 Synchronous Multiple (Burst) Write

Burst write mode provides synchronous single or consecutive accesses.

Figure 9-20 shows a synchronous multiple (burst) write.

Figure 9-20. Synchronous Multiple (Burst) Write



NOTE: The WAIT signal is active low.

In the following section I stands for the chip-select number, I = 0 to 7.

- GPMC.GPMC_CONFIG1_i register settings:
  - WRITEMULTIPLE bit at 1 (write multiple access)
  - WRITETYPE bit at 1 (write synchronous)
  - MUXADDDATA bit at 1 (address/data-multiplexed device)

When WRACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to the PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

- Chip-select signal nCS:
  - nCS assertion time is controlled by the GPMC.GPMC_CONFIG2_i[3:0] CSONTIME field and ensures address setup time to nCS assertion.
  - nCS deassertion time controlled by the GPMC.GPMC_CONFIG2_i[20:16] CSWROFFTIME field and ensures address hold time to nCS deassertion.
- Address valid signal nADV:
  - nADV assertion time is controlled by the GPMC.GPMC_CONFIG3_i[3:0] ADVONTIME field.
  - nADV deassertion time is controlled by the GPMC.GPMC_CONFIG3_i[20:16] ADVWROFFTIME field.
- Write enable signal nWE:
  - nWE assertion indicates a write cycle.
  - nWE assertion time is controlled by the GPMC.GPMC_CONFIG4_i[19:16] WEONTIME field.
  - nWE deassertion time is controlled by the GPMC.GPMC_CONFIG4_i[28:24] WEOFFTIME field.

NOTE: The nWE falling edge must not be used to control the time when the burst first data is driven in the address / data bus because some new devices require the nWE signal at low during the address phase.

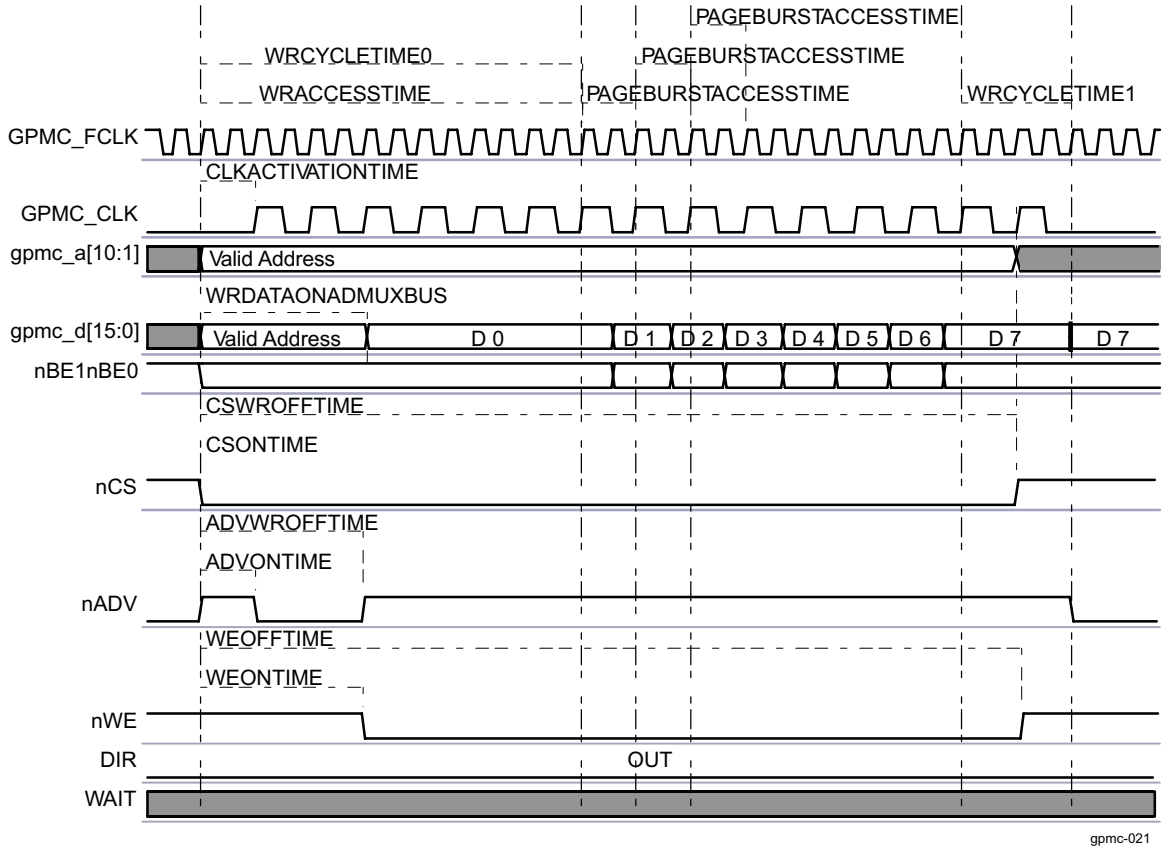
- First write data is driven by the GPMC at WRDATAONADMUXBUS (GPMC_CONFIG6_i[19:16]), when in address/data mux configuration. The next write data of the burst is driven on the bus at WRACCESSTIME + 1 during PAGEBURSTACCESSTIME GPMC_FCLK cycles. The last data of the synchronous burst write is driven until WRCYCLETIME completes.
  - WRACCESSTIME is defined in the GPMC.GPMC_CONFIG5_i register.
  - The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMCFCLKDIVIDER and the memory-device internal configuration.
- Total access time (WRCYCLETIME) corresponds to WRACCESSTIME plus the address hold time from nCS deassertion, plus time from WRACCESSTIME to CSWROFFTIME.
  - WRCYCLETIME is defined in the GPMC.GPMC_CONFIG5_i register.
  - In Figure 9-20, the WRCYCLETIME programmed value equals WRCYCLETIME0 + WRCYCLETIME1.
- Direction signal DIR:

  DIR is OUT during the entire access.

After a write operation, if no other access (read or write) is pending, the data bus keeps the previous value. See Section 9.1.5.3.10, *Bus Keeping Support*.

Figure 9-21 shows the same synchronous burst write access when the chip-select is configured in address/data-multiplexed mode.

Figure 9-21. Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode



The first data of the burst is driven on the a/d bus at GPMC_CONFIG6_i[19:16] WRDATAONADMUXBUS.

9.1.5.11 pSRAM Basic Programming Model

pSRAM devices are SRAM-pin-compatible low-power memories that contain a self-refreshed DRAM memory array. These devices can be accessed by the GPMC with the GPMC.GPMC_CONFIG1_i[11:10] DEVICETYPE field (I = 0 to 7).

The pSRAM devices support the following operations:

- Asynchronous single read
- Asynchronous page read
- Asynchronous single write
- Synchronous single read and write
- Synchronous burst read
- Synchronous burst write

pSRAM devices must be powered up and initialized in a predefined manner according to the specifications of the attached device.

pSRAM devices can be programmed to use either mode: fixed or variable latency. pSRAM devices can either automatically schedule autorefresh operations, which force the GPMC to use its WAIT signal capability when read or write operations occur during an internal self-refresh operation, or pSRAM devices automatically include the autorefresh operation in the access time. These devices do not require additional WAIT signal capability or a minimum nCS high pulse width between consecutive accesses to ensure that the correct internal refresh operation is scheduled.

In both pSRAM cases, the GPMC configuration for this chip-select must be set according to the specifications of the attached device.

9.1.5.12 Error Handling

When an error occurs in the GPMC, the error information is stored in the GPMC.GPMC_ERR_TYPE register and the address of the illegal access is stored in the GPMC.GPMC_ERR_ADDRESS register. The GPMC only keeps the first error abort information until the GPMC.GPMC_ERR_TYPE register is reset. Subsequent accesses that cause errors are not logged until the error is cleared by hardware with the GPMC.GPMC_ERR_TYPE[0] ERRORVALID bit.

- ERRORNOTSUPPADD occurs when an incoming system request address decoding does not match any valid chip-select region, or if two chip-select regions are defined as overlapped, or if a register file access is tried outside the valid address range of 1 Kbyte.
- ERRORNOTSUPPMCMD occurs when an unsupported command request is decoded at the L3 interconnect interface.
- ERRORTIMEOUT: A time-out mechanism prevents the system from hanging. The start value of the 9-bit time-out counter is defined in the GPMC.GPMC_TIMEOUT_CONTROL register and enabled with the GPMC.GPMC_TIMEOUT_CONTROL[0] TIMEOUTENABLE bit. When enabled, the counter starts at start-cycle time until it reaches 0 and data is not responded to from memory, then a time-out error occurs. When data are sent from memory, this counter is reset to its start value. With multiple accesses (asynchronous page mode or synchronous burst mode), the counter is reset to its start value for each data access within the burst.

The GPMC does not generate interrupts on these errors. True abort to the MPU or interrupt generation is handled at interconnect level.

9.1.5.13 Boot Configuration

See Section 9.1.3.3.2, *GPMC CS0 Default Configuration at IC Reset*.

9.1.5.14 NAND Device Basic Programming Model

NAND (8-bit and 16-bit) memory devices using a classical NAND asynchronous address/data-multiplexing scheme can be supported on any chip-select with the appropriate asynchronous configuration settings.

As for any other type of memory compatible with the GPMC interface, accesses to a chip-select allocated to a NAND device can be interleaved with accesses to chip-selects allocated to other external devices. This interleaved capability limits the system to chip enable don't care NAND devices since the chip-select allocated to the NAND device has to be de-asserted if accesses to other chip-selects are requested.

9.1.5.14.1 NAND Memory Device in Byte or Word16 Stream Mode

NAND devices require correct command and address programming before data array read or write accesses. The GPMC does not include specific hardware to translate a random address system request into a NAND-specific multiphase access. In that sense, GPMC NAND support, as opposed to random memory-map device support, is data-stream-oriented (byte or Word16).

The GPMC NAND programming model relies on a software driver for address and command formatting with the correct data address pointer value according to the block and page structure. Because of NAND structure and protocol interface diversity, the GPMC does not support automatic command and address phase programming, and software drivers must access the NAND device ID to ensure that correct command and address formatting are used for the identified device.

NAND device data read and write accesses are achieved through an asynchronous read or write access. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Any chip-select region can be qualified as a NAND region to constrain the nADV/ALE signal as Address Latch Enable (ALE active high, default state value at low) during address program access, and the nBE0/CLE signal as Command Latch Enable (CLE active high, default state value at low) during command program access. GPMC address lines are not used (the previous value is not changed) during NAND access.

9.1.5.14.1.1 Chip-Select Configuration for NAND Interfacing in Byte or Word Stream Mode

The GPMC.GPMC_CONFIG7_i register (where I = 0 to 7) associated with a NAND device region interfaced in byte or word stream mode can be initialized with a minimum size of 16 Mbytes, because any address location in the chip-select memory region can be used to access a NAND data array. The NAND Flash protocol specifies an address sequence where address bits are passed through the data bus in a series of write accesses with the ALE pin asserted. After this address phase, all operations are streamed and the system requests address is irrelevant.

CAUTION

To allow correct command, address, and data-access controls, the GPMC.GPMC_CONFIG1_i register associated with a NAND device region must be initialized in asynchronous read and write modes with the parameters shown in Table 9-4. Failure to comply with these settings corrupts the NAND interface protocol.

Table 9-4. Chip-Select Configuration for NAND Interfacing

Bit Field	Register	Value	Comments
WRAPBURST	GPMC_CONFIG1_i[31] (1)	0	No wrap
READMULTIPLE	GPMC_CONFIG1_i[30]	0	Single access
READTYPE	GPMC_CONFIG1_i[29]	0	Asynchronous mode
WRITEMULTIPLE	GPMC_CONFIG1_i[28]	0	Single access
WRITETYPE	GPMC_CONFIG1_i[27]	0	Asynchronous mode
CLKACTIVATIONTIME	GPMC_CONFIG1_i[26:25]	00	
ATTACHEDDEVICEPAGELENGTH	GPMC_CONFIG1_i[24:23]	Don't care	Single-access mode
WAITREADMONITORING	GPMC_CONFIG1_i[22]	0	Wait not monitored by GPMC access engine
WAITWRITEMONITORING	GPMC_CONFIG1_i[21]	0	Wait not monitored by GPMC access engine
WAITMONITORINGTIME	GPMC_CONFIG1_i[19:18]	Don't care	Wait not monitored by GPMC access engine
WAITPINSELECT	GPMC_CONFIG1_i[17:16]		Select which wait is monitored by edge detectors
DEVICESIZE	GPMC_CONFIG1_i[13:12]	0b00 or 0b01	8- or 16-bit interface
DEVICETYPE	GPMC_CONFIG1_i[11:10]	0b10	NAND device in stream mode
MUXADDDATA	GPMC_CONFIG1_i[9]	0	Nonmultiplexed mode
TIMEPARAGRANULARITY	GPMC_CONFIG1_i[4]	0	Timing achieved with best GPMC clock granularity
GPMCFCLKDIVIDER	GPMC_CONFIG1_i[1:0]	Don't care	Asynchronous mode

(1) I = 0 to 7

The GPMC.GPMC_CONFIG1_i to GPMC.GPMC_CONFIG4_i register (where I = 0 to 7) associated with a NAND device region must be initialized with the correct control-signal timing value according to the NAND device timing parameters.

9.1.5.14.1.2 NAND Device Command and Address Phase Control

NAND devices require multiple address programming phases. The CPU software driver is responsible for issuing the correct number of command and address program accesses, according to the device command set and the device address-mapping scheme.

NAND device-command and address-phase programming is achieved through write requests to the GPMC.GPMC_NAND_COMMAND_i and GPMC.GPMC_NAND_ADDRESS_i register locations (I = 0 to 7) with the correct command and address values. These locations are mapped in the associated chip-select register region. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Command and address values are not latched during the access and cannot be read back at the register location.

- Only write accesses must be issued to these locations, but the GPMC does not discard any read access. Accessing a NAND device with nOE and CLE or ALE asserted (read access) can produce undefined results.
- Write accesses to the GPMC.GPMC_NAND_COMMAND_i register location and to the GPMC.GPMC_NAND_ADDRESS_i register location must be posted for faster operations (I = 0 to 7). The GPMC.GPMC_CONFIG[0] NANDFORCEPOSTEDWRITE bit enables write accesses to these locations as posted, even if they are defined as nonposted.

A write buffer is used to store write transaction information before the external device is accessed:

- Up to eight consecutive posted write accesses can be accepted and stored in the write buffer.
- For nonposted write, the pipeline is one deep.
- An GPMC.GPMC_STATUS[0] EMPTYWRITEBUFFERSTATUS bit stores the empty status of the write buffer.

GPMC.GPMC_NAND_COMMAND_i and GPMC.GPMC_NAND_ADDRESS_i (I = 0 to 7) are Word32 locations, which means any Word32 or Word16 access is split into 4- or 2-byte accesses if an 8-bit wide NAND device is attached. For multiple-command phase or multiple-address phase, the software driver can use Word32 or Word16 access to these registers, but it must account for the splitting and little-endian ordering scheme. When only one byte command or address phase is required, only byte write access to GPMC.GPMC_NAND_COMMAND_i and GPMC.GPMC_NAND_ADDRESS_i can be used, and any of the four byte locations of the registers are valid.

The same applies to a GPMC.GPMC_NAND_COMMAND_i and a GPMC.GPMC_NAND_ADDRESS_i (I = 0 to 7) Word32 write access to a 16-bit wide NAND device (split into two Word16 accesses). In the case of a Word16 write access, the MSByte of the Word16 value must be set according to the NAND device requirement (usually 0). Either Word16 location or any one of the four byte locations of the registers is valid.

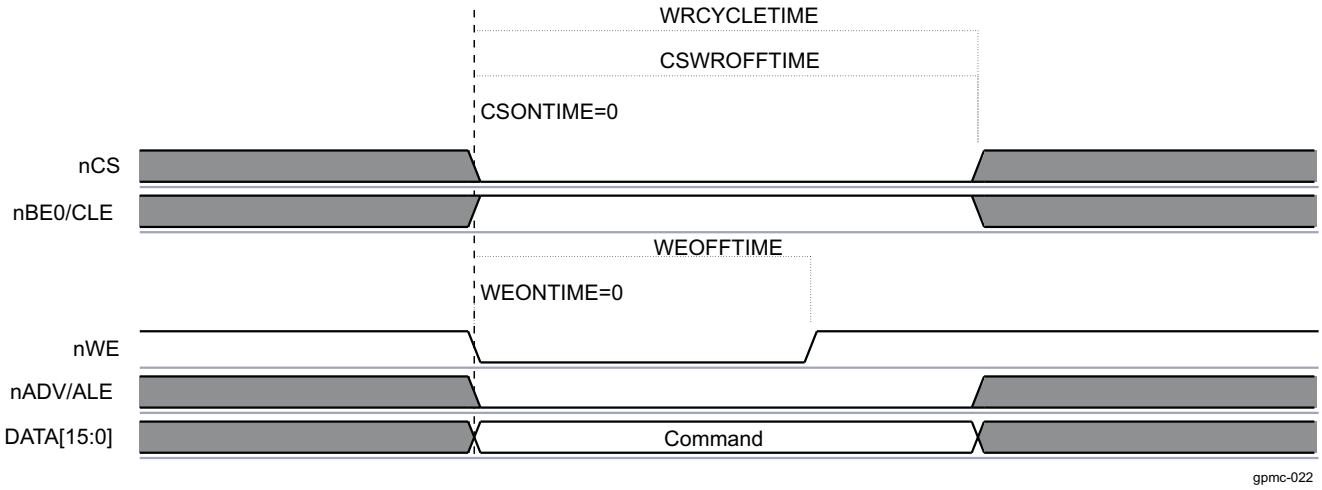
9.1.5.14.1.3 Command Latch Cycle

Writing data at the GPMC.GPMC_NAND_COMMAND_i location (I = 0 to 7) places the data as the NAND command value on the bus, using a regular asynchronous write access.

- nCE is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- CLE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE and nRE (nOE) are maintained inactive.

Figure 9-22 shows the NAND command latch cycle.

Figure 9-22. NAND Command Latch Cycle



NOTE: CLE is shared with the nBE0 output signal and has an inverted polarity from BE0. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, nBE0 (also nBE1) must not toggle, because it is shared with CLE.

NAND Flash memories do not use byte enable signals at all.

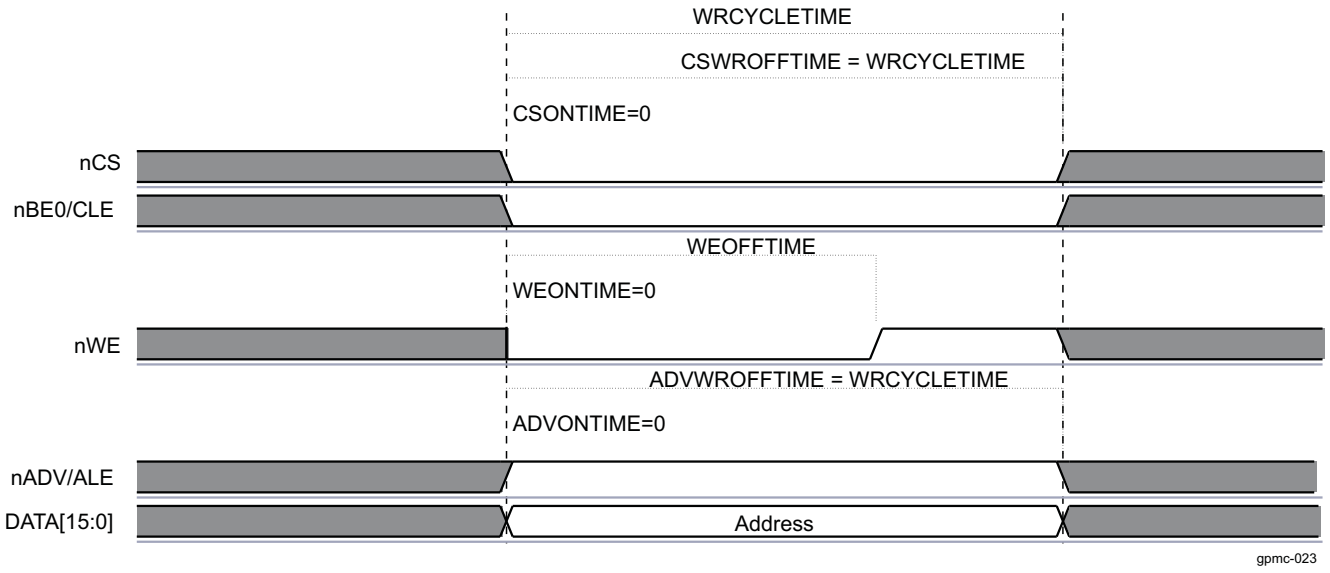
9.1.5.14.1.4 Address Latch Cycle

Writing data at the GPMC.GPMC_NAND_ADDRESS_i location (I = 0 to 7) places the data as the NAND partial address value on the bus, using a regular asynchronous write access.

- nCS is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- ALE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- CLE and nRE (nOE) are maintained inactive.

Figure 9-23 shows the NAND address latch cycle.

Figure 9-23. NAND Address Latch Cycle



NOTE: ALE is shared with the nADV output signal and has an inverted polarity from ADV. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, ALE is kept stable.

9.1.5.14.1.5 NAND Device Data Read and Write Phase Control in Stream Mode

NAND device data read and write accesses are achieved through a read or write request to the chip-select-associated memory region at any address location in the region or through a read or write request to the GPMC.GPMC_NAND_DATA_i location (I = 0 to 7) mapped in the chip-select-associated control register region. GPMC.GPMC_NAND_DATA_i is not a true register, but an address location to enable nRE or nWE signal control. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

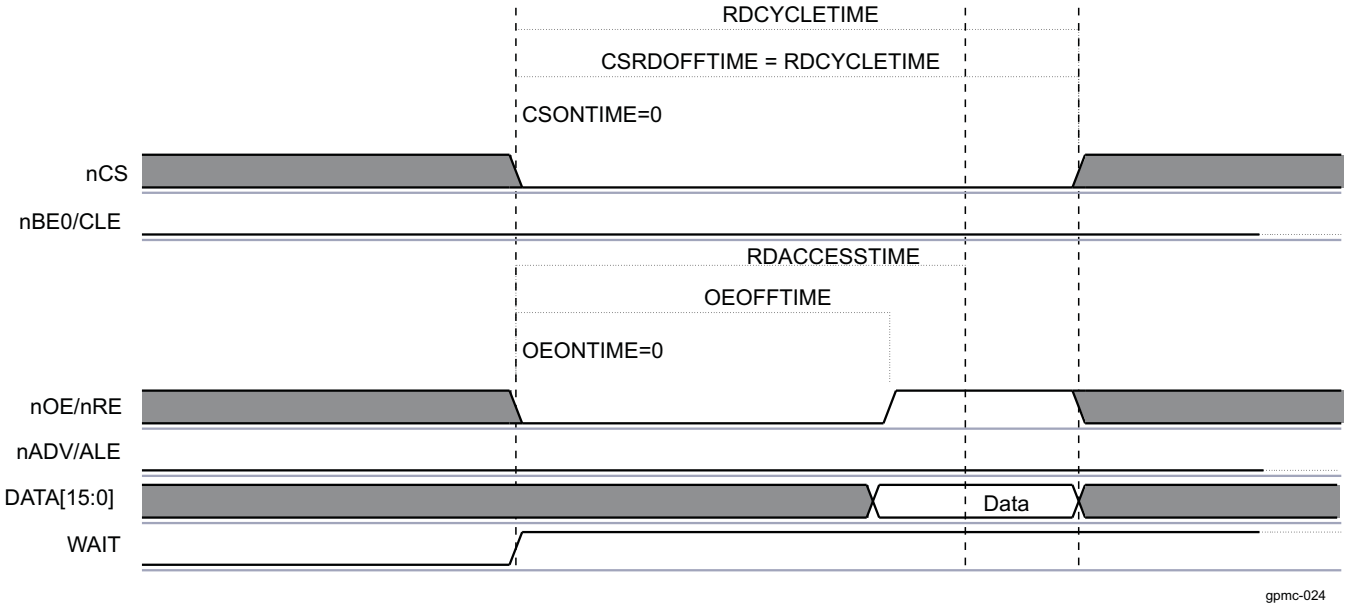
Reading data from the GPMC.GPMC_NAND_DATA_i location or from any location in the associated chip-select memory region activates an asynchronous read access.

- nCS is controlled by the CSONTIME and CSRDOFFTIME timing parameters.
- nRE is controlled by the OEONTIME and OEOFFTIME timing parameters.
- To take advantage of nRE high-to-data invalid minimum timing value, the RDACCESSTIME can be set so that data are effectively captured after nRE deassertion. This allows optimization of NAND read access cycle time completion. For optimal timing parameter settings, see the NAND device and processor IC timing parameters.

ALE, CLE, and nWE are maintained inactive.

Figure 9-24 shows the NAND data read cycle.

Figure 9-24. NAND Data Read Cycle

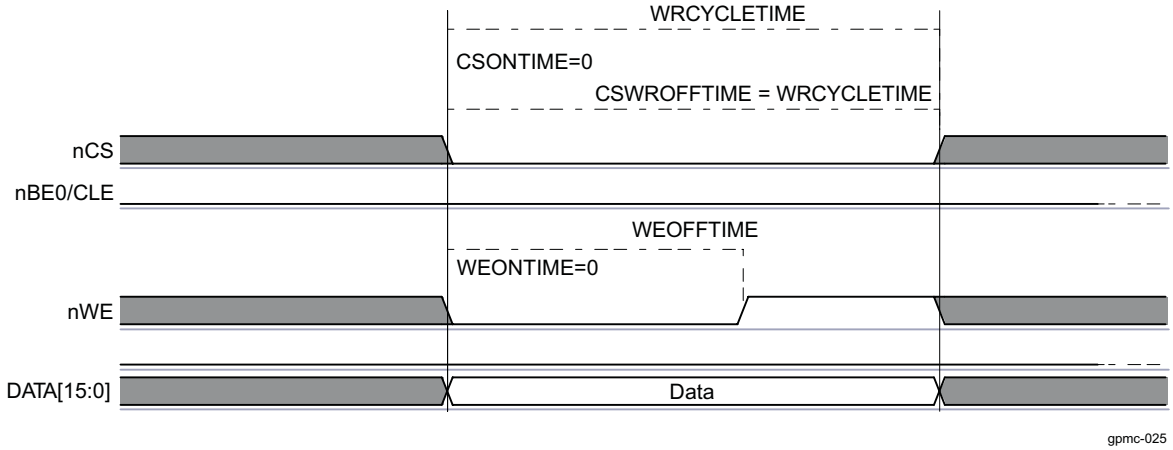


Writing data to the GPMC.GPMC_NAND_DATA_i location or to any location in the associated chip-select memory region activates an asynchronous write access.

- nCS is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- nWE is controlled by the WEONTIME and WEOFFTIME timing parameters.
- ALE, CLE, and nRE (nOE) are maintained inactive.

Figure 9-25 shows the NAND data write cycle.

Figure 9-25. NAND Data Write Cycle



9.1.5.14.1.6 NAND Device General Chip-Select Timing Control Requirement

For most NAND devices, read data access time is dominated by nCS-to-data-valid timing and has faster nRE-to-data-valid timing. Successive accesses with nCS deassertions between accesses are affected by this timing constraint. Because accesses to a NAND device can be interleaved with other chip-select accesses, there is no certainty that nCS always stays low between two accesses to the same chip-select. Moreover, an nCS deassertion time between the same chip-select NAND accesses is likely to be required as follows: the nCS deassertion requires programming CYCLETIME and RDACCESSTIME according to the nCS-to-data-valid critical timing.

To get full performance from NAND read and write accesses, the prefetch engine can dynamically reduce RDCYCLETIME, WRCYCLETIME, RDACCESSTIME, WRACCESSTIME, CSRDOFFTIME, CSWROFFTIME, ADVRDOFFTIME, ADVWROFFTIME, OEOFFTIME, and WEOFFTIME on back-to-back NAND accesses (to the same memory) and suppress the minimum nCS high pulse width between accesses. For more information about optimal prefetch engine access, see Section 9.1.5.14.4, *Prefetch and Write-Posting Engine*.

Some NAND devices require minimum write-to-read idle time, especially for device-status read accesses following status-read command programming (write access). If such write-to-read transactions are used, a minimum nCS high pulse width must be set. For this, CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDELAY must be set according to the appropriate timing requirement to prevent any timing violation.

NAND devices usually have an important nRE high to data bus in tristate mode. This requires a bus turnaround setting (BUSTURNAROUND = 1), so that the next access to a different chip-select is delayed until the BUSTURNAROUND delay completes. Back-to-back NAND read accesses to the same NAND Flash are not affected by the programmed bus turnaround delay.

9.1.5.14.1.7 Read and Write Access Size Adaptation

9.1.5.14.1.7.1 8-bit Wide NAND Device

Host Word16 and Word32 read and write access requests to a chip-select associated with an 8-bit wide NAND device are split into successive read and write byte accesses to the NAND memory device. Byte access is ordered according to little-endian organization. A NAND 8-bit wide device must be interfaced on the D0D7 interface bus lane. GPMC data accesses are justified on this bus lane when the chip-select is associated with an 8-bit wide NAND device.

9.1.5.14.1.7.2 16-bit Wide NAND Device

Host Word32 read and write access requests to a chip-select associated with a 16-bit wide NAND device are split into successive read and write Word16 accesses to the NAND memory device. Word16 access is ordered according to little-endian organization.

Host byte read and write access requests to a 16-bit wide NAND device are completed as 16-bit accesses on the device itself, because there is no byte-addressing capability on 16-bit wide NAND devices. This means that the NAND device address pointer is incremented on a Word16 basis and not on a byte basis. For a read access, only the requested byte is given back to the host, but the remaining byte is not stored or saved by the GPMC, and the next byte or Word16 read access gets the next Word16 NAND location. For a write access, the invalid byte part of the Word16 is driven to FF, and the next byte or Word16 write access programs the next Word16 NAND location.

Generally, byte access to a 16-bit wide NAND device should be avoided, especially when ECC calculation is enabled. 8-bit or 16-bit ECC-based computations are corrupted by a byte read to a 16-bit wide NAND device, because the nonrequested byte is considered invalid on a read access (not captured on the external data bus; FF is fed to the ECC engine) and is set to FF on a write access.

Host requests (read/write) issued in the chip-select memory region are translated in successive single or split accesses (read/write) to the attached device. Therefore, incrementing 32-bit burst requests are translated in multiple 32-bit sequential accesses following the access adaptation of the 32-bit to 8- or 16-bit device.

9.1.5.14.2 NAND Device-Ready Pin

The NAND memory device provides a ready pin to indicate data availability after a block/page opening and to indicate that data programming is complete. The ready pin can be connected to one of the four WAIT GPMC input pins; data read accesses must not be tried when the ready pin is sampled inactive (device is not ready) even if the associated chip-select WAITREADMONITORING bit field is set. The duration of the NAND device busy state after the block/page opening is so long (up to 50 s) that accesses occurring when the ready pin is sampled inactive can stall GPMC access and eventually cause a system time-out.

NOTE: If a read access to a NAND flash is done using the wait monitoring mode, the device is blocked during a page opening, and so is the GPMC. If the correct settings are used, other chip-selects can be used while the memory processes the page opening command.

To avoid a time-out caused by a block/page opening delay in NAND flash, disable the wait pin monitoring for read and write accesses (that is, set the GPMC.GPMC_CONFIG1_i[21] WAITWRITEMONITORING and GPMC.GPMC_CONFIG1_i[22] WAITREADMONITORING bits to 0, where I = 0 to 7), and use one of the following methods instead:

- Use software to poll the WAITnSTATUS bit (n = 0 to 3) of the GPMC_STATUS register.
- Configure an interrupt that is generated on the WAIT signal change (through the GPMC.GPMC_IRQENABLE register bits[11:8]).

Even if the READWAITMONITORING bit is not set, the external memory nR/B pin status is captured in the programmed WAIT bit in the GPMC_STATUS register.

The READWAITMONITORING bit method must be used for other memories than NAND flash, if they require the use of a WAIT signal.

9.1.5.14.2.1 Ready Pin Monitored by Software Polling

The ready signal state can be monitored through the GPMC.GPMC_STATUS WAITxSTATUS bit (x = 0 to 3). The software must monitor the ready pin only when the signal is declared valid. Refer to the NAND device timing parameters to set the correct software temporization to monitor ready only after the invalid window is complete from the last read command written to the NAND device.

9.1.5.14.2.2 Ready Pin Monitored by Hardware Interrupt

Each gpmc_wait input pin can generate an interrupt when a wait-to-no-wait transition is detected. Depending on whether the GPMC.GPMC_CONFIG WAITxPINPOLARITY bits (x = 0 to 3) is active low or active high, the wait-to-no-wait transition is a low-to-high external WAIT signal transition or a high-to-low external WAIT signal transition, respectively.

The wait transition pin detector must be cleared before any transition detection. This is done by writing 1 to the WAITxEDGEDETECTIONSTATUS bit (x = 0 to 3) of the GPMC.GPMC_IRQSTATUS register according to the gpmc_wait pin used for the NAND device-ready signal monitoring. To detect a wait-to-no-wait transition, the transition detector requires a wait active time detection of a minimum of two GPMC_FCLK cycles. Software must incorporate precautions to clear the wait transition pin detector before wait (busy) time completes.

A wait-to-no-wait transition detection can issue a GPMC interrupt if the WAITxEDGEDETECTIONENABLE bit in the GPMC.GPMC_IRQENABLE register is set and if the WAITxEDGEDETECTIONSTATUS bit field in the GPMC.GPMC_IRQSTATUS register is set.

The WAITMONITORINGTIME field does not affect wait-to-no-wait transition time detection.

It is also possible to poll the WAITxEDGEDETECTIONSTATUS bit field in the GPMC.GPMC_IRQSTATUS register according to the gpmc_wait pin used for the NAND device ready signal monitoring.

9.1.5.14.3 ECC Calculator

The General Purpose Memory Controller includes an Error Code Correction (ECC) calculator circuitry that enables on the fly ECC calculation during data read or data program (that is, write) operations.

The user can choose from two different algorithms with different error correction capabilities, Hamming code (for 1-bit error code correction) and BCH code (for 4- or 8-bit error correction) through the GPMC_ECC_CONFIG[16] ECCALGORITHM bit. Only one ECC context can be active at any given time through the GPMC_ECC_CONFIG[3:1] ECCCS bit. Even if two CS use different ECC algorithms, one the Hamming code and the other a BCH code, they must define separate ECC contexts because some of the ECC registers are common to all types of algorithms.

9.1.5.14.3.1 Hamming Code

All references to Error Code Correction (ECC) in this subsection refer to the 1-bit error correction Hamming code.

The ECC is based on a two-dimensional (row and column) bit parity accumulation known as Hamming Code. The parity accumulation is done for a programmed number of bytes or Word16 read from the memory device or written to the memory device in stream mode.

There is no automatic error detection or correction, and it is the software NAND driver responsibility to read the multiple ECC calculation results, compare them to the expected code value, and take the appropriate corrective actions according to the error handling strategy (ECC storage in spare byte, error correction on read, block invalidation).

The ECC engine includes a single accumulation context. It can be allocated to a single designated chip-select at a time and parallel computations on different chip-selects are not possible. Since it is allocated to a single chip-select, the ECC computation is not affected by interleaved GPMC accesses to other chip-selects and devices. The ECC accumulation is sequentially processed in the order of data read from or written to the memory on the designated chip-select. The ECC engine does not differentiate read accesses from write accesses and does not differentiate data from command or status information. It is the software responsibility to make sure only relevant data are passed to the NAND flash memory while the ECC computation engine is active.

The starting NAND page location must be programmed first, followed by an ECC accumulation context reset with an ECC enabling, if required. The NAND device accesses discussed in the following sections must be limited to data read or write until the specified number of ECC calculations is completed.

9.1.5.14.3.1.1 ECC Result Register and ECC Computation Accumulation Size

The GPMC includes up to nine ECC result registers (GPMC.GPMC_ECCj_RESULT, j = 1 to 9) to store ECC computation results when the specified number of bytes or Word16s has been computed.

The ECC result registers are used sequentially; one ECC result is stored in one ECC result register on the list, the next ECC result is stored in the next ECC result register on the list, and so forth, until the last ECC computation. The GPMC.GPMC_ECCj_RESULT register value is valid only when the programmed number of bytes or Word16s has been accumulated, which means that the same number of bytes or Word16s has been read from or written to the NAND device in sequence.

The GPMC.GPMC_ECC_CONTROL[3:0] ECCPOINTER field must be set to the correct value to select the ECC result register to be used first in the list for the incoming ECC computation process. The ECCPointer can be read to determine which ECC register is used in the next ECC result storage for the ongoing ECC computation. The GPMC.GPMC_ECCj_RESULT register value (j = 1 to 9) can be considered valid when ECCPOINTER equals j + 1. When GPMC.GPMC_ECCj_RESULT (where j = 9) is updated, ECCPOINTER is frozen at 10, and ECC computing is stopped (ECCENABLE = 0).

The ECC accumulator must be reset before any ECC computation accumulation process. The GPMC.GPMC_ECC_CONTROL[8] ECCCLEAR bit must be set to 1 (nonpersistent bit) to clear the accumulator and all ECC result registers.

For each ECC result (each GPMC.GPMC_ECCj_RESULT register, j = 1 to 9), the number of bytes or Word16s used for ECC computing accumulation can be selected from between two programmable values.

The ECCjRESULTSIZE bits (j = 1 to 9) in the GPMC.GPMC_ECC_SIZE_CONFIG register select which programmable size value (ECCSIZE0 or ECCSIZE1) must be used for this ECC result (stored in GPMC.GPMC_ECCj_RESULT).

The ECCSIZE0 and ECCSIZE1 fields allow selection of the number of bytes or Word16s used for ECC computation accumulation. Any even values from 2 to 512 are allowed.

Flexibility in the number of ECCs computed and the number of bytes or Word16s used in the successive ECC computations enables different NAND page error-correction strategies. Usually based on 256 or 512 bytes and on 128 or 256 Word16, the number of ECC results required is a function of the NAND device page size. Specific ECC accumulation size can be used when computing the ECC on the NAND spare byte.

For example, with a 2 Kbyte data page 8-bit wide NAND device, eight ECCs accumulated on 256 bytes can be computed and added to one extra ECC computed on the 24 spare bytes area where the eight ECC results used for comparison and correction with the computed data page ECC are stored. The GPMC then provides nine GPMC.GPMC_ECCj_RESULT registers (j) to store the results. In this case, ECCSIZE0 is set to 256, and ECCSIZE1 is set to 24; the ECC[1:8]RESULTSIZE bits are set to 0, and the ECC9RESULTSIZE bit is set to 1.

9.1.5.14.3.1.2 ECC Enabling

The GPMC.GPMC_ECC_CONFIG[3:0] ECCCS field selects the allocated chip-select. The GPMC.GPMC_ECC_CONFIG[0] ECCENABLE bit enables ECC computation on the next detected read or write access to the selected chip-select.

The ECCPOINTER, ECCCLEAR, ECCSIZE, ECCjRESULTSIZE (where j = 1 to 9), ECC16B, and ECCCS fields must not be changed or cleared while an ECC computation is in progress.

The ECC accumulator and ECC result register must not be changed or cleared while an ECC computation is in progress.

Table 9-5 describes the ECC enable settings.

Table 9-5. ECC Enable Settings

Bit Field	Register	Value	Comments
ECCCS	GPMC_ECC_CONFIG	0-7	Selects the chip-select where ECC is computed
ECC16B	GPMC_ECC_CONFIG	0/1	Selects column number for ECC calculation
ECCCLEAR	GPMC_ECC_CONTROL	0-7	Clears all ECC result registers

Table 9-5. ECC Enable Settings (continued)

Bit Field	Register	Value	Comments
ECCPOINTER	GPMC_ECC_CONTROL	0-7	A write to this bit field selects the ECC result register where the first ECC computation is stored. Set to 1 by default.
ECCSIZE1	GPMC_ECC_SIZE_CONFIG	0x00-0xFF	Defines ECCSIZE1
ECCSIZE0	GPMC_ECC_SIZE_CONFIG	0x00-0xFF	Defines ECCSIZE0
ECCkRESULTSIZE (j from 1 to 9)	GPMC_ECC_SIZE_CONFIG	0/1	Selects the size of ECCn result register
ECCENABLE	GPMC_ECC_CONFIG	1	Enables the ECC computation

9.1.5.14.3.1.3 ECC Computation

The ECC algorithm is a multiple parity bit accumulation computed on the odd and even bit streams extracted from the byte or Word 16 streams. The parity accumulation is split into row and column accumulations, as shown in Figure 9-26 and Figure 9-27. The intermediate row and column parities are used to compute the upper level row and column parities. Only the final computation of each parity bit is used for ECC comparison and correction.

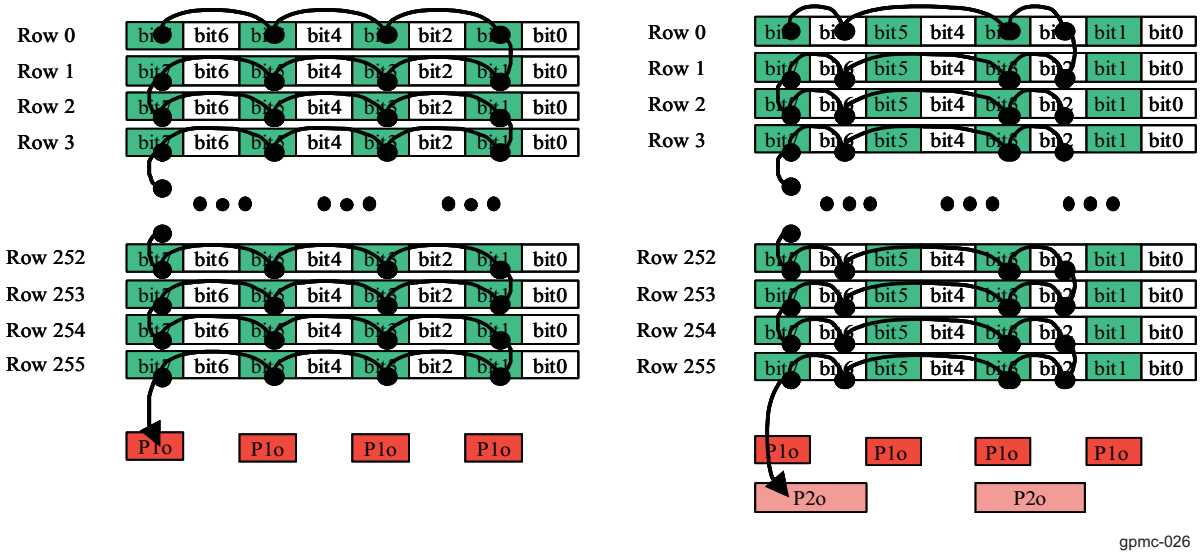
P1o = bit7 XOR bit5 XOR bit3 XOR bit1 on each byte of the data stream
P1e = bit6 XOR bit4 XOR bit2 XOR bit0 on each byte of the data stream

P2o = bit7 XOR bit6 XOR bit3 XOR bit2 on each byte of the data stream
P2e = bit5 XOR bit4 XOR bit1 XOR bit0 on each byte of the data stream

P4o = bit7 XOR bit6 XOR bit5 XOR bit4 on each byte of the data stream
P4e = bit3 XOR bit2 XOR bit1 XOR bit0 on each byte of the data stream

Each column parity bit is XORed with the previous accumulated value.

Figure 9-26. Hamming Code Accumulation Algorithm (½)

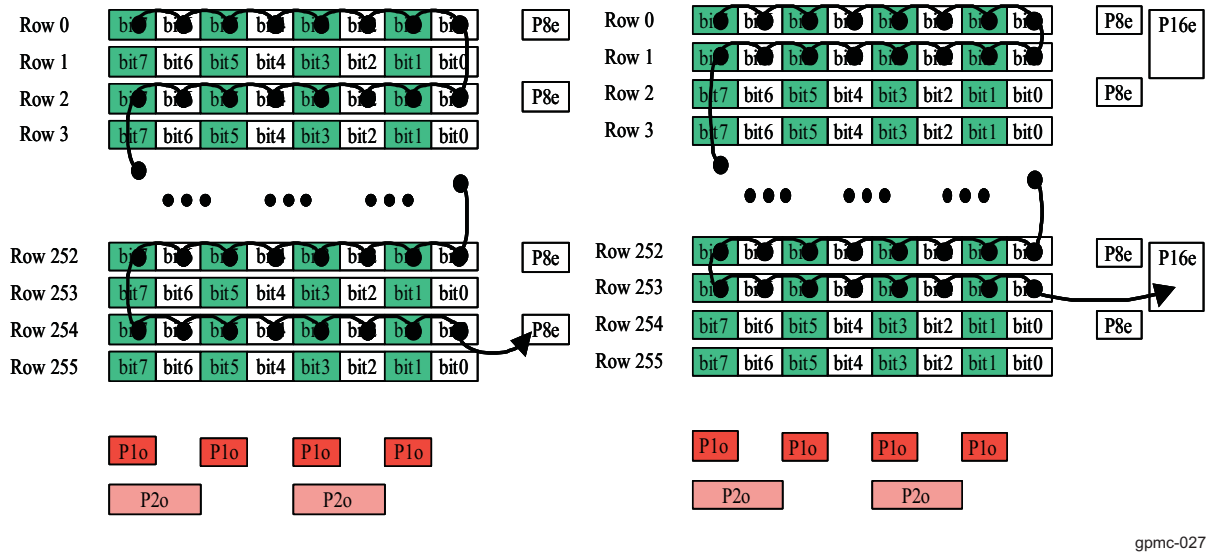


For line parities, the bits of each new data are XORed together, and line parity bits are computed as described below:

P8e = row0 XOR row2 XOR row4 XOR XOR row254
P8o = row1 XOR row3 XOR row5 XOR XOR row255

P16e = row0 XOR row1 XOR row4 XOR row5 XOR XOR row252 XOR row 253
P16o = row2 XOR row3 XOR row6 XOR row7 XOR XOR row254 XOR row 255

Figure 9-27. Hamming Code Accumulation Algorithm (2/2)



Unused parity bits in the result registers are set to 0.

Figure 9-28 shows ECC computation for a 256-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and sixteen row parity bits (P8o-P16o-P32o--P1024o for odd parities, and P8e-P16e-P32e--P1024e for even parities).

Figure 9-28. ECC Computation for a 256-Byte Data Stream (Read or Write)

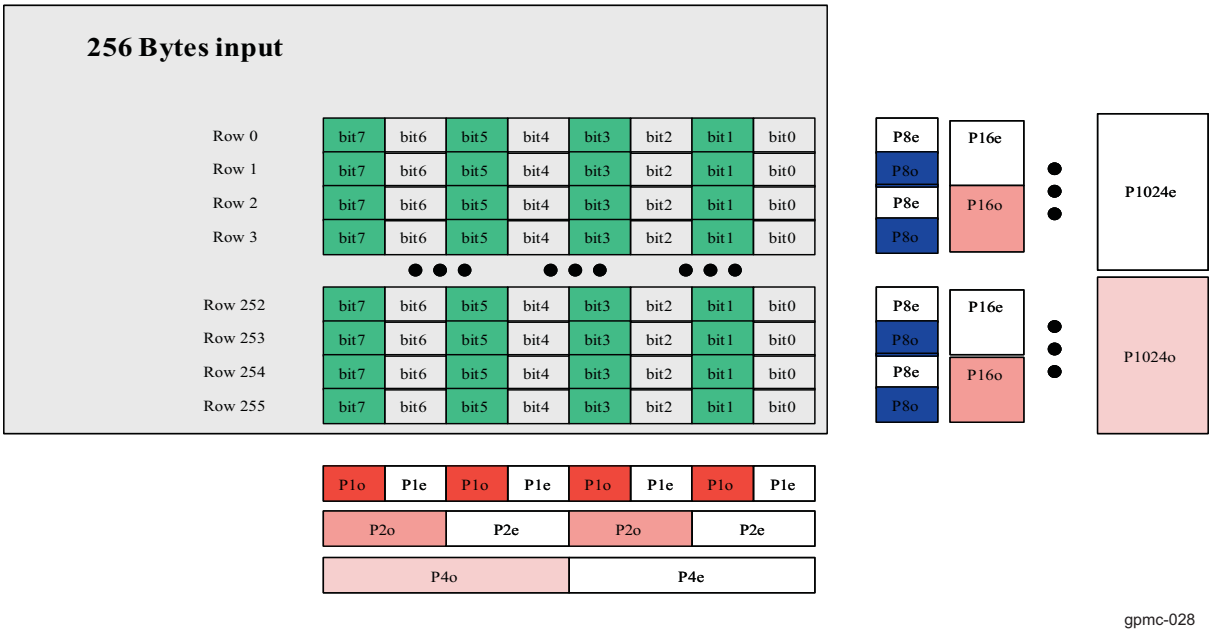
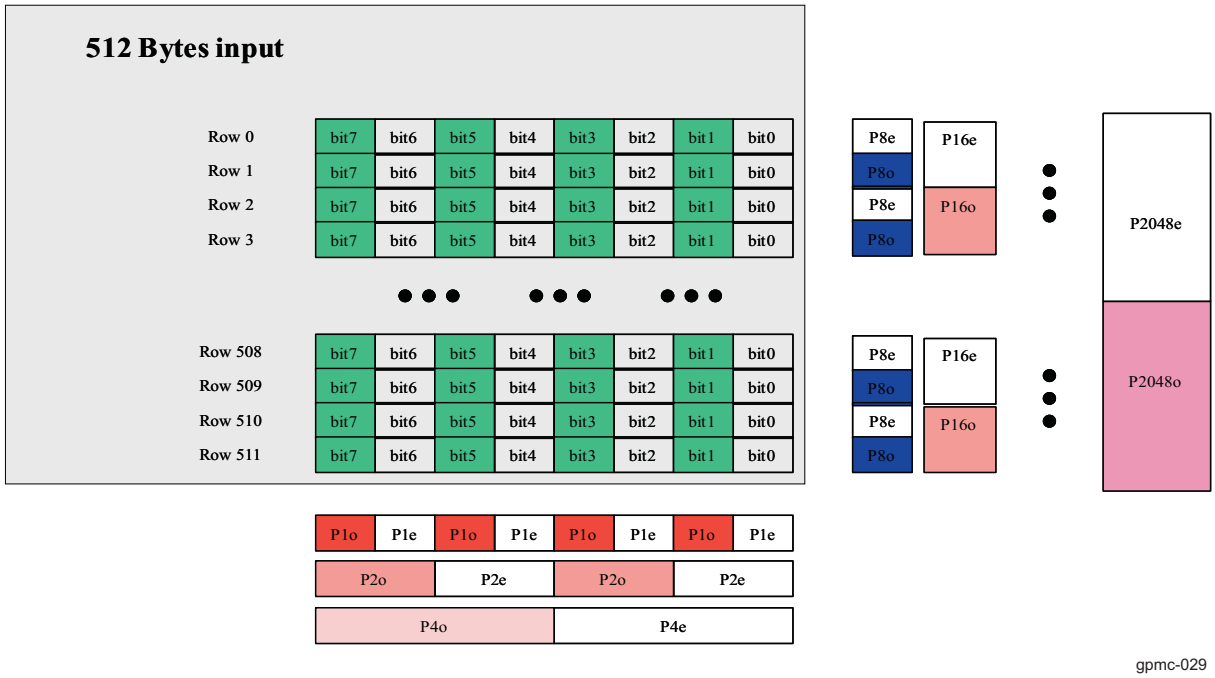


Figure 9-29 shows ECC computation for a 512-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and eighteen row parity bits (P8o-P16o-P32o--P1024o- - P2048o for odd parities, and P8e-P16e-P32e--P1024e- P2048e for even parities).

Figure 9-29. ECC Computation for a 512-Byte Data Stream (Read or Write)



For a 2 Kbytes page, four 512 bytes ECC calculations plus one for the spare area are required. Results are stored in the GPMC_ECCj_RESULT registers (j = 1 to 9).

9.1.5.14.3.1.4 ECC Comparison and Correction

To detect an error, the computed ECC result must be XORed with the parity value stored in the spare area of the accessed page.

- If the result of this logical XOR is all 0s, no error is detected and the read data is correct.
- If every second bit in the parity result is a 1, one bit is corrupted and is located at bit address (P2048o, P1024o, P512o, P256o, P128o, P64o, P32o, P16o, P8o, P4o, P2o, P1o). The software must correct the corresponding bit.
- If only one bit in the parity result is 1, it is an ECC error and the read data is correct.

9.1.5.14.3.1.5 ECC Calculation Based on 8-Bit Word

The 8-bit based ECC computation is used for 8-bit wide NAND device interfacing.

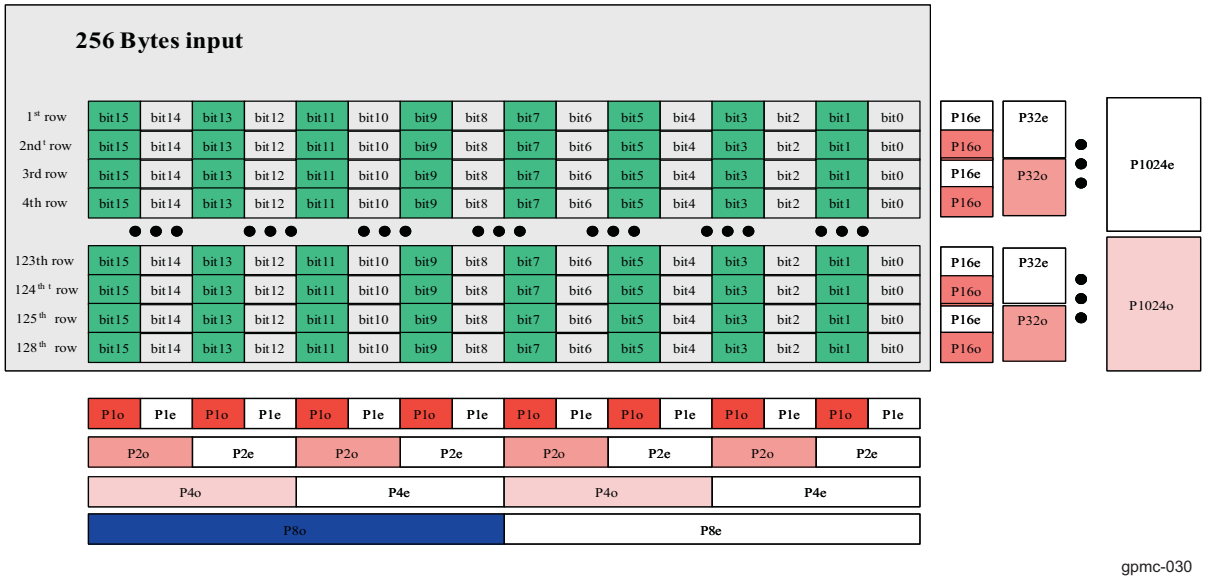
The 8-bit based ECC computation can be used for 16-bit wide NAND device interfacing to get backward compatibility on the error-handling strategy used with 8-bit wide NAND devices. In this case, the 16-bit wide data read from or written to the NAND device is fragmented into 2 bytes. According to little-endian access, the least significant bit (LSB) of the 16-bit wide data is ordered first in the byte stream used for 8-bit based ECC computation.

9.1.5.14.3.1.6 ECC Calculation Based on 16-Bit Word

ECC computation based on a 16-bit word is used for 16-bit wide NAND device interfacing. This ECC computation is not supported when interfacing an 8-bit wide NAND device, and the GPMC.GPMC_ECC_CONFIG[7] ECC16B bit must be set to 0 when interfacing an 8-bit wide NAND device.

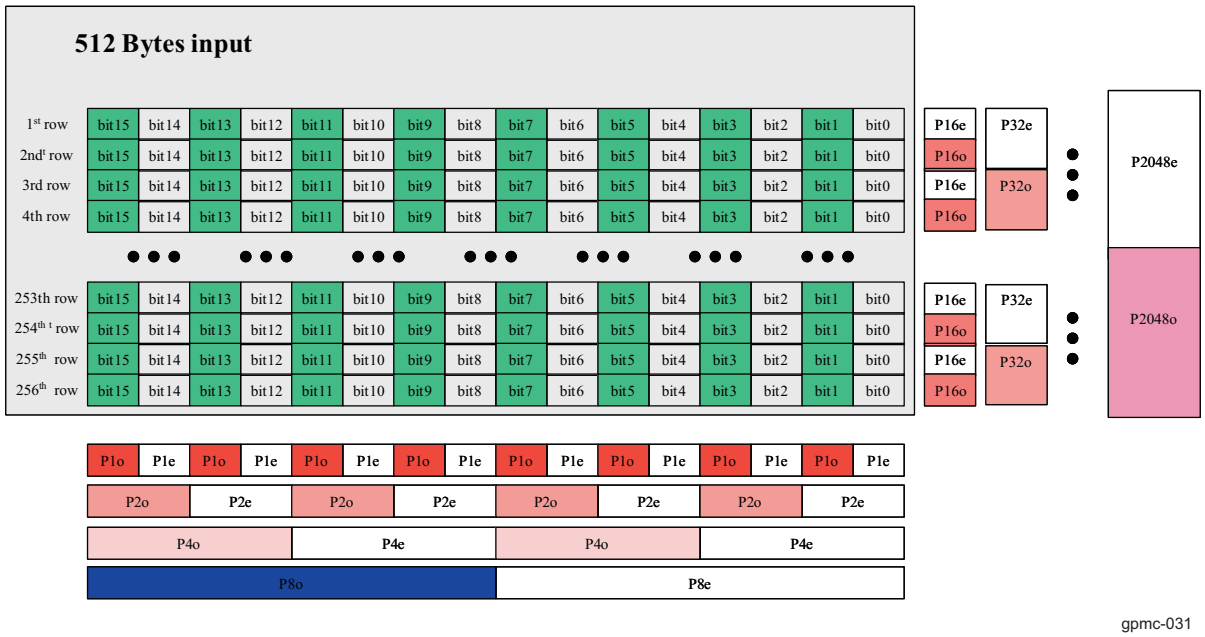
The parity computation based on 16-bit words affects the row and column parity mapping. The main difference is that the odd and even parity bits P8o and P8e are computed on rows for an 8-bit based ECC while there are computed on columns for a 16-bit based ECC. Figure 9-30 and Figure 9-31 show a 128 Word 16 ECC computation scheme and a 256 Word 16 ECC computation scheme.

Figure 9-30. 128 Word 16 ECC Computation



gpmc-030

Figure 9-31. 256 Word 16 ECC Computation



gpmc-031

9.1.5.14.3.2 BCH Code

All references to Error Code Correction (ECC) in this subsection refer to the 4- or 8-bit error correction BCH code.

9.1.5.14.3.2.1 Requirements

1. Read and write accesses to a NAND flash take place by whole pages, in a predetermined sequence: first the data byte page itself, then some spare bytes, including the BCH ECC (and other information). The NAND IC can cache a full page, including spares, for read and write accesses.
   Typical page write sequence:
   - Sequential write to NAND cache of main data + spare data, for a page. ECC is calculated on the fly. Calculated ECC may be inserted on the fly in the spares, or replaced by dummy accesses.

- When the calculated ECC is replaced by dummy accesses, it must be written to the cache in a second, separate phase. The ECC module is disabled during that time.
- NAND writes its cache line (page) to the array.

Typical page read sequence:

- Sequential read of a page. ECC is calculated on the fly.
- ECC module buffers status determines the presence of errors.

2. Accesses to several memories may be interleaved by the GPMC, but only one of those memories can be a NAND using the BCH engine at a time; in other words, only one BCH calculation (for example, for a single page) can be on-going at any time. Note also that the sequential nature of NAND accesses guarantees that the data is always written / read out in the same order. BCH-relevant accesses are selected by the GPMCs chip-select.
3. Each page may hold up to 4 Kbytes of data, spare bytes not included. This means up to 8 × 512-byte BCH messages. Since all the data is written / read out first, followed by the BCH ECC, this means that the BCH engine must be able to hold 8 104-bit remainders or syndromes (or smaller, 52-bit ones) at the same time.
   The BCH module has the capacity to store all remainders internally. After the page start, an internal counter is used to detect the 512-byte sector boundaries. On those boundaries, the current remainder is stored and the divider reset for the next calculation. At the end of the page, the BCH module contains all remainders.
4. NAND access cycles hold 8 or 16 bits of data each (1 or 2 bytes); Each NAND cycle takes at least 4 cycles of the GPMCs internal clock. This means the NAND flash timing parameters must define a RDCYCLETIME and a WRCYCLETIME of at least 4 clock cycles after optimization when using the BCH calculator.
5. The spare area is assumed to be large enough to hold the BCH ECC, that is, to have at least a message of 13 bytes available per 512-byte sector of data. The zone of unused spare area by the ECC may or may not be protected by the same ECC scheme, by extending the BCH message beyond 512 bytes (maximum codeword is 1023-byte long, ECC included, which leaves a lot of space to cover some spares bytes).

9.1.5.14.3.2.2 Memory-Mapping of the BCH Codeword

BCH encoding considers a block of data to protect as a polynomial message M(x). In our standard case, 512 bytes of data (that is, 2^{12} bits = 4096 bits) are seen as a polynomial of degree 2^{12} - 1 = 4095, with parameters ranging from M0 to M4095. For 512 bytes of data, 52 bits are required for 4-bit error correction, and 104 bits are required for 8-bit error correction. The ECC is a remainder polynomial R(x) of degree 103 (or 51, depending on the selected mode). The complete codeword C(x) is the concatenation of M(x) and R(x) as shown in Table 9-6.

Table 9-6. Flattened BCH Codeword Mapping (512 Bytes + 104 Bits)

	Message M(x)			ECC R(x)		
Bit number	M4095	...	M0	R103	...	R0

If the message is extended by the addition of spare bytes to be protected by the same ECC, the principle is still valid. For example, a 3-byte extension of the message gives a polynomial message M(x) of degree ((512 + 3) * 8) - 1 = 4119, for a total of 3+13 = 16 spare bytes of spare, all protected as part of the same codeword.

The message and the ECC bits are manipulated and mapped in the GPMC byte-oriented system. The ECC bits are stored in GPMC_BCH_RESULT0_i, GPMC_BCH_RESULT1_i, GPMC_BCH_RESULT2_i, and GPMC_BCH_RESULT3_i (where I = 0 to 7).

9.1.5.14.3.2.2.1 Memory-Mapping of the Data Message

The data message mapping shall follow the following rules:

- Bit endianness within a byte is little-endian, that is, the bytes LS bit is also the lowest-degree polynomial parameter: a byte b7-b0 (with b0 the LS bit) represents a segment of polynomial b7 * $x^{(7+I)}$ + b6 * $x^{(6+I)}$ + ... + b0 * x^{I}

- The message is mapped in the NAND starting with the highest-order parameters, that is, in the lowest addresses of a NAND page.
- Byte endianness within the NANDs 16-bit words is big endian. This means that the same message mapped in 8- and 16-bit memories has the same content at the same byte address.

NOTE: The BCH module has no visibility over actual addresses. The most important point is the sequence of data word the BCH sees. However, the NAND page is always scanned incrementally in read and write accesses, and this produces the mapping patterns described below.

The following tables represent the mapping of the same 512-byte vector (typically a BCH message) in the NANDs memory space. Note that the byte 'address' is only an offset modulo 512 (0x200), since the same page may contain several contiguous 512-byte sectors (BCH blocks). The LSB and MSB are respectively the bits M0 and M(2^{12}-1) of the codeword mapping given above. In both cases the data vectors are aligned, that is, their boundaries coincide with the RAMs data word boundaries.

Table 9-7. Aligned Message Byte Mapping in 8-bit NAND

Byte offset	8-bit word
0x000	(msb) Byte 511 (0x1FF)
0x001	Byte 510 (0x1FE)
...	...
0x1FF	Byte 0 (0x0) (lsb)

Table 9-8. Aligned Message Byte Mapping in 16-bit NAND

Byte offset	16-bit words MSB	16-bit words LSB
0x000	Byte 510 (0x1FE)	(msb) Byte 511 (0x1FF)
0x002	Byte 508 (0x1FC)	Byte 509 (0x1FD)
...	...	...
0x1FE	Byte 0 (0x0)	(lsb) Byte 1 (0x1)

The following tables show the mapping in memory of arbitrarily-sized messages, starting on access (byte or 16-bit word) boundaries for more clarity. Note that message may actually start and stop on arbitrary nibbles. A nibble is a 4-bit entity. The unused nibbles are not discarded, and they can still be used by the BCH module, but as part of the next message section (for example, on another sectors ECC).

Table 9-9. Aligned Nibble Mapping of Message in 8-bit NAND

Byte offset	8-bit word	
	4-bit most significant Nibble	4-bit less significant Nibble
1	(msb) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
...	...	...
S/2 - 2	Nibble 3	Nibble 2
S/2 - 1	Nibble 1	Nibble 0 (lsb)

Table 9-10. Misaligned Nibble Mapping of Message in 8-bit NAND

Byte offset	8-bit word	
	4-bit most significant Nibble	4-bit less significant Nibble
1	(msb) Nibble S-1	Nibble S-2
2	Nibble S-3	Nibble S-4
...	...	...
(S+1)/2 - 2	Nibble 2	Nibble 1
(S+1)/2 - 1	Nibble 0 (lsb)	

Table 9-11. Aligned Nibble Mapping of Message in 16-bit NAND

Byte offset	16-bit word			
	4-bit most significant Nibble		4-bit less significant Nibble	
0	Nibble S-3	Nibble S-4	(msb) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
S/2 - 4	Nibble 5	Nibble 4	Nibble 7	Nibble 6
S/2 - 2	Nibble 1	Nibble 0 (lsb)	Nibble 3	Nibble 2

Table 9-12. Misaligned Nibble Mapping of Message in 16-bit NAND (1 Unused Nibble)

Byte offset	16-bit word			
	4-bit most significant Nibble		4-bit less significant Nibble	
0	Nibble S-3	Nibble S-4	(msb) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
(S+1)/2 - 4	Nibble 4	Nibble 3	Nibble 6	Nibble 5
(S+1)/2 - 2	Nibble 0 (lsb)		Nibble 2	Nibble 1

Table 9-13. Misaligned Nibble Mapping of Message in 16-bit NAND (2 Unused Nibbles)

Byte offset	16-bit word			
	4-bit most significant Nibble		4-bit less significant Nibble	
0	Nibble S-3	Nibble S-4	(msb) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
(S+2)/2 - 4	Nibble 3	Nibble 2	Nibble 5	Nibble 4
(S+2)/2 - 2			Nibble 1	Nibble 0 (lsb)

Table 9-14. Misaligned Nibble Mapping of Message in 16-bit NAND (3 Unused Nibbles)

Byte offset	16-bit word			
	4-bit most significant Nibble		4-bit less significant Nibble	
0	Nibble S-3	Nibble S-4	(msb) Nibble S-1	Nibble S-2
2	Nibble S-7	Nibble S-8	Nibble S-5	Nibble S-6
...	...		...	
(S+3)/2 - 4	Nibble 2	Nibble 1	Nibble 4	Nibble 3
(S+3)/2 - 2			Nibble 0 (lsb)	

Note that many other cases exist than the ones represented above, for example, where the message does not start on a word boundary.

9.1.5.14.3.2.2.2 Memory-Mapping of the ECC

The ECC (or remainder) is presented by the BCH module as a single 104-bit (or 52-bit), little-endian vector. It is up to the software to fetch those 13 bytes (or 6 ½ bytes) from the modules interface, then store them to the NANDs spare area (page write) or to an intermediate buffer for comparison with the stored ECC (page read). There are no constraints on the ECC mapping inside the spare area: it is a software controlled operation.

However, it is advised to maintain a coherence in the respective formats of the message or the ECC remainder once they have been read out of the NAND. The error correction algorithm works from the complete codeword (concatenated message and remainder) once an error as been detected. The creation of this codeword should be made as straightforward as possible.

There are cases where the same NAND access contains both data and the ECC protecting that data. This is the case when the data/ECC boundary (which can be on any nibble) does not coincide with an access boundary. The ECC is calculated on-the-fly following the write. In that case, the write must also contain part of the ECC because it is impossible to insert the ECC on-the-fly. Instead:

- During the initial page write (BCH encoding), the ECC is replaced by dummy bits. The BCH encoder is by definition turned OFF during the ECC section, so the BCH result is unmodified.
- During a second phase, the ECC is written to the correct location, next to the actual data.
- The completed line buffer is then written to the NAND array.

9.1.5.14.3.2.2.3 Wrapping Modes

For a given wrapping mode, the module automatically goes through a specific number of sections, as data is being fed into the module. For each section, the BCH core can be enabled (in which case the data is fed to the BCH divider) or not (in which case the BCH simply counts to the end of the section). When enabled, the data is added to the ongoing calculation for a given sector number (for example, number 0).

Wrapping modes are described below. To get a better understanding and see the real-life read and write sequences implemented with each mode, see Section 9.1.5.14.3.2.3.

For each mode:

- a sequence describes the mode in pseudo-language, with for each section the size and the buffer used for ECC processing (if ON). The programmable lengths are size, size0 and size1.
- a checksum condition is given. If the checksum condition is not respected for a given mode, the modules behavior is unpredictable. S is the number of sectors in the page; size0 and size1 are the section sizes programmed for the mode, in nibbles.

Note that wrapping modes 8, 9, 10, and 11 insert a 1-nibble padding where the BCH processing is OFF. This is intended for t = 4 ECC, where ECC is 6 ½ bytes long and the ECC area is expected to include (at least) 1 unused nibble to remain byte-aligned.

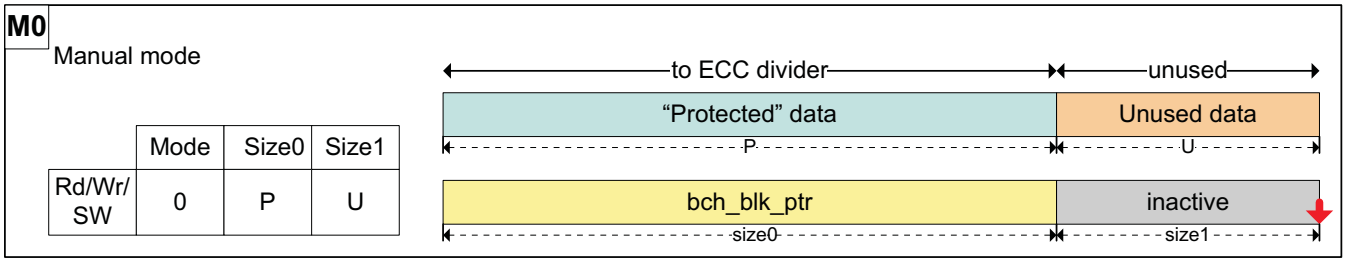
9.1.5.14.3.2.2.4 Manual Mode (0x0)

This mode is intended for short sequences, added manually to a given buffer through the software data port input. A complete page may be built out of several such sequences.

To process an arbitrary sequence of 4-bit nibbles, accesses to the software data port shall be made, containing the appropriate data. If the sequence end does not coincide with an access boundary (for example, to process 5 nibbles = 20 bits in 16-bit access mode) and those nibbles need to be skipped, a number of unused nibbles shall be programmed in size1 (in the same example: 5 nibbles to process + 3 to discard = 8 nibbles = exactly 2 x 16-bit accesses: we must program size0 = 5, size1 = 3).

NOTE: In the following figures size and size0 are the same parameter.

Figure 9-32. Manual Mode Sequence and Mapping



Section processing sequence:

- One time with buffer
  - size0 nibbles of data, processing ON
  - size1 nibbles of unused data, processing OFF

Checksum: size0 + size1 nibbles must fit in a whole number of accesses.

9.1.5.14.3.2.2.5 Mode 0x1

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.1.5.14.3.2.2.6 Mode 0xA (10)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - 1 nibble pad spare, processing OFF
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + 1 + size1)

9.1.5.14.3.2.2.7 Mode 0x2

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.1.5.14.3.2.2.8 Mode 0x3

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - size1)

9.1.5.14.3.2.2.9 Mode 0x7

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = size0 + (S - size1)

9.1.5.14.3.2.2.10 Mode 0x8

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - (1+size1))

9.1.5.14.3.2.2.11 Mode 0x4

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - size1)

9.1.5.14.3.2.2.12 Mode 0x9

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - (1+size1))

9.1.5.14.3.2.2.13 Mode 0x5

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.1.5.14.3.2.2.14 Mode 0xB (11)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + 1 + size1)

9.1.5.14.3.2.2.15 Mode 0x6

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + size1)

9.1.5.14.3.2.3 Supported NAND Page Mappings and ECC Schemes

The following rules apply throughout the entire mapping description:

- Main data area (sectors) size is hardcoded to 512 bytes.
- Spare area size is programmable.
- All page sections (of main area data bytes, protected spare bytes, unprotected spare bytes, and ECC) are defined as explained in Section 9.1.5.14.3.2.2.1.

Each one of the following sections shows a NAND page mapping example (per-sector spare mappings, pooled spare mapping, per-sector spare mapping, with ECC separated at the end of the page).

In the mapping diagrams, sections that belong to the same BCH codeword have the same color (blue or green); unprotected sections are not covered (orange) by the BCH scheme.

Below each mapping diagram, a write (encoding) and read (decoding: syndrome generation) sequence is given, with the number of the active buffers at each point in time (yellow). In the inactive zones (grey), no computing is taking place but the data counter is still active.

A table on the left summarizes the mode, size0, size1 parameters to program for respectively write and read processing of a page, with the given mapping, where :

- P is the size of spare byte section Protected by the ECC (in nibbles)
- U is the size of spare byte section Unprotected by the ECC (in nibbles)
- E is the size of the ECC itself (in nibbles)
- S is the number of Sectors per page (2 in the current diagrams)

Each time the processing of a BCH block is complete (ECC calculation for write/encoding, syndrome generation for read/decoding, indicated by red arrows), the update pointer is pulsed. Note that the processing for block 0 can be the first or the last to complete, depending on the NAND page mapping and operation (read or write). All examples show a page size of 1kByte + spares, that is, S = 2 sectors of 512 bytes. The same principles can be extended to larger pages by adding more sectors.

The actual BCH codeword size is used during the error location work to restrict the search range: by definition, errors can only happen in the codeword that was actually written to the NAND, and not in the mathematical codeword of $n = 2^{13} - 1 = 8191$ bits. That codeword (higher-order bits) is all-zero and implicit during computations.

The actual BCH codeword size depends on the mode, on the programmed sizes and on the sector number (all sizes in nibbles):

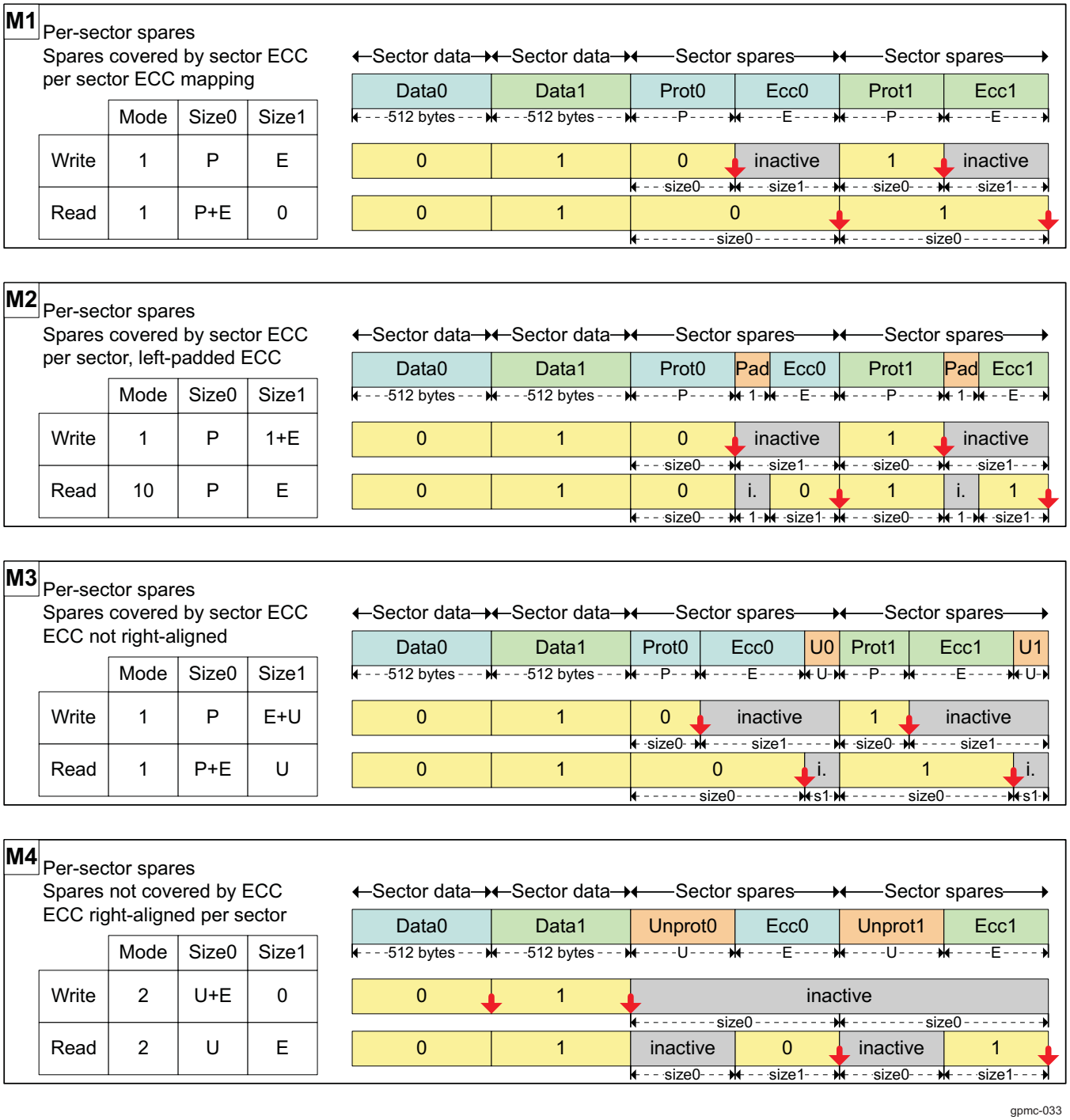
- Spares mapped and protected per sector (below: see M1-M2-M3-M9-M10):
  - all sectors: (512) + P + E
- Spares pooled and protected by sector 0 (below: see M5-M6):
  - sector 0 codeword: (512) + P + E
  - other sectors: (512) + E
- Unprotected spares (below: see M4-M7-M8-M11-M12):
  - all codewords (512) + E

9.1.5.14.3.2.3.1 Per-Sector Spare Mappings

In these schemes, each 512-byte sector of the main area has its own dedicated section of the spare area. The spare area of each sector is composed of :

- ECC, which must be located after the data it protects
- other data, which may or may not be protected by the sectors ECC

Figure 9-33. NAND Page Mapping and ECC: Per-Sector Schemes

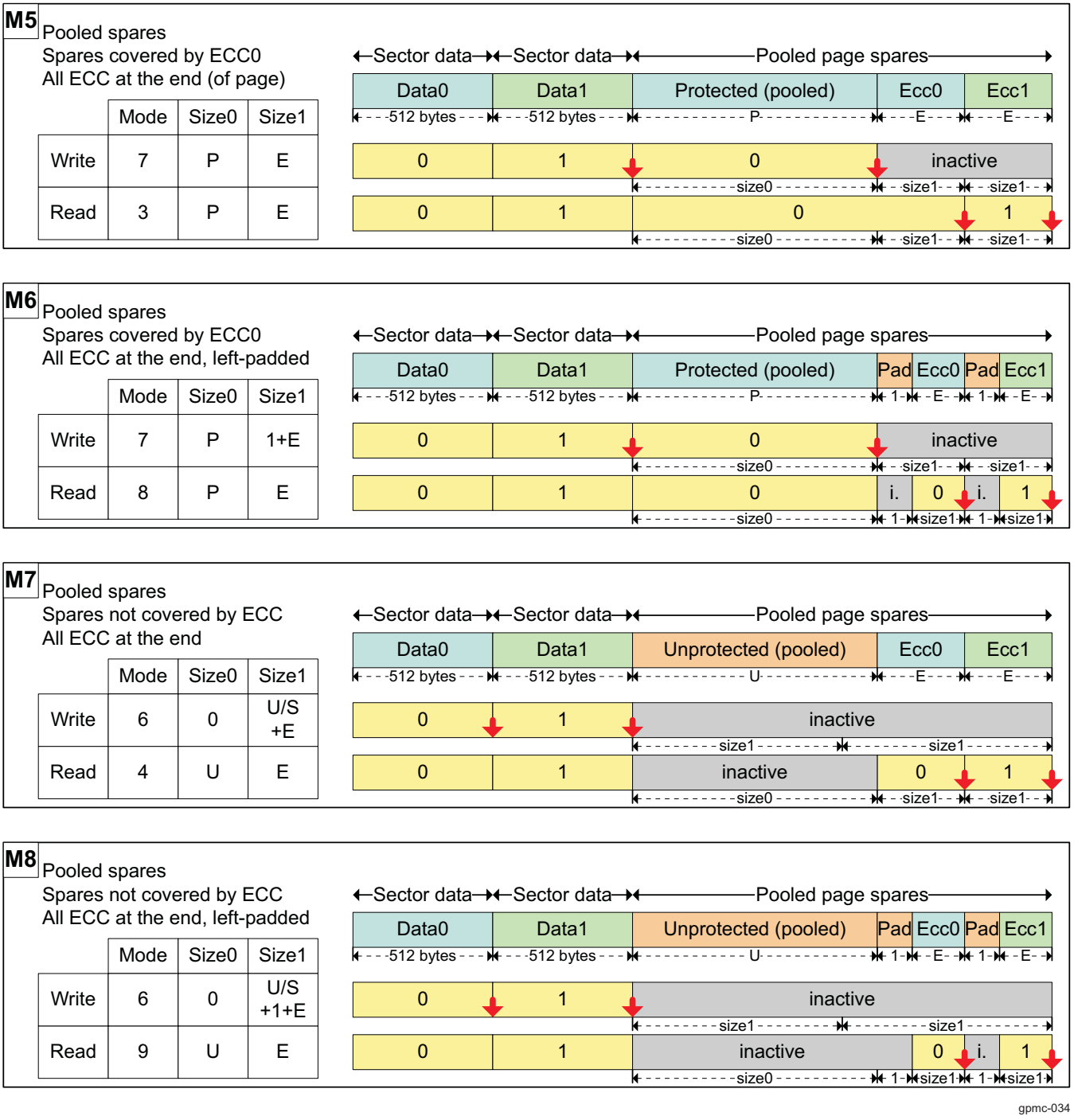


9.1.5.14.3.2.3.2 Pooled Spare Mapping

In the schemes below, the spare area is pooled for the page.

- The ECC of each sector is aligned at the end of the spare area.
- The non-ECC spare data may or may not be covered by the ECC of sector 0

Figure 9-34. NAND Page Mapping and ECC: Pooled Spare Schemes

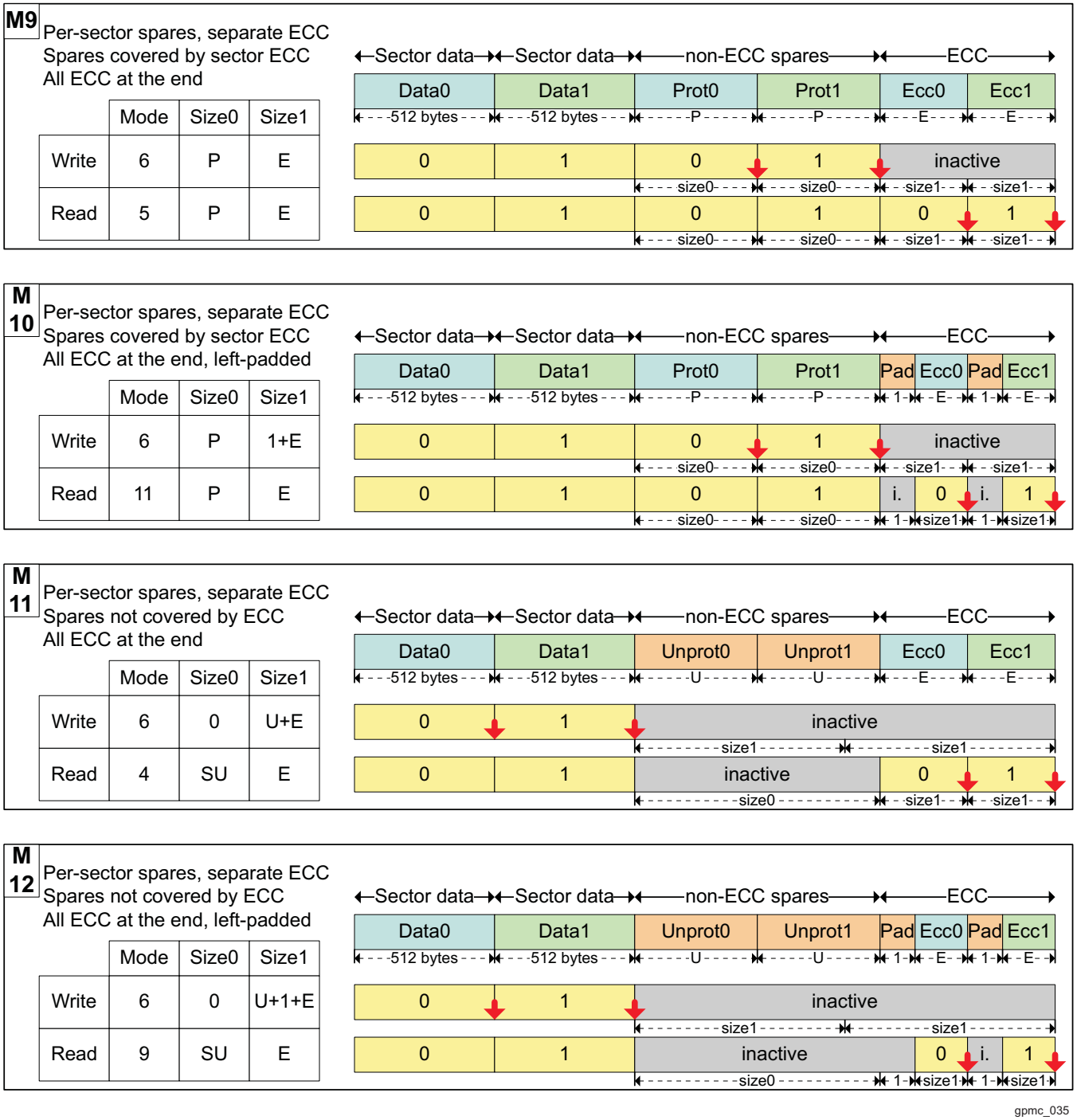


9.1.5.14.3.2.3.3 Per-Sector Spare Mapping, with ECC Separated at the End of the Page

In these schemes, each 512-byte sector of the main area is associated with 2 sections of the spare area.

- ECC section, all aligned at the end of the page
- other data section, aligned before the ECCs, each of which may or may not be protected by its sectors ECC

Figure 9-35. NAND Page Mapping and ECC: Per-Sector Schemes, with Separate ECC



9.1.5.14.4 Prefetch and Write-Posting Engine

NAND device data access cycles are usually much slower than the MCU system frequency; such NAND read or write accesses issued by the processor will impact the overall system performance, especially considering long read or write sequences required for NAND page loading or programming. To minimize this effect on system performance, the GPMC includes a prefetch and write-posting engine, which can be used to read from or write to any chip-select location in a buffered manner.

The prefetch and write-posting engine uses an embedded 64 bytes (32 Word16) FIFO to prefetch data from the NAND device in read mode (prefetch mode) or to store host data to be programmed into the NAND device in write mode (write-posting mode). The FIFO draining and filling (read and write) can be controlled either by the MCU through interrupt synchronization (an interrupt is triggered whenever a programmable threshold is reached) or the sDMA through DMA request synchronization, with a programmable request byte size in both prefetch or posting mode.

The prefetch and write-posting engine includes a single memory pool. Therefore, only one mode, read or write, can be used at any given time. In other words, the prefetch and write-posting engine is a single-context engine that can be allocated to only one chip-select at a time for a read prefetch or a write-posting process.

The engine does not support atomic command and address phase programming and is limited to linear memory read or write access. In consequence, it is limited to NAND data-stream access. The engine relies on the MCU NAND software driver to control block and page opening with the correct data address pointer initialization, before the engine can read from or write to the NAND memory device.

Once started, the engine data reads and writes sequencing is solely based on FIFO location availability and until the total programmed number of bytes is read or written.

Any host-concurrent accesses to a different chip-select are correctly interleaved with ongoing engine accesses. The engine has the lowest priority access so that host accesses to a different chip-select do not suffer a large latency.

A round-robin arbitration scheme can be enabled to ensure minimum bandwidth to the prefetch and write-posting engine in the case of back-to-back direct memory requests to a different chip-select. If the GPMC.GPMC_PREFETCH_CONFIG1[23] PFPWENROUNDROBIN bit is enabled, the arbitration grants the prefetch and write posting engine access to the GPMC bus for a number of requests programmed in the GPMC.GPMC_PREFETCH_CONFIG1[19:16] PFPWWEIGHTEDPRIO field.

The prefetch and write-posting engine is dedicated to data-stream access (as opposed to random data access). The engine does not include an address generator, and the request is limited to chip-select target identification. The prefetch/write-posting engine read or write request is routed to the access engine with the chip-select destination ID. After the required arbitration phase, the access engine processes the request as a single access with the data access size equal to the device size specified in the corresponding chip-select configuration.

NOTE: The destination chip-select configuration must be set to the NAND protocol-compatible configuration for which address lines are not used (the address bus is not changed from its current value). Selecting a different chip-select configuration can produce undefined behavior.

9.1.5.14.4.1 General Basic Programming Model

The engine can be configured only if the GPMC.GPMC_PREFETCH_CONTROL[0] STARTENGINE bit is de-asserted.

The engine must be correctly configured in prefetch or write-posting mode and must be linked to a NAND chip-select before it can be started. The chip-select is linked using the GPMC.GPMC_PREFETCH_CONFIG1[26:24] ENGINECSSELECTOR field.

In both prefetch and write-posting modes, the engine respectively uses byte or Word16 access requests for an 8- or 16-bit wide NAND device attached to the linked chip-select. The FIFOTHRESHOLD and TRANSFERCOUNT fields must be programmed accordingly as a number of bytes or a number of Word16.

When the GPMC.GPMC_PREFETCH_CONFIG1[7] ENABLEENGINE bit is set, the FIFO entry on the L3 interconnect port side is accessible at any address in the associated chip-select memory region. When the ENABLEENGINE bit is set, any host access to this chip-select is rerouted to the FIFO input. Directly accessing the NAND device linked to this chip-select from the host is still possible through the GPMC.GPMC_NAND_COMMAND_i, GPMC.GPMC_NAND_ADDRESS_i, and GPMC.GPMC_NAND_DATA_i registers (where I = 0 to 7).

The FIFO entry on the L3 interconnect port can be accessed with Byte, Word16, or Word32 access size, according to little-endian format, even though the FIFO input is 32-bit wide.

The FIFO control is made easier through the use of interrupts or DMA requests associated with the FIFOTHRESHOLD bit field. The GPMC.GPMC_PREFETCH_STATUS[30:24] FIFOPOINTER field monitors the number of available bytes to be read in prefetch mode or the number of free empty slots which can be written in write-posting mode. The GPMC.GPMC_PREFETCH_STATUS[13:0] COUNTVALUE field monitors the number of remaining bytes to be read or written by the engine according to the TRANSFERCOUNT value. The FIFOPOINTER and COUNTVALUE bit fields are always expressed as a number of bytes even if a 16-bit wide NAND device is attached to the linked chip-select.

In prefetch mode, when the FIFOPOINTER equals 0, that is, the FIFO is empty, a host read access receives the byte last read from the FIFO as its response. In case of Word32 or Word16 read accesses, the last byte read from the FIFO is copied the required number of times to fit the requested word size. In write-posting mode, when the FIFOPOINTER equals 0, that is, the FIFO is full, a host write overwrites the last FIFO byte location. There is no underflow or overflow error reporting in the GPMC.

9.1.5.14.4.2 Prefetch Mode

The prefetch mode is selected when the GPMC.GPMC_PREFETCH_CONFIG1[0] ACCESSMODE bit is cleared.

The MCU NAND software driver must issue the block and page opening (READ) command with the correct data address pointer initialization before the engine can be started to read from the NAND memory device. The engine is started by asserting the GPMC.GPMC_PREFETCH_CONTROL[0] STARTENGINE bit. The STARTENGINE bit automatically clears when the prefetch process completes.

If required, the ECC calculator engine must be initialized (configured, reset, and enabled) before the prefetch engine is started, so that the ECC is correctly computed on all data read by the prefetch engine.

When the GPMC.GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE bit is cleared, the prefetch engine starts requesting data as soon as the STARTENGINE bit is set. If using this configuration, the host must monitor the NAND device-ready pin so that it only sets the STARTENGINE bit when the NAND device is in a ready state, meaning data is valid for prefetching.

When the GPMC.GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE bit is set, the prefetch engine starts requesting data when an active to inactive wait signal transition is detected. The transition detector must be cleared before any transition detection; see Section 9.1.5.14.2.2. The GPMC.GPMC_PREFETCH_CONFIG1[5:4] WAITPINSELECTOR field selects which gpmc_wait pin edge detector triggers the prefetch engine in this synchronized mode.

If the STARTENGINE bit is set after the NAND address phase (page opening command), the engine is effectively started only after the actual NAND address phase completion. To prevent GPMC stall during this NAND address phase, set the STARTENGINE bit field before NAND address phase completion when in synchronized mode. The prefetch engine will start when an active to inactive wait signal transition is detected. The STARTENGINE bit is automatically cleared on prefetch process completion.

The prefetch engine issues a read request to ensure that the FIFO is always filled with as much data as acceptable, until the programmed GPMC.GPMC_PREFETCH_CONFIG2[13:0] TRANSFERCOUNT field is completed.

Table 9-15. Prefetch Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	0	Prefetch engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1[26:24]	0 to 7	Selects the chip-select associated with a NAND device where the prefetch engine is active.
ACCESSMODE	GPMC_PREFETCH_CONFIG1[0]	0	Selects prefetch mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1[14:8]		Selects the maximum number of bytes read or written by the host on DMA or interrupt request

Table 9-15. Prefetch Mode Configuration (continued)

Bit Field	Register	Value	Comments
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2[13:0]		Selects the number of bytes to be read or written by the engine to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1[3]	0/1	Selects when the engine starts the access to the chip-select
WAITPINSELECT	GPMC_PREFETCH_CONFIG1[17:16]	0 to 3	(If SynchroMode = 1) Selects wait pin edge detector
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1[27]	0/1	See Section 9.1.5.14.4.6.
CYCLEOPTIMIZATION	GPMC_PREFETCH_CONFIG1[30:28]		
ENABLEENGINE	GPMC_PREFETCH_CONFIG1[7]	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	1	Starts the prefetch engine

9.1.5.14.4.3 FIFO Control in Prefetch Mode

The FIFO can be drained directly by the MPU or by an sDMA channel.

In MPU draining mode, the FIFO status can be monitored through the GPMC.GPMC_PREFETCH_STATUS[30:24] FIFOPOINTER field or through the GPMC.GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bit. The FIFOPOINTER indicates the current number of available data to be read; FIFOTHRESHOLDSTATUS set to 1 indicates that at least FIFOTHRESHOLD bytes are available from the FIFO.

An interrupt can be triggered by the GPMC if the GPMC.GPMC_IRQENABLE[0] FIFOEVENTENABLE bit is set. The FIFO interrupt event is logged, and the GPMC.GPMC_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, the MPU must read all the available bytes, or at least enough bytes to get below the programmed FIFO threshold, and the FIFOEVENTSTATUS bit must be cleared to enable further interrupt events. The FIFOEVENTSTATUS bit must always be reset prior to asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt generation must be enabled after enabling the STARTENGINE bit.

Prefetch completion can be monitored through the GPMC.GPMC_PREFETCH_STATUS[13:0] COUNTVALUE field. COUNTVALUE indicates the number of currently remaining data to be requested according to the TRANSFERCOUNT value. An interrupt can be triggered by the GPMC when the prefetch process is complete (that is, COUNTVALUE equals 0) if the GPMC.GPMC_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. At prefetch completion, the TERMINALCOUNT interrupt event is also logged, and the GPMC.GPMC_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the MPU must clear the TERMINALCOUNTSTATUS bit. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

NOTE: The COUNTVALUE value is only valid when the prefetch engine is active (started), and an interrupt is only triggered when COUNTVALUE reaches 0, that is, when the prefetch engine automatically goes from an active to an inactive state.

The number of bytes to be prefetched (programmed in TRANSFERCOUNT) must be a multiple of the programmed FIFOTHRESHOLD to trigger the correct number of interrupts allowing a deterministic and transparent FIFO control. If this guideline is respected, the number of ISR accesses is always required and the FIFO is always empty after the last interrupt is triggered. In other cases, the TERMINALCOUNT interrupt must be used to read the remaining bytes in the FIFO (the number of remaining bytes being lower than the FIFOTHRESHOLD value).

In DMA draining mode, the GPMC.GPMC_PREFETCH_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes are ready to be read from the FIFO. The DMA channel owning this DMA request must be programmed so that the number of bytes programmed in FIFOTHRESHOLD is read from the FIFO during the DMA request process. The DMA request is kept active until this number of bytes has effectively been read from the FIFO, and no other DMA request can be issued until the ongoing active request is complete.

In prefetch mode, the TERMINALCOUNT event is also a source of DMA requests if the number of bytes to be prefetched is not a multiple of FIFOTHRESHOLD, the remaining bytes in the FIFO can be read by the DMA channel using the last DMA request. This assumes that the number of remaining bytes to be read is known and controlled through the DMA channel programming model.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive to active prefetch (the STARTENGINE bit is set to 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit so that the out-of-date active DMA request does not trigger spurious DMA transfers.

9.1.5.14.4.4 Write-Posting Mode

The write-posting mode is selected when the GPMC.GPMC_PREFETCH_CONFIG1[0] ACCESSMODE bit is set.

The MCU NAND software driver must issue the correct address pointer initialization command (page program) before the engine can start writing data into the NAND memory device. The engine starts when the GPMC.GPMC_PREFETCH_CONTROL[0] STARTENGINE bit is set to 1. The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the MCU NAND software driver must issue the second cycle program command and monitor the status for programming process completion (adding ECC handling, if required).

If used, the ECC calculator engine must be started (configured, reset, and enabled) before the posting engine is started so that the ECC parities are properly calculated on all data written by the prefetch engine to the associated chip-select.

In write-posting mode, the GPMC.GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE bit must be cleared so that posting starts as soon as the STARTENGINE bit is set and the FIFO is not empty.

If the STARTENGINE bit is set after the NAND address phase (page program command), the STARTENGINE setting is effective only after the actual NAND command completion. To prevent GPMC stall during this NAND command phase, set the STARTENGINE bit field before the NAND address completion and ensure that the associated DMA channel is enabled after the NAND address phase.

The posting engine issues a write request when valid data are available from the FIFO and until the programmed GPMC.GPMC_PREFETCH_CONFIG2[13:0] TRANSFERCOUNT accesses have been completed.

The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the MCU NAND software driver must issue the second cycle program command and monitor the status for programming process completion. The closing program command phase must only be issued when the full NAND page has been written into the NAND flash write buffer, including the spare area data and the ECC parities, if used.

Table 9-16. Write-Posting Mode Configuration

Bit Field	Register	Value	Comments
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	0	Write-posting engine can be configured only if STARTENGINE is set to 0.
ENGINECSSELECTOR	GPMC_PREFETCH_CONFIG1[26:24]	0 to 7	Selects the chip-select associated with a NAND device where the prefetch engine is active
ACCESSMODE	GPMC_PREFETCH_CONFIG1[0]	1	Selects write-posting mode
FIFOTHRESHOLD	GPMC_PREFETCH_CONFIG1[14:8]		Selects the maximum number of bytes read or written by the host on DMA or interrupt request
TRANSFERCOUNT	GPMC_PREFETCH_CONFIG2[13:0]		Selects the number of bytes to be read or written by the engine from/to the selected chip-select
SYNCHROMODE	GPMC_PREFETCH_CONFIG1[3]	0	Engine starts the access to chip-select as soon as STARTENGINE is set.
ENABLEOPTIMIZEDACCESS	GPMC_PREFETCH_CONFIG1[27]	0/1	See Section 9.1.5.14.4.6.
CYCLEOPTIMIZATION	GPMC_PREFETCH_CONFIG1[30:28]		

Table 9-16. Write-Posting Mode Configuration (continued)

Bit Field	Register	Value	Comments
ENABLEENGINE	GPMC_PREFETCH_CONFIG1[7]	1	Engine enabled
STARTENGINE	GPMC_PREFETCH_CONTROL[0]	1	Starts the prefetch engine

9.1.5.14.4.5 FIFO Control in Write-Posting Mode

The FIFO can be filled directly by the MPU or by an sDMA channel.

In MPU filling mode, the FIFO status can be monitored through the FIFOPOINTER or through the GPMC.GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bit. FIFOPOINTER indicates the current number of available free byte places in the FIFO, and the FIFOTHRESHOLDSTATUS bit, when set, indicates that at least FIFOTHRESHOLD free byte places are available in the FIFO.

An interrupt can be issued by the GPMC if the GPMC.GPMC_IRQENABLE[0] FIFOEVENTENABLE bit is set. When the interrupt is fired, the GPMC.GPMC_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, the MPU must write enough bytes to fill the FIFO, or enough bytes to get below the programmed threshold, and the FIFOEVENTSTATUS bit must be cleared to get further interrupt events. The FIFOEVENTSTATUS bit must always be cleared prior to asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt must be enabled after enabling the STARTENGINE bit.

The posting completion can be monitored through the GPMC.GPMC_PREFETCH_STATUS[13:0] COUNTVALUE field. COUNTVALUE indicates the current number of remaining data to be written based on the TRANSFERCOUNT value. An interrupt is issued by the GPMC when the write-posting process completes (that is, COUNTVALUE equal to 0) if the GPMC.GPMC_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. When the interrupt is fired, the GPMC.GPMC_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the MPU must clear the TERMINALCOUNTSTATUS bit. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

NOTE: The COUNTVALUE value is only valid if the write-posting engine is active and started, and an interrupt is only issued when COUNTVALUE reaches 0, that is, when the posting engine automatically goes from active to inactive.

In DMA filling mode, the DMAMode bit field in the GPMC.GPMC_PREFETCH_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes-free places are available in the FIFO. The DMA channel owning this DMA request must be programmed so that a number of bytes equal to the value programmed in the FIFOTHRESHOLD bit field are written into the FIFO during the DMA access. The DMA request remains active until the associated number of bytes has effectively been written into the FIFO, and no other DMA request can be issued until the ongoing active request has been completed.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive to active prefetch (STARTENGINE set to 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit so that an out-of-date active DMA request does not trigger spurious DMA transfers.

In write-posting mode, the DMA or the MPU fill the FIFO with no consideration to the associated byte enables. Any byte stored in the FIFO is written into the memory device.

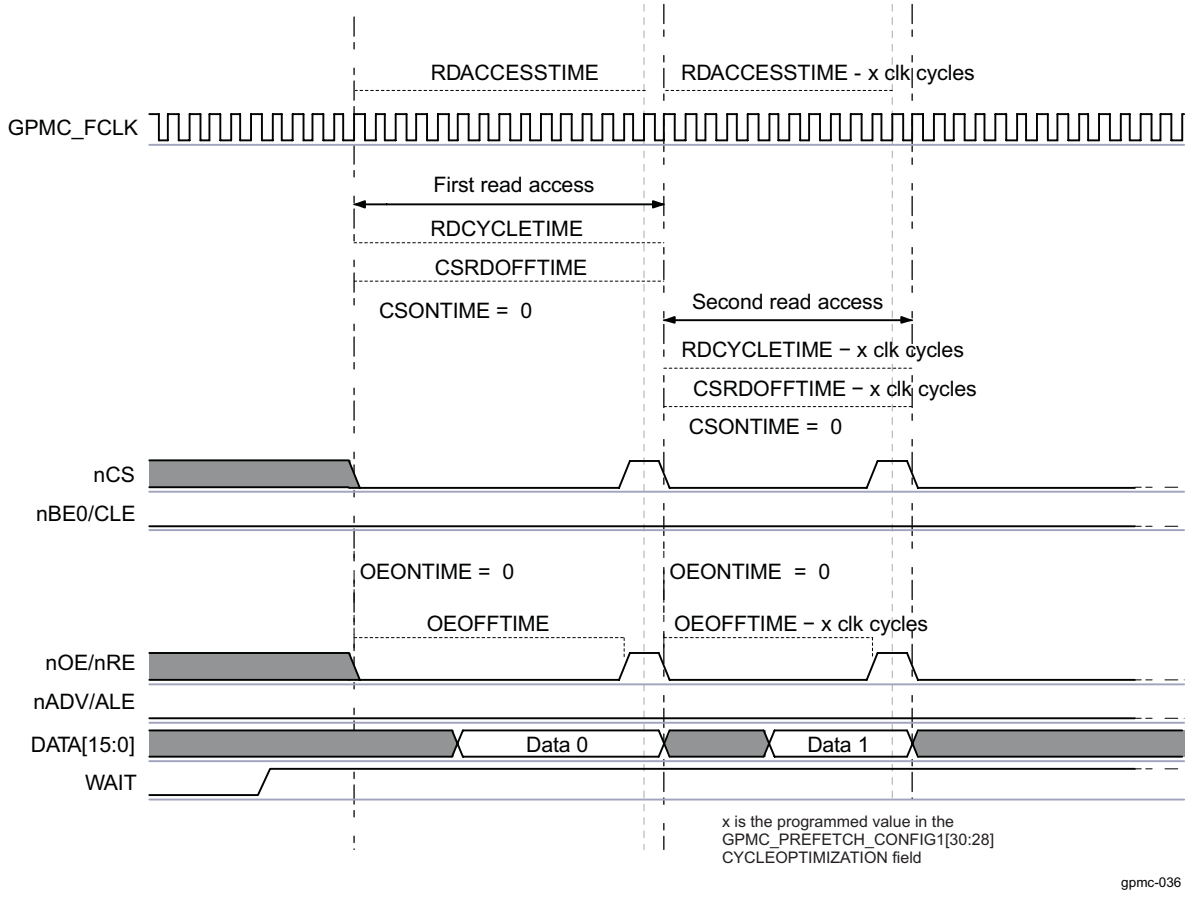
9.1.5.14.4.6 Optimizing NAND Access Using the Prefetch and Write-Posting Engine

Access time to a NAND memory device can be optimized for back-to-back accesses if the associated nCS signal is not deasserted between accesses. The GPMC access engine can track prefetch engine accesses to optimize the access timing parameter programmed for the allocated chip-select, if no accesses to other chip-selects (that is, interleaved accesses) occur. Similarly, the access engine also eliminates the CYCLE2CYCLEDELAY even if CYCLE2CYCLESAMECSEN is set. This capability is limited to the prefetch and write-posting engine accesses, and MPU accesses to a NAND memory device (through the defined chip-select memory region or through the GPMC.GPMC_NAND_DATA_i location, I = 0 to 7) are never optimized.

The GPMC.GPMC_PREFETCH_CONFIG1[27] ENABLEOPTIMIZEDACCESS bit must be set to enable optimized accesses. To optimize access time, the GPMC.GPMC_PREFETCH_CONFIG1[30:28] CYCLEOPTIMIZATION field defines the number of GPMC_FCLK cycles to be suppressed from the RDCYCLETIME, WRCYCLETIME, RDACCESSTIME, WRACCESSTIME, CSOFFTIME, ADVOFFTIME, OEOFFTIME, and WEOFFTIME timing parameters.

Figure 9-36 highlights that, in the case of back-to-back accesses to the NAND flash through the prefetch engine, CYCLE2CYCLESAMECSEN is forced to 0 when using optimized accesses. The first access uses the regular timing settings for this chip-select. All accesses after this one use settings reduced by x clock cycles, x being defined by the GPMC_PREFETCH_CONFIG1[30:28] CYCLEOPTIMIZATION field.

Figure 9-36. NAND Read Cycle Optimization Timing Description



9.1.5.14.4.7 Interleaved Accesses between Prefetch and Write-Posting Engine and Other Chip-Selects

Any on-going read or write access from the prefetch and write-posting engine is completed before an access to any other chip-select can be initiated. As a default, the arbiter uses a fixed-priority algorithm, and the prefetch and write-posting engine has the lowest priority. The maximum latency added to access starting time in this case equals the RDCYCLETIME or WRCYCLETIME (optimized or not) plus the requested BUSTURNAROUND delay for bus turnaround completion programmed for the chip-select to which the NAND device is connected to.

Alternatively, a round-robin arbitration can be used to prioritize accesses to the external bus. This arbitration scheme is enabled by setting the GPMC.GPMC_PREFETCH_CONFIG1[23] PFPWENROUNDROBIN bit. When a request to another chip-select is received while the prefetch and write-posting engine is active, priority is given to the new request. The request processed thereafter is the prefetch and write-posting engine request, even if another interconnect request is passed in the mean time. The engine keeps control of the bus for an additional number of requests programmed in the GPMC.GPMC_PREFETCH_CONFIG1[19:16] PFPWWEIGHTEDPRIO bit field. Control is then passed to the direct interconnect request.

As an example, the round-robin arbitration scheme is selected with PFPWWEIGHTEDPRIO set to 0x2. Considering the prefetch and write-posting engine and the interconnect interface are always requesting access to the external interface, the GPMC grants priority to the direct interconnect access for one request. The GPMC then grants priority to the engine for three requests, and finally back to the direct interconnect access, until the arbiter is reset when one of the two initiators stops initiating requests.

9.1.6 GPMC Use Cases and Tips

9.1.6.1 How to Set GPMC Timing Parameters for Typical Accesses

9.1.6.1.1 External Memory Attached to the GPMC Module

As discussed in the introduction to this chapter, the GPMC module supports the following external memory types:

- Asynchronous or synchronous, 8-bit or 16-bit-width memory or device
- 16-bit address/data-multiplexed or not multiplexed NOR flash device
- 8- or 16-bit NAND flash device

The following examples show how to calculate GPMC timing parameters by showing a typical parameter setup for the access to be performed.

The example is based on a 512-Mb multiplexed NOR flash memory with the following characteristics:

- Manufacturer: NOR Flash
- Type: NOR flash (address/data-multiplexed mode)
- Size: 512M bits
- Data Bus: 16 bits wide
- Speed: 104 MHz clock frequency
- Read access time: 80 ns

9.1.6.1.2 Typical GPMC Setup

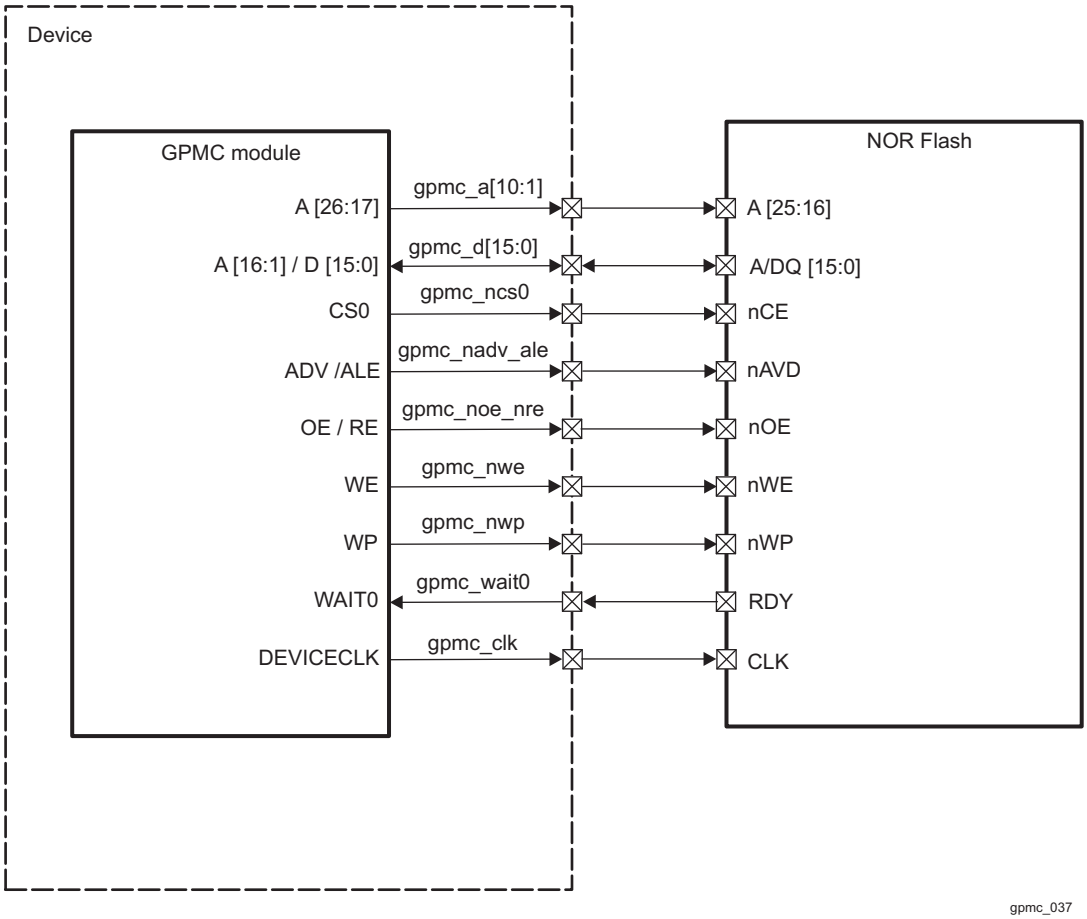
Table 9-17 lists some of the I/Os of the GPMC module.

Table 9-17. GPMC Signals

Signal Name	I/0	Description
GPMC_FCLK	Internal	Functional and interface clock. Acts as the time reference.
gpmc_clk	I/O	External clock provided to the external device for synchronous operations
gpmc_a[10: 1]	O	Address
gpmc_d[15: 0]	I/O	Data-multiplexed with addresses A[16:1] on memory side
gpmc_ncs	O	Chip-select
gpmc_nadv_ale	O	Address valid enable
gpmc_noe_nre	O	Output enable (read access only)
gpmc_nwe	O	Write enable (write access only)
gpmc_wait[3:0]	I	Ready signal from memory device. Indicates when valid burst data is ready to be read

Figure 9-37 shows the typical connection between the GPMC module and the attached NOR Flash memory.

Figure 9-37. GPMC Connection to External NOR Flash Memory



The following sections demonstrate how to calculate GPMC parameters for three access types:

- Synchronous burst read
- Asynchronous read
- Asynchronous single write

9.1.6.1.2.1 GPMC Configuration for Synchronous Burst Read Access

The clock runs at 104 MHz (f = 104 MHz; T = 9, 615 ns).

Table 9-18 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 9-19 shows how to calculate timings for the GPMC using the memory parameters.

Figure 9-38 shows the synchronous burst read access.

Table 9-18. Useful Timing Parameters on the Memory Side

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCES	nCS setup time to clock	0
tACS	Address setup time to clock	3
tIACC	Synchronous access time	80

Table 9-18. Useful Timing Parameters on the Memory Side (continued)

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tBACC	Burst access time valid clock to output delay	5,2
tCEZ	Chip-select to High-Z	7
tOEZ	Output enable to High-Z	7
tAVC	nADV setup time	6
tAVD	nAVD pulse	6
tACH	Address hold time from clock	3

The following terms, which describe the timing interface between the controller and its attached device, are used to calculate the timing parameters on the GPMC side:

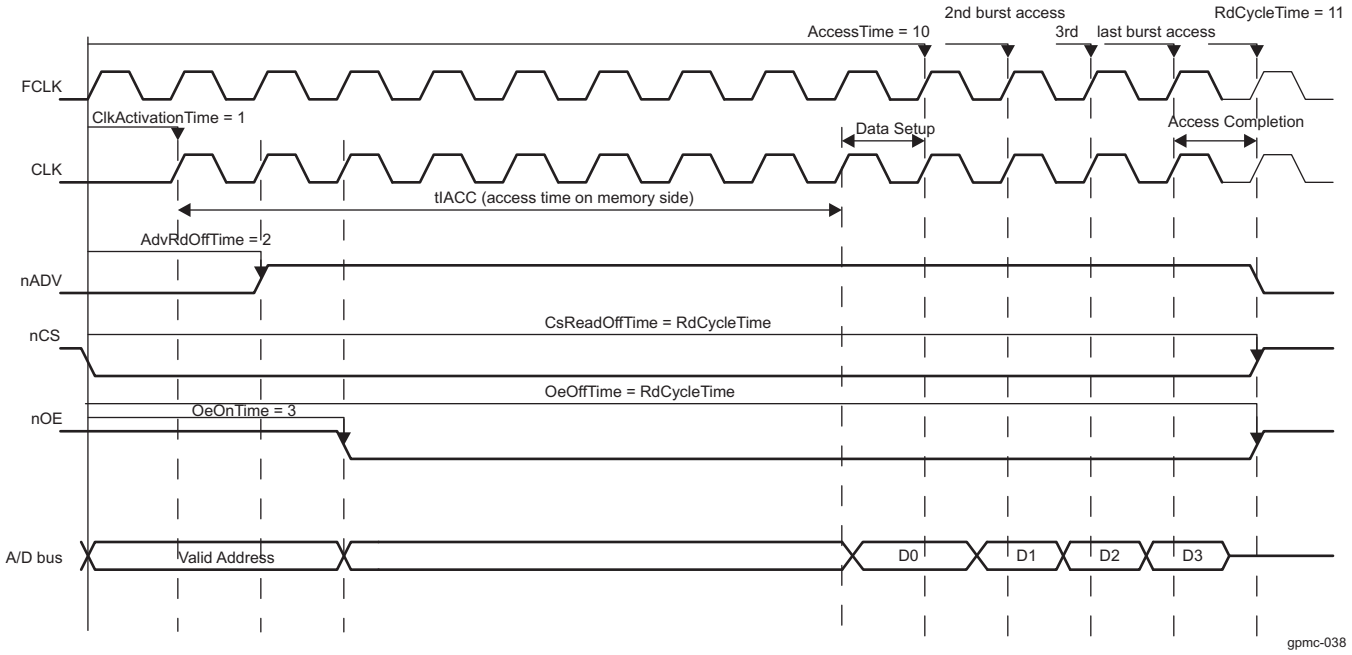
- Read Access time (GPMC side): Time required to activate the clock + read access time requested on the memory side + data setup time required for optimal capture of a burst of data
- Data setup time (GPMC side): Ensures a good capture of a burst of data (as opposed to taking a burst of data out). One burst of data is processed in one clock cycle (T = 9,615 ns). The read access time between 2 bursts of data is tBACC = 5,2 ns. Therefore, data setup time is a clock period - tBACC = 4,415 ns of data setup.
- Access completion (GPMC side): (Different from page burst access time) Time required between the last burst access and access completion: nCS/nOE hold time (nCS and nOE must be released at the end of an access. These signals are held to allow the access to complete).
- Read cycle time (GPMC side): Read Access time + access completion
- Write cycle time for burst access: Not supported for NOR flash memory

Table 9-19. Calculating GPMC Timing Parameters

Parameter Name on GPMC Side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
ClkActivation Time	min (tCES, tACS)	3	1	CLKACTIVATIONTIME = 0x1
RdAccessTime	roundmax (ClkActivationTime + tIACC + DataSetupTime)	94,03: (9,615 + 80 + 4,415)	10 : roundmax (94,03 / 9,615)	ACCESSTIME = 0x0A
PageBurst AccessTime	roundmax (tBACC)	roundmax (5,2)	1	PAGEBURSTACCESSTIME = 0x1
RdCycleTime	AccessTime + max (tCEZ, tOEZ)	101, 03: (94, 03 + 7)	11	RDCYCLETIME = 0x0B
CsOnTime	tCES	0	0	CSONTIME = 0x0
CsReadOffTime	RdCycleTime	-	11	CSRDOFFTIME = 0x0B
AdvOnTime	tAVC (1)	0	0	ADVONTIME = 0x0
AdvRdOffTime	tAVD + tAVC (2)	12	2	ADVRDOFFTIME = 0x02
OeOnTime (3)	(ClkActivationTime + tACH) < OeOnTime < (ClkActivationTime + tIACC)	-	3 for instance.	OEONTIME = 0x3
OeOffTime	RdCycleTime	-	11	OEOFFTIME = 0x0B

(1) The external clock provided to the NOR flash is not yet available.
(2) AdvRdOffTime - AdvOnTime = tAVD; thus, AdvRdOffTime = tAVD + AdvOnTime = tAVD + tAVC.
(3) OeOnTime must guarantee that addresses are available. It must not exceed the availability of the first burst of data.

Figure 9-38. Synchronous Burst Read Access (Timing Parameters in Clock Cycles)



9.1.6.1.2.2 GPMC Configuration for Asynchronous Read Access

The clock runs at 104 MHz (f = 104 MHz; T = 9, 615 ns).

Table 9-20 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Table 9-21 shows how to calculate timings for the GPMC using the memory parameters.

Figure 9-39 shows the asynchronous read access.

Table 9-20. AC Characteristics for Asynchronous Read Access

AC Read Characteristics on the Memory Side	Description	Duration (ns)
tCE	Read Access time from nCS low	80
tAAVDS	Address setup time to rising edge of nADV	3
tAVDP	nADV low time	6
tCAS	nCS setup time to nADV	0
tOE	Output enable to output valid	6
tOEZ	Output enable to High-Z	7

Use the following formula to calculate the RdCycleTime parameter for this typical access:

RdCycleTime = RdAccessTime + AccessCompletion = RdAccessTime + 1 clock cycle + tOEZ:

- First, on the memory side, the external memory makes the data available to the output bus. This is the memory-side read access time defined in Table 9-20: the number of clock cycles between the address capture (nADV rising edge) and the data valid on the output bus.

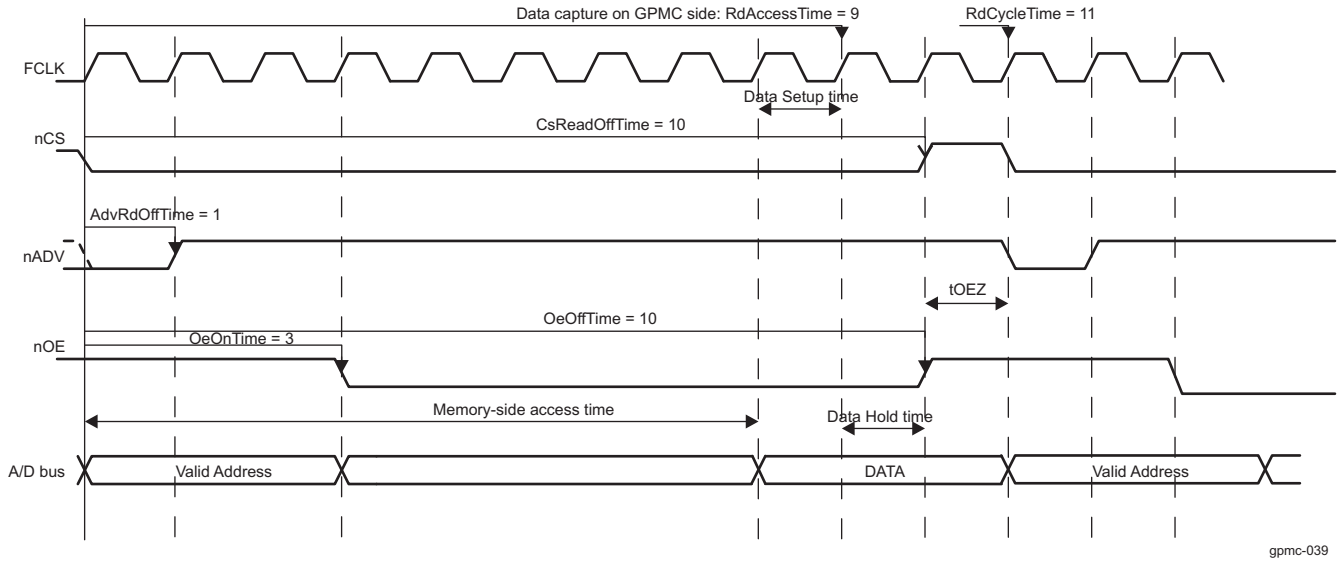
  The GPMC side requires some hold to allow the data to be captured correctly and the access to be finished.
- To read the data correctly, the GPMC must capture it with enough data setup time; the GPMC module captures the data on the next rising edge. This is access time on the GPMC side.
- There must also be a data hold time for correctly reading the data (checking that there is no nOE/nCS deassertion while reading the data). This data hold time is 1 clock cycle (AccessTime + 1).

- To complete the access, nOE/nCS signals are driven to High-Z. AccessTime + 1 + tOEZ is the read cycle time.
- Addresses can now be relatched and a new read cycle begun.

Table 9-21. GPMC Timing Parameters for Asynchronous Read Access

Parameter Name on GPMC side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Register Configurations
ClkActivationTime	n/a (asynchronous mode)			
AccessTime	round max (tCE)	80	9	ACCESSTIME = 0x09
PageBurstAccess Time	n/a (single access)			
RdCycleTime	AccessTime + 1cycle + tOEZ	96, 615	11	RDCYCLETIME = 0x0B
CsOnTime	tCAS	0	0	CSONTIME = 0x0
CsReadOffTime	AccessTime + 1 cycle	89, 615	10	CSRDOFFTIME = 0x0A
AdvOnTime	tAAVDS	3	1	ADVONTIME = 0X1
AdvRdOffTime	tAAVDS + tAVDP	9	1	ADVRDOFFTIME = 0x01
OeOnTime	OeOnTime >= AdvRdOffTime (multiplexed mode)	-	3 for instance	OEONTIME = 0x3
OeOffTime	AccessTime + 1cycle	89, 615	10	OEOFFTIME = 0x0A

Figure 9-39. Asynchronous Single Read Access (Timing Parameters in Clock Cycles)



9.1.6.1.2.3 GPMC Configuration for Asynchronous Single Write Access

The clock runs at 104 MHz: (f = 104 MHz; T = 9, 615 ns).

Table 9-23 shows how to calculate timings for the GPMC using the memory parameters.

Table 9-22 shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

Figure 9-40 shows the synchronous burst write access.

Table 9-22. AC Characteristics for Asynchronous Single Write (Memory Side)

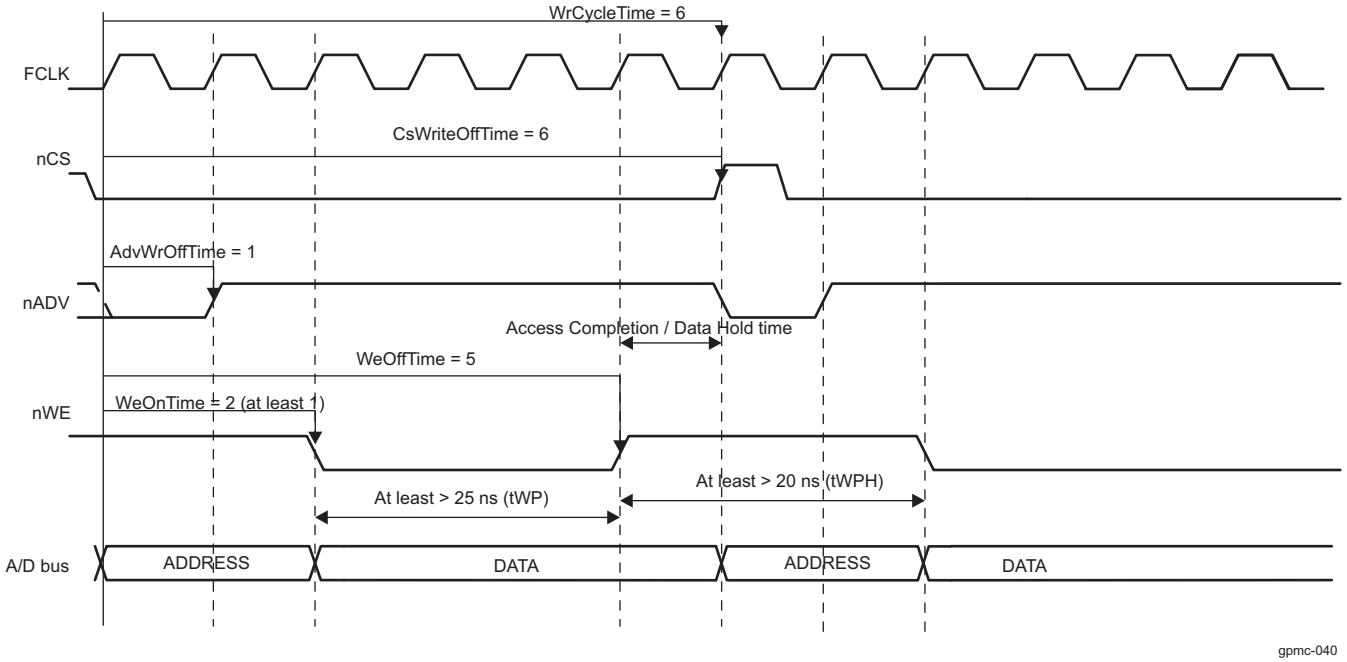
AC Characteristics on the Memory Side	Description	Duration (ns)
tWC	Write cycle time	60
tAVDP	nADV low time	6
tWP	Write pulse width	25
tWPH	Write pulse width high	20
tCS	nCS setup time to nWE	3
tCAS	nCS setup time to nADV	0
tAVSC	nADV setup time	3

For asynchronous single write access, write cycle time is WrCycleTime = WeOffTime + AccessCompletion = WeOffTime + 1. For the AccessCompletion: 1 cycle is required for data hold time (nCS deassertion).

Table 9-23. GPMC Timing Parameters for Asynchronous Single Write

Parameter Name on GPMC side	Formula	Duration (ns)	Number of Clock Cycles (F = 104 MHz)	GPMC Registers Configuration
ClkActivationTime	n/a (asynchronous mode)			
AccessTime	Applicable only to WAITMONITORING (the value is the same as for read access)			
PageBurstAccessTime	n/a (single access)			
WrCycleTime	WeOffTime + AccessCompletion	57, 615	6	WRCYCLETIME = 0x06
CsOnTime	tCAS	0	0	CSONTIME = 0x0
CsWrOffTime	WeOffTime + 1	57, 615	6	CSWROFFTIME = 0x06
AdvOnTime	tAVSC	3	1	ADVONTIME = 0x1
AdvWrOffTime	tAVSC + tAVDP	9	1	ADVWROFFTIME = 0x01
WeOnTime	tCS	3	1	WEONTIME = 0x1
WeOffTime	tCS + tWP + tWPH	48	5	WEOFFTIME = 0x05

Figure 9-40. Asynchronous Single Write Access (Timing Parameters in Clock Cycles)



9.1.6.2 How to Choose a Suitable Memory to use with the GPMC

This section is intended to help the user select a suitable memory device to interface with the GPMC controller.

9.1.6.2.1 Supported Memories or Devices

NAND flash and NOR flash architectures are the two flash technologies. The GPMC supports various types of external memory or device, basically any one that supports NAND or NOR protocols:

- 8- and 16-bit width asynchronous or synchronous memory or device (8-bit: non burst device only)
- 16-bit address and data multiplexed NOR flash devices (pSRAM, OneNAND™, ...)
- 8- and 16-bit NAND flash device

NOTE: Non-multiplexed NOR flash devices are supported by the GPMC but their usage is highly limited. As only ten address pins are available on the GPMC interface, the maximum device size supported is 2KB.

9.1.6.2.1.1 Memory Pin Multiplexing

This section highlights the interfacing differences of the GPMC supported memories.

Table 9-24. Supported Memories Interfaces

Function	16-bit Address/Data muxed pSRAM or NOR Flash [1]	OneNAND	16-bit NAND	8-bit NAND
gpmc_a10	A26			
gpmc_a9	A25			
gpmc_a8	A24			
gpmc_a7	A23			
gpmc_a6	A22			
gpmc_a5	A21			
gpmc_a4	A20			
gpmc_a3	A19			
gpmc_a2	A18			
gpmc_a1	A17			
gpmc_d15	D15 or A16		IO15	
gpmc_d14	D14 or A15		IO14	
gpmc_d13	D13 or A14		IO13	
gpmc_d12	D12 or A13		IO12	
gpmc_d11	D11 or A12		IO11	
gpmc_d10	D10 or A11		IO10	
gpmc_d9	D9 or A10		IO9	
gpmc_d8	D8 or A9		IO8	
gpmc_d7	D7 or A8		IO7	
gpmc_d6	D6 or A7		IO6	
gpmc_d5	D5 or A6		IO5	
gpmc_d4	D4 or A5		IO4	
gpmc_d3	D3 or A4		IO3	
gpmc_d2	D2 or A3		IO2	
gpmc_d1	D1 or A2		IO1	
gpmc_d0	D0 or A1		IO0	
gpmc_clk	CLK			

[1] Addresses seen from the device side. When interfacing to the external IC, A1 is connected to the memory A0, A2 to the memory A1, and so on...

Table 9-24. Supported Memories Interfaces (continued)

Function	16-bit Address/Data muxed pSRAM or NOR Flash [1]	OneNAND	16-bit NAND	8-bit NAND
gpmc_ncs0	nCS0 (Chip Select)		nCE0 (Chip Enable)	
gpmc_ncs1	nCS1		nCE1	
gpmc_ncs2	nCS2		nCE2	
GPMC_ncs3	nCS3		nCE3	
GPMC_ncs4	nCS4		nCE4	
GPMC_ncs5	nCS5		nCE5	
GPMC_ncs6	nCS6		nCE6	
GPMC_ncs7	nCS7		nCE7	
gpmc_nadv_ale	nADV (Address Valid)		ALE (Address Latch Enable)	
gpmc_noe	nOE (Output Enable)		nRE (Read Enable)	
gpmc_nwe	nWE (Write Enable)		nWE (Write Enable)	
gpmc_nbe0_cle	nBE0 (Byte Enable)		CLE (Command Latch Enable)	
gpmc_nbe1	nBE1			
gpmc_nwp	nWP (Write Protect)		nWP (Write Protect)	
gpmc_wait0	WAIT0		R/nB0 (Ready/Busy)	
gpmc_wait1	WAIT1		R/nB1	
gpmc_wait2	WAIT2		R/nB2	
gpmc_wait3	WAIT3		R/nB3	

9.1.6.2.1.2 NAND Interface Protocol

NAND flash architecture, introduced in 1989, is a flash technology. NAND is a page-oriented memory device, i.e. read and write accesses are done by pages. NAND achieves great density by sharing common areas of the storage transistor, which creates strings of serially connected transistors (in NOR devices, each transistor stands alone). Thanks to its high density NAND is best suited to devices requiring high capacity data storage, such as pictures, music, or data files. NAND non-volatility, makes of it a good storage solution for many applications where mobility, low power, and speed are key factors. Low pin count and simple interface are other advantages of NAND.

Table 9-25 summarizes the NAND interface signals level applied to external device or memories.

Table 9-25. NAND Interface Bus Operations Summary

Bus operation	CLE	ALE	nCE	nWE [1]	nRE [1]	nWP
Read (cmd input)	H	L	L	RE	H	x
Read (add input)	L	H	L	RE	H	x
Write (cmd input)	H	L	L	RE	H	H
Write (add input)	L	H	L	RE	H	H
Data input	L	L	L	RE	H	H
Data output	L	L	L	H	FE	x
Busy (during read)	x	x	H [2]	H [2]	H [2]	x
Busy (during program)	x	x	x	x	x	H
Busy (during erase)	x	x	x	x	x	H
Write protect	x	x	x	x	x	L
Stand-by	x	x	H	x	x	H/L [3]

(1) RE stands for rising edge, FE stands for falling edge
(2) Can be either nCE high, or WE and nRE high.
(3) nWP should be biased to CMOS high or CMOS low for standby

9.1.6.2.1.3 NOR Interface Protocol

NOR flash architecture, introduced in 1988, is a flash technology. Unlike NAND which is a sequential access device, NOR is directly addressable, i.e. is designed to be a random access device. NOR is best suited to devices used to store and run code or firmware, usually in small capacities. While NOR has fast read capabilities it has slow write and erase functions compared to NAND architecture.

Table 9-26 summarizes the NOR interface signals level applied to external device or memories.

Table 9-26. NOR Interface Bus Operations Summary

Bus operation	CLK	nADV	nCS	nOE	nWE	WAIT	DQ[15:0]
Read (asynchronous)	x	L	L	L	H	Asserted	Output
Read (synchronous)	Running	L	L	L	H	Driven	Output
Read (burst suspend)	Halted	x	L	H	H	Active	Output
Write	x	L	L	H	L	Asserted	Input
Output disable	x	x	L	H	H	Asserted	High-Z
Standby	x	x	H	x	x	High-Z	High-Z

9.1.6.2.1.4 Other Technologies

Other supported device type interact with the GPMC through the NOR interface protocol.

OneNAND™ is a high density and low-power memory device. OneNAND™ is based on single- or multi-level-cell NAND core with SRAM and logic, and interfaces as a synchronous NOR Flash, plus has synchronous write capability. It reads faster than conventional NAND and write faster than conventional NOR flash. Hence, it is appropriate for both mass storage and code storage.

pSRAM stands for pseudo-static random access memory. pSRAM is a low-power memory device for mobile applications. pSRAM is based on the DRAM cell with internal refresh and address control features, and interfaces as a synchronous NOR Flash, plus has synchronous write capability.

9.1.6.2.2 GPMC Features and Settings

This section lists GPMC features and settings:

- Supported device type: up to eight NAND or NOR protocol external memories or devices
- Operating Voltage: 1.8V;
- Maximum operating frequency provided externally: up to 100MHz (single device) with an L3-clock of 100MHz. Up to 83MHz (L3-clock divided by two) with an L3-clock of 166MHz. See the device-specific Data Manual for precise information.
- Maximum GPMC addressing capability: 1 GByte divided into eight chip-selects
- Maximum supported memory size: 128 MBytes (must be a power-of-2)
- Minimum supported memory size: 16 MBytes (must be a power-of-2). Aliasing occurs when addressing smaller memories.
- Data path to external memory or device: 8- and 16-bit wide
- Burst and page access: burst of 4-8-16 Word16
- Supports bus keeping
- Supports bus turn around

9.1.7 GPMC Registers

This section provides information about the GPMC instance in this product. Table 9-28 provides a summary of the GPMC registers. The remaining parts of this section describe the registers within the module instance.

Table 9-27. Instance Summary

Module Name	Base Address	Size
GPMC	0x6E00 0000	16 Mbytes

9.1.7.1 GPMC Register Mapping Summary

Table 9-28. GPMC Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
GPMC_SYSCONFIG	RW	32	0x0000 0010	0x6E00 0010	Section 9.1.7.2.1
GPMC_SYSSTATUS	R	32	0x0000 0014	0x6E00 0014	Section 9.1.7.2.2
GPMC_IRQSTATUS	RW	32	0x0000 0018	0x6E00 0018	Section 9.1.7.2.3
GPMC_IRQENABLE	RW	32	0x0000 001C	0x6E00 001C	Section 9.1.7.2.4
GPMC_TIMEOUT_CONTROL	RW	32	0x0000 0040	0x6E00 0040	Section 9.1.7.2.5
GPMC_ERR_ADDRESS	RW	32	0x0000 0044	0x6E00 0044	Section 9.1.7.2.6
GPMC_ERR_TYPE	RW	32	0x0000 0048	0x6E00 0048	Section 9.1.7.2.7
GPMC_CONFIG	RW	32	0x0000 0050	0x6E00 0050	Section 9.1.7.2.8
GPMC_STATUS	RW	32	0x0000 0054	0x6E00 0054	Section 9.1.7.2.9
GPMC_CONFIG1_i (1)	RW	32	0x0000 0060 + (0x0000 0030 * I)	0x6E00 0060 + (0x0000 0030 * I)	Section 9.1.7.2.10
GPMC_CONFIG2_i (1)	RW	32	0x0000 0064 + (0x0000 0030 * I)	0x6E00 0064 + (0x0000 0030 * I)	Section 9.1.7.2.11
GPMC_CONFIG3_i (1)	RW	32	0x0000 0068 + (0x0000 0030 * I)	0x6E00 0068 + (0x0000 0030 * I)	Section 9.1.7.2.12
GPMC_CONFIG4_i (1)	RW	32	0x0000 006C + (0x0000 0030 * I)	0x6E00 006C + (0x0000 0030 * I)	Section 9.1.7.2.13
GPMC_CONFIG5_i (1)	RW	32	0x0000 0070 + (0x0000 0030 * I)	0x6E00 0070 + (0x0000 0030 * I)	Section 9.1.7.2.14
GPMC_CONFIG6_i (1)	RW	32	0x0000 0074 + (0x0000 0030 * I)	0x6E00 0074 + (0x0000 0030 * I)	Section 9.1.7.2.15
GPMC_CONFIG7_i (1)	RW	32	0x0000 0078 + (0x0000 0030 * I)	0x6E00 0078 + (0x0000 0030 * I)	Section 9.1.7.2.16
GPMC_NAND_COMMAND_i (1)	W	32	0x0000 007C + (0x0000 0030 * I)	0x6E00 007C + (0x0000 0030 * I)	Section 9.1.7.2.17
GPMC_NAND_ADDRESS_i (1)	W	32	0x0000 0080 + (0x0000 0030 * I)	0x6E00 0080 + (0x0000 0030 * I)	Section 9.1.7.2.18
GPMC_NAND_DATA_i (1)	RW	32	0x0000 0084 + (0x0000 0030 * I)	0x6E00 0084 + (0x0000 0030 * I)	Section 9.1.7.2.19
GPMC_PREFETCH_CONFIG1	RW	32	0x0000 01E0	0x6E00 01E0	Section 9.1.7.2.20
GPMC_PREFETCH_CONFIG2	RW	32	0x0000 01E4	0x6E00 01E4	Section 9.1.7.2.21
GPMC_PREFETCH_CONTROL	RW	32	0x0000 01EC	0x6E00 01EC	Section 9.1.7.2.22
GPMC_PREFETCH_STATUS	RW	32	0x0000 01F0	0x6E00 01F0	Section 9.1.7.2.23
GPMC_ECC_CONFIG	RW	32	0x0000 01F4	0x6E00 01F4	Section 9.1.7.2.24
GPMC_ECC_CONTROL	RW	32	0x0000 01F8	0x6E00 01F8	Section 9.1.7.2.25
GPMC_ECC_SIZE_CONFIG	RW	32	0x0000 01FC	0x6E00 01FC	Section 9.1.7.2.26
GPMC_ECCj_RESULT (2) where k = j - 1.	RW	32	0x0000 0200 + (0x0000 0004 * k) (3)	0x6E00 0200 + (0x0000 0004 * k) (3)	Section 9.1.7.2.27

(1) I = 0 to 7.
(2) j = 1 to 9.
(3) k = 0 to 8.

Table 9-28. GPMC Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
GPMC_BCH_RESULT0_i [(1)]	RW	32	0x0000 0240 + (0x0000 0010 * I)	0x6E00 0240 + (0x0000 0010 * I)	Section 9.1.7.2.28
GPMC_BCH_RESULT1_i [(1)]	RW	32	0x0000 0244 + (0x0000 0010 * I)	0x6E00 0244 + (0x0000 0010 * I)	Section 9.1.7.2.29
GPMC_BCH_RESULT2_i [(1)]	RW	32	0x0000 0248 + (0x0000 0010 * I)	0x6E00 0248 + (0x0000 0010 * I)	Section 9.1.7.2.30
GPMC_BCH_RESULT3_i [(1)]	RW	32	0x0000 024C + (0x0000 0010 * I)	0x6E00 024C + (0x0000 0010 * I)	Section 9.1.7.2.31
GPMC_BCH_SWDATA	RW	32	0x0000 02D0	0x6E00 02D0	Section 9.1.7.2.32

9.1.7.2 GPMC Register Descriptions

NOTE: All GPMC registers are aligned to 32-bit address boundaries. All register file accesses, except to GPMC_NAND_DATA_i register, are little endian. If the GPMC_NAND_DATA_i register location is accessed, the endianness is access-dependent.

9.1.7.2.1 GPMC_SYSCONFIG

Table 9-29. GPMC_SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x6E00 0010	**Instance**	GPMC
Description	This register controls the various parameters of the Interconnect.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											IDLEMODE		RESERVED	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Write 0s for future compatibility. Reads return 0s.	RW	0x0000000
4:3	IDLEMODE	0x0: Force-idle. An idle request is acknowledged unconditionally 0x1: No-idle. An idle request is never acknowledged 0x2: Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module 0x3: Do not use	RW	0x0
2	RESERVED	Write 0 for future compatibility Reads returns 0	RW	0x0
1	SOFTRESET	Software reset. Set this bit to 1 triggers a module reset. This bit is automatically reset by hardware. During reads, it always returns 0. 0x0: Normal mode 0x1: The module is reset	RW	0x0
0	AUTOIDLE	Internal Interface clock gating strategy 0x0: Interface clock is free-running 0x1: Automatic Interface clock gating strategy is applied, based on the Interconnect activity	RW	0x0

9.1.7.2.2 GPMC_SYSSTATUS

Table 9-30. GPMC_SYSSTATUS

Address Offset	0x0000 0014		
Physical Address	0x6E00 0014	**Instance**	GPMC
Description	This register provides status information about the module, excluding the interrupt status information		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RESERVED							RESETDONE

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reads returns 0	R	0x000000
7:1	RESERVED	Reads returns 0 (reserved for Interconnect-socket status information)	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset in ongoing 0x1: Reset completed	R	0x-

9.1.7.2.3 GPMC_IRQSTATUS

Table 9-31. GPMC_IRQSTATUS

Address Offset	0x0000 0018		
Physical Address	0x6E00 0018	**Instance**	GPMC
Description	This interrupt status register regroups all the status of the module internal events that can generate an interrupt.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				WAIT3EDGEDETECTIONSTATUS	WAIT2EDGEDETECTIONSTATUS	WAIT1EDGEDETECTIONSTATUS	WAIT0EDGEDETECTIONSTATUS	RESERVED						TERMINALCOUNTSTATUS	FIFOEVENTSTATUS

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
11	WAIT3EDGEDETECTION STATUS	Status of the Wait3 Edge Detection interrupt Read 0x0: A transition on WAIT3 input pin has not been detected Write 0x0: WAIT3EDGEDETECTIONSTATUS bit unchanged Read 0x1: A transition on WAIT3 input pin has been detected Write 0x1: WAIT3EDGEDETECTIONSTATUS bit is reset	RW	0x0
10	WAIT2EDGEDETECTION STATUS	Status of the Wait2 Edge Detection interrupt Read 0x0: A transition on WAIT2 input pin has not been detected Write 0x0: WAIT2EDGEDETECTIONSTATUS bit unchanged Read 0x1: A transition on WAIT2 input pin has been detected Write 0x1: WAIT2EDGEDETECTIONSTATUS bit is reset	RW	0x0
9	WAIT1EDGEDETECTION STATUS	Status of the Wait1 Edge Detection interrupt Read 0x0: A transition on WAIT1 input pin has not been detected Write 0x0: WAIT1EDGEDETECTIONSTATUS bit unchanged Read 0x1: A transition on WAIT1 input pin has been detected Write 0x1: WAIT1EDGEDETECTIONSTATUS bit is reset	RW	0x0
8	WAIT0EDGEDETECTION STATUS	Status of the Wait0 Edge Detection interrupt Read 0x0: A transition on WAIT0 input pin has not been detected Write 0x0: WAIT0EDGEDETECTIONSTATUS bit unchanged Read 0x1: A transition on WAIT0 input pin has been detected Write 0x1: WAIT0EDGEDETECTIONSTATUS bit is reset	RW	0x0
7:2	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00

Bits	Field Name	Description	Type	Reset
1	TERMINALCOUNTSTATUS	Status of the TerminalCountEvent interrupt Read 0x0: Indicates that CountValue is greater than 0 Write 0x0: TERMINALCOUNTSTATUS bit unchanged Read 0x1: Indicates that CountValue is equal to 0 Write 0x1: TERMINALCOUNTSTATUS bit is reset	RW	0x0
0	FIFOEVENTSTATUS	Status of the FIFOEvent interrupt Read 0x0: Indicates than less than FIFOThreshold bytes are available in prefetch mode and less than FIFOThreshold bytes free places are available in write-posting mode. Write 0x0: FIFOEVENTSTATUS bit unchanged Read 0x1: Indicates than at least FIFOThreshold bytes are available in prefetch mode and at least FIFOThreshold bytes free places are available in write-posting mode. Write 0x1: FIFOEVENTSTATUS bit is reset	RW	0x0

9.1.7.2.4 GPMC_IRQENABLE

Table 9-32. GPMC_IRQENABLE

Address Offset	0x0000 001C		
Physical Address	0x6E00 001C	**Instance**	GPMC
Description	The interrupt enable register allows to mask/unmask the module internal sources of interrupt, on a event-by-event basis.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				WAIT3EDGEDETECTIONENABLE	WAIT2EDGEDETECTIONENABLE	WAIT1EDGEDETECTIONENABLE	WAIT0EDGEDETECTIONENABLE	RESERVED						TERMINALCOUNTEVENTENABLE	FIFOEVENTENABLE

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
11	WAIT3EDGEDETECTION ENABLE	Enables the Wait3 Edge Detection interrupt 0x0: Wait3EdgeDetection interrupt is masked 0x1: Wait3EdgeDetection event generates an interrupt if occurs	RW	0x0
10	WAIT2EDGEDETECTION ENABLE	Enables the Wait2 Edge Detection interrupt 0x0: Wait2EdgeDetection interrupt is masked 0x1: Wait2EdgeDetection event generates an interrupt if occurs	RW	0x0
9	WAIT1EDGEDETECTION ENABLE	Enables the Wait1 Edge Detection interrupt 0x0: Wait1EdgeDetection interrupt is masked 0x1: Wait1EdgeDetection event generates an interrupt if occurs	RW	0x0
8	WAIT0EDGEDETECTION ENABLE	Enables the Wait0 Edge Detection interrupt 0x0: Wait0EdgeDetection interrupt is masked 0x1: Wait0EdgeDetection event generates an interrupt if occurs	RW	0x0
7:2	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
1	TERMINALCOUNTEVENT ENABLE	Enables TerminalCountEvent interrupt issuing in pre-fetch or write-posting mode 0x0: TerminalCountEvent interrupt is masked 0x1: TerminalCountEvent interrupt is not masked	RW	0x0
0	FIFOEVENTENABLE	Enables the FIFOEvent interrupt 0x0: FIFOEvent interrupt is masked 0x1: FIFOEvent interrupt is not masked	RW	0x0

9.1.7.2.5 GPMC_TIMEOUT_CONTROL

Table 9-33. GPMC_TIMEOUT_CONTROL

Address Offset	0x0000 0040		
Physical Address	0x6E00 0040	**Instance**	GPMC
Description	The GPMC_TIMEOUT_CONTROL register allows the user to set the start value of the timeout counter		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																TIMEOUTSTARTVALUE												RESERVED			TIMEOUTENABLE

Bits	Field Name	Description	Type	Reset
31:13	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
12:4	TIMEOUTSTARTVALUE	Start value of the time-out counter 0x000: Zero GPMC_FCLK cycle 0x001: One GPMC_FCLK cycle ... 0x1FF: 511 GPMC_FCLK cycles	RW	0x1FF
3:1	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
0	TIMEOUTENABLE	Enable bit of the TimeOut feature 0x0: TimeOut feature is disabled 0x1: TimeOut feature is enabled	RW	0x0

9.1.7.2.6 GPMC_ERR_ADDRESS

Table 9-34. GPMC_ERR_ADDRESS

Address Offset	0x0000 0044		
Physical Address	0x6E00 0044	**Instance**	GPMC
Description	The GPMC_ERR_ADDRESS register stores the address of the illegal access when an error occurs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	ILLEGALADD																														

Bits	Field Name	Description	Type	Reset
31	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
30:0	ILLEGALADD	Address of illegal access A30: 0 for memory region, 1 for GPMC register region A29-A0: 1 GBytes max	R	0x00000000

9.1.7.2.7 GPMC_ERR_TYPE

Table 9-35. GPMC_ERR_TYPE

Address Offset	0x0000 0048		
Physical Address	0x6E00 0048	**Instance**	GPMC
Description	The GPMC_ERR_TYPE register stores the type of error when an error occurs		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					ILLEGALMCMD			RESERVED			ERRORNOTSUPPADD	ERRORNOTSUPPMCMD	ERRORTIMEOUT	RESERVED	ERRORVALID

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000000
10:8	ILLEGALMCMD	System Command of the transaction that caused the error	R	0x0
7:5	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
4	ERRORNOTSUPPADD	Not supported Address error 0x0: No error occurs 0x1: The error is due to a non supported Address	R	0x0
3	ERRORNOTSUPPMCMD	Not supported Command error 0x0: No error occurs 0x1: The error is due to a non supported Command	R	0x0
2	ERRORTIMEOUT	Time-out error 0x0: No error occurs 0x1: The error is due to a time out	R	0x0
1	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
0	ERRORVALID	Error validity status - Must be explicitly cleared with a write 1 transaction 0x0: All error fields no longer valid 0x1: Error detected and logged in the other error fields	RW	0x0

9.1.7.2.8 GPMC_CONFIG

Table 9-36. GPMC_CONFIG

Address Offset	0x0000 0050		
Physical Address	0x6E00 0050	**Instance**	GPMC
Description	The configuration register allows global configuration of the GPMC		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				WAIT3PINPOLARITY	WAIT2PINPOLARITY	WAIT1PINPOLARITY	WAIT0PINPOLARITY	RESERVED			WRITEPROTECT	RESERVED		LIMITEDADDRESS	NANDFORCEPOSTEDWRITE

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
11	WAIT3PINPOLARITY	Selects the polarity of input pin WAIT3 0x0: WAIT3 active low 0x1: WAIT3 active high	RW	0x1
10	WAIT2PINPOLARITY	Selects the polarity of input pin WAIT2 0x0: WAIT2 active low 0x1: WAIT2 active high	RW	0x0
9	WAIT1PINPOLARITY	Selects the polarity of input pin WAIT1 0x0: WAIT1 active low 0x1: WAIT1 active high	RW	0x1
8	WAIT0PINPOLARITY	Selects the polarity of input pin WAIT0 0x0: WAIT0 active low 0x1: WAIT0 active high	RW	0x0
7:5	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
4	WRITEPROTECT	Controls the WP output pin level 0x0: WP output pin is low 0x1: WP output pin is high	RW	0x0
3:2	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
1	LIMITEDADDRESS	Limited Address device support 0x0: No effect 0x1: A26-A11 are not modified during an external memory access.	RW	0x0
0	NANDFORCEPOSTEDWRITE	Enables the Force Posted Write feature to NAND Cmd/Add/Data location 0x0: Disables Force Posted Write 0x1: Enables Force Posted Write	RW	0x0

9.1.7.2.9 GPMC_STATUS

Table 9-37. GPMC_STATUS

Address Offset	0x0000 0054		
Physical Address	0x6E00 0054	**Instance**	GPMC
Description	The status register provides global status bits of the GPMC		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				WAIT3STATUS	WAIT2STATUS	WAIT1STATUS	WAIT0STATUS	RESERVED							EMPTYWRITEBUFFERSTATUS

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
11	WAIT3STATUS	Is a copy of input pin WAIT3. (Reset value is WAIT3 input pin sampled at IC reset) 0x0: WAIT3 asserted (inactive state) 0x1: WAIT3 de-asserted	R	0x-
10	WAIT2STATUS	Is a copy of input pin WAIT2. (Reset value is WAIT2 input pin sampled at IC reset) 0x0: WAIT2 asserted (inactive state) 0x1: WAIT2 de-asserted	R	0x-
9	WAIT1STATUS	Is a copy of input pin WAIT1. (Reset value is WAIT1 input pin sampled at IC reset) 0x0: WAIT1 asserted (inactive state) 0x1: WAIT1 de-asserted	R	0x-
8	WAIT0STATUS	Is a copy of input pin WAIT0. (Reset value is WAIT0 input pin sampled at IC reset) 0x0: WAIT0 asserted (inactive state) 0x1: WAIT0 de-asserted	R	0x-
7:1	RESERVED	Write 0s for future compatibility. Reads returns 0	RW	0x00
0	EMPTYWRITEBUFFERSTATUS	Stores the empty status of the write buffer 0x0: Write Buffer is not empty 0x1: Write Buffer is empty	R	0x1

9.1.7.2.10 GPMC_CONFIG1_i

Table 9-38. GPMC_CONFIG1_i

Address Offset	0x0000 0060 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0060 + (0x0000 0030 * I)	**Instance**	GPMC
Description	The configuration register 1 sets signal control parameters per chip-select		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRAPBURST		READMULTIPLE		READTYPE		WRITEMULTIPLE		WRITETYPE		CLKACTIVATIONTIME				ATTACHEDDEVICEPAGELENGTH		WAITREADMONITORING		WAITWRITEMONITORING		RESERVED		WAITMONITORINGTIME				WAITPINSELECT		RESERVED		DEVICESIZE		DEVICETYPE		MUXADDDATA		RESERVED				TIMEPARAGRANULARITY		RESERVED		GPMCFCLKDIVIDER	

Bits	Field Name	Description	Type	Reset
31	WRAPBURST	Enables the wrapping burst capability. Must be set if the attached device is configured in wrapping burst 0x0: Synchronous wrapping burst not supported 0x1: Synchronous wrapping burst supported	RW	0x0
30	READMULTIPLE	Selects the read single or multiple access 0x0: Single access 0x1: Multiple access (burst if synchronous, page if asynchronous)	RW	0x0
29	READTYPE	Selects the read mode operation 0x0: Read Asynchronous 0x1: Read Synchronous	RW	0x0
28	WRITEMULTIPLE	Selects the write single or multiple access 0x0: Single access 0x1: Multiple access (burst if synchronous, considered as single if asynchronous)	RW	0x0
27	WRITETYPE	Selects the write mode operation 0x0: Write Asynchronous 0x1: Write Synchronous	RW	0x0
26:25	CLKACTIVATIONTIME	Output GPMC_CLK activation time 0x0: First rising edge of GPMC_CLK at start access time 0x1: First rising edge of GPMC_CLK one GPMC_FCLK cycle after start access time 0x2: First rising edge of GPMC_CLK two GPMC_FCLK cycles after start access time 0x3: Reserved	RW	0x0
24:23	ATTACHEDDEVICEPAGE LENGTH	Specifies the attached device page (burst) length 0x0: 4 Words 0x1: 8 Words 0x2: 16 Words 0x3: Reserved (1 Word = Interface size)	RW	0x0

Bits	Field Name	Description	Type	Reset
22	WAITREADMONITORING	Selects the Wait monitoring configuration for Read accesses (Reset value is BOOTWAITEN input pin sampled at IC reset) 0x0: Wait pin is not monitored for read accesses 0x1: Wait pin is monitored for read accesses	RW	0x-
21	WAITWRITEMONITORING	Selects the Wait monitoring configuration for Write accesses 0x0: Wait pin is not monitored for write accesses 0x1: Wait pin is monitored for write accesses	RW	0x0
20	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
19:18	WAITMONITORINGTIME	Selects input pin Wait monitoring time 0x0: Wait pin is monitored with valid data 0x1: Wait pin is monitored one GPMC_CLK cycle before valid data 0x2: Wait pin is monitored two GPMC_CLK cycle before valid data 0x3: Reserved	RW	0x0
17:16	WAITPINSELECT	Selects the input WAIT pin for this chip-select (Reset value is BOOTWAITSELECT input pin sampled at IC reset for CS0 and 0 for CS1-7) 0x0: Wait input pin is WAIT0 0x1: Wait input pin is WAIT1 0x2: Wait input pin is WAIT2 0x3: Wait input pin is WAIT3	RW	0x-
15:14	RESERVED	Write 0s for future compatibility. Reads returns 0	RW	0x0
13:12	DEVICESIZE	Selects the device size attached (Reset value is BOOTDEVICESIZE input pin sampled at IC reset for CS0 and 0x1 for CS1 to CS7) 0x0: 8 bit 0x1: 16 bit 0x2: Reserved 0x3: Reserved	RW	0x-
11:10	DEVICETYPE	Selects the attached device type 0x0: NOR Flash like, asynchronous and synchronous devices 0x1: Reserved 0x2: NAND Flash like devices, stream mode 0x3: Reserved	RW	0x0
9	MUXADDDATA	Enables the Address and data multiplexed protocol (Reset value is CS0MUXDEVICE input pin sampled at IC reset for CS0 and 0 for CS1-7) 0x0: Non Multiplexed attached device 0x1: Address and data multiplexed attached device	RW	0x-
8:5	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
4	TIMEPARAGRANULARITY	Signals timing latencies scalar factor (Rd/WrCycleTime, Rd/WrAccessTime, PageBurstAccessTime, CSOnTime, CSRd/WrOffTime, ADVOnTime, ADVRd/WrOffTime, OEOnTime, OEOffTime, WEOnTime, WEOffTime, Cycle2CycleDelay, BusTurnAround, TimeOutStartValue) 0x0: x1 latencies 0x1: x2 latencies	RW	0x0
3:2	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0

Bits	Field Name	Description	Type	Reset
1:0	GPMCFCLKDIVIDER	Divides the GPMC_FCLK clock 0x0: GPMC_CLK frequency = GPMC_FCLK frequency 0x1: GPMC_CLK frequency = GPMC_FCLK frequency / 2 0x2: GPMC_CLK frequency = GPMC_FCLK frequency / 3 0x3: GPMC_CLK frequency = GPMC_FCLK frequency /4	RW	0x0

9.1.7.2.11 GPMC_CONFIG2_i

Table 9-39. GPMC_CONFIG2_i

Address Offset	0x0000 0064 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0064 + (0x0000 0030 * I)	**Instance**	GPMC
Description	CS signal timing parameter configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											CSWROFFTIME					RESERVED			CSRDOFFTIME					CSEXTRADELAY	RESERVED				CSONTIME		

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Write 0s for future compatibility. Reads returns 0	RW	0x000
20:16	CSWROFFTIME	CS I de-assertion time from start cycle time for write accesses 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x10
15:13	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
12:8	CSRDOFFTIME	CS I de-assertion time from start cycle time for read accesses 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x10
7	CSEXTRADELAY	CS I Add Extra Half GPMC_FCLK cycle 0x0: CS I Timing control signal is not delayed 0x1: CS I Timing control signal is delayed of half GPMC_FCLK clock cycle	RW	0x0
6:4	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
3:0	CSONTIME	CS I assertion time from start cycle time 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x1

9.1.7.2.12 GPMC_CONFIG3_i

Table 9-40. GPMC_CONFIG3_i

Address Offset	0x0000 0068 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0068 + (0x0000 0030 * I)	**Instance**	GPMC
Description	nADV signal timing parameter configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								ADVWROFFTIME								RESERVED			ADVRDOFFTIME					ADVEXTRADELAY	RESERVED				ADVONTIME		

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000
20:16	ADVWROFFTIME	nADV de-assertion time from start cycle time for write accesses 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x02
15:13	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
12:8	ADVRDOFFTIME	nADV de-assertion time from start cycle time for read accesses 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x02
7	ADVEXTRADELAY	nADV Add Extra Half GPMC_FCLK cycle 0x0: nADV Timing control signal is not delayed 0x1: nADV Timing control signal is delayed of half GPMC_FCLK clock cycle	RW	0x0
6:4	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
3:0	ADVONTIME	nADV assertion time from start cycle time 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x1

9.1.7.2.13 GPMC_CONFIG4_i

Table 9-41. GPMC_CONFIG4_i

Address Offset	0x0000 006C + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 006C + (0x0000 0030 * I)	**Instance**	GPMC
Description	nWE and nOE signals timing parameter configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			WEOFFTIME					WEEXTRADELAY	RESERVED			WEONTIME				RESERVED			OEOFFTIME					OEEXTRADELAY	RESERVED			OEONTIME			

Bits	Field Name	Description	Type	Reset
31:29	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
28:24	WEOFFTIME	nWE de-assertion time from start cycle time 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x10
23	WEEXTRADELAY	nWE Add Extra Half GPMC_FCLK cycle 0x0: nWE Timing control signal is not delayed 0x1: nWE Timing control signal is delayed of half GPMC_FCLK clock cycle	RW	0x0
22:20	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
19:16	WEONTIME	nWE assertion time from start cycle time 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x3
15:13	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
12:8	OEOFFTIME	nOE de-assertion time from start cycle time 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x10
7	OEEXTRADELAY	nOE Add Extra Half GPMC_FCLK cycle 0x0: nOE Timing control signal is not delayed 0x1: nOE Timing control signal is delayed of half GPMC_FCLK clock cycle	RW	0x0
6:4	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
3:0	OEONTIME	nOE assertion time from start cycle time 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x3

9.1.7.2.14 GPMC_CONFIG5_i

Table 9-42. GPMC_CONFIG5_i

Address Offset	0x0000 0070 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0070 + (0x0000 0030 * I)	**Instance**	GPMC
Description	RdAccessTime and CycleTime timing parameters configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				PAGEBURSTACCESSTIME				RESERVED			RDACCESSTIME					RESERVED			WRCYCLETIME					RESERVED			RDCYCLETIME				

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
27:24	PAGEBURSTACCESSTIME	Delay between successive words in a multiple access 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x1
23:21	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
20:16	RDACCESSTIME	Delay between start cycle time and first data valid 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x0F
15:13	RESERVED	Write 0s for future compatibility. Reads returns 0	RW	0x0
12:8	WRCYCLETIME	Total write cycle time 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x11
7:5	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
4:0	RDCYCLETIME	Total read cycle time 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x11

9.1.7.2.15 GPMC_CONFIG6_i

Table 9-43. GPMC_CONFIG6_i

Address Offset	0x0000 0074 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0074 + (0x0000 0030 * I)	**Instance**	GPMC
Description	WrAccessTime, WrDataOnADmuxBus, Cycle2Cycle and BusTurnAround parameters configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			WRACCESSTIME					RESERVED				WRDATAONADMUXBUS				RESERVED				CYCLE2CYCLEDELAY				CYCLE2CYCLESAMECSEN	CYCLE2CYCLEDIFFCSEN	RESERVED		BUSTURNAROUND			

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved.	RW	0x1
30:29	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
28:24	WRACCESSTIME	Delay from start access time to the GPMC_FCLK rising edge corresponding the GPMC_CLK rising edge used by the attached memory for the first data capture 0x00: 0 GPMC_FCLK cycle 0x01: 1 GPMC_FCLK cycle ... 0x1F: 31 GPMC_FCLK cycles	RW	0x0F
23:20	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
19:16	WRDATAONADMUXBUS	Specifies on which GPMC_FCLK rising edge the first data the synchronous burst write is driven in the add/data mux bus	RW	0x3
15:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
11:8	CYCLE2CYCLEDELAY	Chip-select high pulse delay between successive accesses 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x0
7	CYCLE2CYCLESAMECSEN	Add CYCLE2CYCLEDELAY between successive accesses to the same CS (any access type) 0x0: No delay between the two accesses 0x1: Add CYCLE2CYCLEDELAY	RW	0x0
6	CYCLE2CYCLEDIFFCSEN	Add CYCLE2CYCLEDELAY between successive accesses to a different CS (any access type) 0x0: No delay between the two accesses 0x1: Add CYCLE2CYCLEDELAY	RW	0x0
5:4	RESERVED	Write 0s for future compatibility. Reads returns 0	RW	0x0
3:0	BUSTURNAROUND	Bus turn around latency between successive accesses to the same CS (read to write) or to a different CS (read to read and read to write) 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0xF: 15 GPMC_FCLK cycles	RW	0x0

9.1.7.2.16 GPMC_CONFIG7_i

Table 9-44. GPMC_CONFIG7_i

Address Offset	0x0000 0078 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0078 + (0x0000 0030 * I)	**Instance**	GPMC
Description	CS address mapping configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				MASKADDRESS				RESERVED	CSVALID	BASEADDRESS					

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
11:8	MASKADDRESS	CS mask address. 0x1000: Chip-select size of 128 Mbytes 0x1100: Chip-select size of 64 Mbytes 0x1110: Chip-select size of 32 Mbytes 0x1111: Chip-select size of 16 Mbytes Other values must be avoided as they create holes in the chip-select address space.	RW	0xF
7	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
6	CSVALID	CS enable 0x0: CS disabled 0x1: CS enabled	RW	See (1)
5:0	BASEADDRESS	CS base address (16 Mbytes minimum granularity) Bits [2:0] correspond to A26, A25 and A24, and bits [5:3] are not used. See Figure 9-6.	RW	0x00

(1) Reset value is 0x1 for CS0 and 0x0 for CS1 to CS7

9.1.7.2.17 GPMC_NAND_COMMAND_i

Table 9-45. GPMC_NAND_COMMAND_i

Address Offset	0x0000 007C + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 007C + (0x0000 0030 * I)	**Instance**	GPMC
Description	This register is not a true register, just an address location.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPMC_NAND_COMMAND																															

Bits	Field Name	Description	Type	Reset
31:0	GPMC_NAND_COMMAND	This register is not a true register, just an address location.	W	n/a

9.1.7.2.18 GPMC_NAND_ADDRESS_i

Table 9-46. GPMC_NAND_ADDRESS_i

Address Offset	0x0000 0080 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0080 + (0x0000 0030 * I)	**Instance**	GPMC
Description	This register is not a true register, just an address location.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPMC_NAND_ADDRESS																															

Bits	Field Name	Description	Type	Reset
31:0	GPMC_NAND_ADDRESS	This register is not a true register, just an address location.	W	n/a

9.1.7.2.19 GPMC_NAND_DATA_i

Table 9-47. GPMC_NAND_DATA_i

Address Offset	0x0000 0084 + (0x0000 0030 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0084 + (0x0000 0030 * I)	**Instance**	GPMC
Description	This register is not a true register, just an address location.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPMC_NAND_DATA																															

Bits	Field Name	Description	Type	Reset
31:0	GPMC_NAND_DATA	This register is not a true register, just an address location.	W	n/a

9.1.7.2.20 GPMC_PREFETCH_CONFIG1

Table 9-48. GPMC_PREFETCH_CONFIG1

Address Offset	0x0000 01E0		
Physical Address	0x6E00 01E0	**Instance**	GPMC
Description	Prefetch engine configuration 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	CYCLEOPTIMIZATION			ENABLEOPTIMIZEDACCESS	ENGINECSSELECTOR			PFPWENROUNDROBIN	RESERVED			PFPWWEIGHTEDPRIO				RESERVED	FIFOTHRESHOLD							ENABLEENGINE	RESERVED	WAITPINSELECTOR		SYNCHROMODE	DMAMODE	RESERVED	ACCESSMODE

Bits	Field Name	Description	Type	Reset
31	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
30:28	CYCLEOPTIMIZATION	Define the number of GPMC_FCLK cycles to be subtracted from RdCycleTime, WrCycleTime, AccessTime, CSRdOffTime, CSWrOffTime, ADVRdOffTime, ADVWrOffTime, OEOffTime, WEOffTime 0x0: 0 GPMC_FCLK cycle 0x1: 1 GPMC_FCLK cycle ... 0x7: 7 GPMC_FCLK cycles	RW	0x0
27	ENABLEOPTIMIZEDACCESS	Enables access cycle optimization 0x0: Access cycle optimization is disabled 0x1: Access cycle optimization is enabled	RW	0x0
26:24	ENGINECSSELECTOR	Selects the CS where Prefetch Postwrite engine is active 0x0 corresponds to CS0 0x1: CS1 ... 0x7: CS7	RW	0x0
23	PFPWENROUNDROBIN	Enables the PFPW RoundRobin arbitration 0x0: Prefetch Postwrite engine round robin arbitration is disabled 0x1: Prefetch Postwrite engine round robin arbitration is enabled	RW	0x0
22:20	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
19:16	PFPWWEIGHTEDPRIO	When an arbitration occurs between a DMA and a PFPW engine access, the DMA is always serviced. If the PFPWEnRoundRobin is enabled, 0x0: the next access is granted to the PFPW engine, 0x1: the two next accesses are granted to the PFPW engine, ..., 0xF: the 16 next accesses are granted to the PFPW engine.	RW	0x0
15	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
14:8	FIFOTHRESHOLD	Selects the maximum number of bytes read from the FIFO or written to the FIFO by the host on a DMA or interrupt request 0x00: 0 byte 0x01: 1 byte ... 0x40: 64 bytes	RW	0x40
7	ENABLEENGINE	Enables the Prefetch Postwite engine 0x0: Prefetch Postwrite engine is disabled 0x1: Prefetch Postwrite engine is enabled	RW	0x0
6	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
5:4	WAITPINSELECTOR	Select which wait pin edge detector should start the engine in synchronized mode 0x0: Selects Wait0EdgeDetection 0x1: Selects Wait1EdgeDetection 0x2: Selects Wait2EdgeDetection 0x3: Selects Wait3EdgeDetection	RW	0x0
3	SYNCHROMODE	Selects when the engine starts the access to CS 0x0: Engine starts the access to CS as soon as STARTENGINE is set 0x1: Engine starts the access to CS as soon as STARTENGINE is set AND wait to non wait edge detection on the selected wait pin	RW	0x0
2	DMAMODE	Selects interrupt synchronization or DMA request synchronization 0x0: Interrupt synchronization is enabled. Only interrupt line will be activated on FIFO threshold crossing. 0x1: DMA request synchronization is enabled. A DMA request protocol is used.	RW	0x0
1	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
0	ACCESSMODE	Selects pre-fetch read or write-posting accesses 0x0: Pre-fetch read mode 0x1: Write-posting mode	RW	0x0

9.1.7.2.21 GPMC_PREFETCH_CONFIG2

Table 9-49. GPMC_PREFETCH_CONFIG2

Address Offset	0x0000 01E4		
Physical Address	0x6E00 01E4	**Instance**	GPMC
Description	Prefetch engine configuration 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								TRANSFERCOUNT							

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000
13:0	TRANSFERCOUNT	Selects the number of bytes to be read or written by the engine to the selected CS 0x0000: 0 byte 0x0001: 1 byte ... 0x2000: 8 Kbytes	RW	0x0000

9.1.7.2.22 GPMC_PREFETCH_CONTROL

Table 9-50. GPMC_PREFETCH_CONTROL

Address Offset	0x0000 01EC		
Physical Address	0x6E00 01EC	**Instance**	GPMC
Description	Prefetch engine control		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															STARTENGINE

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00000000
0	STARTENGINE	Resets the FIFO pointer and starts the engine Read 0x0: Engine is stopped Write 0x0 stops the engine Read 0x1: Engine is running Write 0x1 resets the FIFO pointer to 0x0 in prefetch mode and 0x40 in postwrite mode and starts the engine	RW	0x0

9.1.7.2.23 GPMC_PREFETCH_STATUS

Table 9-51. GPMC_PREFETCH_STATUS

Address Offset	0x0000 01F0		
Physical Address	0x6E00 01F0	**Instance**	GPMC
Description	Prefetch engine status		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	FIFOPOINTER							RESERVED							FIFOTHRESHOLDSTATUS	RESERVED	COUNTVALUE														

Bits	Field Name	Description	Type	Reset
31	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0
30:24	FIFOPOINTER	Number of available bytes to be read or number of free empty byte places to be written 0x00: 0 byte available to be read or 0 free empty place to be written ... 0x40: 64 bytes available to be read or 64 empty places to be written	R	0x00
23:17	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
16	FIFOTHRESHOLDSTATUS	Set when FIFOPointer exceeds FIFOThreshold value 0x0: FIFOPointer smaller or equal to FIFOThreshold. Writing to this bit has no effect 0x1: FIFOPointer greater than FIFOThreshold. Writing to this bit has no effect	R	0x0
15:14	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
13:0	COUNTVALUE	Number of remaining bytes to be read or to be written by the engine according to the TransferCount value 0x0000: 0 byte remaining to be read or to be written 0x0001: 1 byte remaining to be read or to be written ... 0x2000: 8 Kbytes remaining to be read or to be written	R	0x0000

9.1.7.2.24 GPMC_ECC_CONFIG

Table 9-52. GPMC_ECC_CONFIG

Address Offset	0x0000 01F4		
Physical Address	0x6E00 01F4	**Instance**	GPMC
Description	ECC configuration		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															ECCALGORITHM	RESERVED			ECCBCHT8	ECCWRAPMODE				ECC16B	ECCTOPSECTOR			ECCCS			ECCENABLE

Bits	Field Name	Description	Type	Reset
31:17	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0000
16	ECCALGORITHM	ECC algorithm used 0x0: Hamming code 0x1: BCH code	RW	0x0
15:13	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
12	ECCBCHT8	Error correction capability used for BCH 0x0: Up to 4 bits error correction (t = 4) 0x1: Up to 8 bits error correction (t = 8)	RW	0x1
11:8	ECCWRAPMODE	Spare area organization definition for the BCH algorithm. See the BCH syndrome/parity calculator module functional specification for more details	RW	0x0
7	ECC16B	Selects an ECC calculated on 16 columns 0x0: ECC calculated on 8 columns 0x1: ECC calculated on 16 columns	RW	0x0
6:4	ECCTOPSECTOR	Number of sectors to process with the BCH algorithm 0x0: 1 sector (512kB page) 0x1: 2 sectors ... 0x3: 4 sectors (2kB page) ... 0x7: 8 sectors (4kB page)	RW	0x3
3:1	ECCCS	Selects the CS where ECC is computed 0x0: Chip-select 0 0x1: Chip-select 1 0x2: Chip-select 2 0x3: Chip-select 3 0x4: Chip-select 4 0x5: Chip-select 5 0x6: Chip-select 6 0x7: Chip-select 7	RW	0x0
0	ECCENABLE	Enables the ECC feature 0x0: ECC disabled 0x1: ECC enabled	RW	0x0

9.1.7.2.25 GPMC_ECC_CONTROL

Table 9-53. GPMC_ECC_CONTROL

Address Offset	0x0000 01F8		
Physical Address	0x6E00 01F8	**Instance**	GPMC
Description	ECC control		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							ECCCLEAR	RESERVED				ECCPOINTER			

Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000000
8	ECCCLEAR	Clear all ECC result registers Reads returns 0 Write 0x1 to this field clear all ECC result registers Write 0x0 is ignored	RW	0x0
7:4	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
3:0	ECCPOINTER	Selects ECC result register (Reads to this field give the dynamic position of the ECC pointer - Writes to this field select the ECC result register where the first ECC computation will be stored); Other enums: writing other values disables the ECC engine (ECCENABLE bit of GPMC_ECC_CONFIG set to 0) 0x0: Writing 0x0 disables the ECC engine (ECCENABLE bit of GPMC_ECC_CONFIG set to 0) 0x1: ECC result register 1 selected 0x2: ECC result register 2 selected 0x3: ECC result register 3 selected 0x4: ECC result register 4 selected 0x5: ECC result register 5 selected 0x6: ECC result register 6 selected 0x7: ECC result register 7 selected 0x8: ECC result register 8 selected 0x9: ECC result register 9 selected	RW	0x0

9.1.7.2.26 GPMC_ECC_SIZE_CONFIG

Table 9-54. GPMC_ECC_SIZE_CONFIG

Address Offset	0x0000 01FC		
Physical Address	0x6E00 01FC	**Instance**	GPMC
Description	ECC size		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								ECCSIZE1								RESERVED				ECCSIZE0								RESERVED				ECC9RESULTSIZE	ECC8RESULTSIZE	ECC7RESULTSIZE	ECC6RESULTSIZE	ECC5RESULTSIZE	ECC4RESULTSIZE	ECC3RESULTSIZE	ECC2RESULTSIZE	ECC1RESULTSIZE

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
29:22	ECCSIZE1	Defines ECC size 1 0x00: 2 Bytes 0x01: 4 Bytes 0x02: 6 Bytes 0x03: 8 Bytes ... 0xFF: 512 Bytes	RW	0xFF
21:20	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
19:12	ECCSIZE0	Defines ECC size 0 0x00: 2 Bytes 0x01: 4 Bytes 0x02: 6 Bytes 0x03: 8 Bytes ... 0xFF: 512 Bytes	RW	0xFF
11:9	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
8	ECC9RESULTSIZE	Selects ECC size for ECC 9 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
7	ECC8RESULTSIZE	Selects ECC size for ECC 8 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
6	ECC7RESULTSIZE	Selects ECC size for ECC 7 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
5	ECC6RESULTSIZE	Selects ECC size for ECC 6 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
4	ECC5RESULTSIZE	Selects ECC size for ECC 5 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
3	ECC4RESULTSIZE	Selects ECC size for ECC 4 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0

Bits	Field Name	Description	Type	Reset
2	ECC3RESULTSIZE	Selects ECC size for ECC 3 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
1	ECC2RESULTSIZE	Selects ECC size for ECC 2 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0
0	ECC1RESULTSIZE	Selects ECC size for ECC 1 result register 0x0: ECCSIZE0 selected 0x1: ECCSIZE1 selected	RW	0x0

9.1.7.2.27 GPMC_ECCj_RESULT

Table 9-55. GPMC_ECCj_RESULT

Address Offset	0x0000 0200 + (0x0000 0004 * k)	**Index**	k = 0 to 8
Physical Address	0x6E00 0200 + (0x0000 0004 * k)	**Instance**	GPMC
Description	ECC result register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				P2048o	P1024o	P512o	P256o	P128o	P64o	P32o	P16o	P8o	P4o	P2o	P1o	RESERVED				P2048e	P1024e	P512e	P256e	P128e	P64e	P32e	P16e	P8e	P4e	P2e	P1e

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
27	P2048o	Odd row parity bit 2048, only used for ECC computed on 512 Bytes	R	0x0
26	P1024o	Odd row parity bit 1024	R	0x0
25	P512o	Odd row parity bit 512	R	0x0
24	P256o	Odd row parity bit 256	R	0x0
23	P128o	Odd row parity bit 128	R	0x0
22	P64o	Odd row parity bit 64	R	0x0
21	P32o	Odd row parity bit 32	R	0x0
20	P16o	Odd row parity bit 16	R	0x0
19	P8o	Odd row parity bit 8	R	0x0
18	P4o	Odd Column Parity bit 4	R	0x0
17	P2o	Odd Column Parity bit 2	R	0x0
16	P1o	Odd Column Parity bit 1	R	0x0
15:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
11	P2048e	Even row parity bit 2048, only used for ECC computed on 512 Bytes	R	0x0
10	P1024e	Even row parity bit 1024	R	0x0
9	P512e	Even row parity bit 512	R	0x0
8	P256e	Even row parity bit 256	R	0x0
7	P128e	Even row parity bit 128	R	0x0
6	P64e	Even row parity bit 64	R	0x0
5	P32e	Even row parity bit 32	R	0x0
4	P16e	Even row parity bit 16	R	0x0
3	P8e	Even row parity bit 8	R	0x0
2	P4e	Even column parity bit 4	R	0x0
1	P2e	Even column parity bit 2	R	0x0
0	P1e	Even column parity bit 1	R	0x0

9.1.7.2.28 GPMC_BCH_RESULT0_i

Table 9-56. GPMC_BCH_RESULT0_i

Address Offset	0x0000 0240 + (0x0000 0010 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0240 + (0x0000 0010 * I)	**Instance**	GPMC
Description	BCH ECC result (bits 0 to 31)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCH_RESULT_0																															

Bits	Field Name	Description	Type	Reset
31:0	BCH_RESULT_0	BCH ECC result (bits 0 to 31)	RW	0x00000000

9.1.7.2.29 GPMC_BCH_RESULT1_i

Table 9-57. GPMC_BCH_RESULT1_i

Address Offset	0x0000 0244 + (0x0000 0010 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0244 + (0x0000 0010 * I)	**Instance**	GPMC
Description	BCH ECC result (bits 32 to 63)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCH_RESULT_1																															

Bits	Field Name	Description	Type	Reset
31:0	BCH_RESULT_1	BCH ECC result (bits 32 to 63)	RW	0x00000000

9.1.7.2.30 GPMC_BCH_RESULT2_i

Table 9-58. GPMC_BCH_RESULT2_i

Address Offset	0x0000 0248 + (0x0000 0010 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 0248 + (0x0000 0010 * I)	**Instance**	GPMC
Description	BCH ECC result (bits 64 to 95)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCH_RESULT_2																															

Bits	Field Name	Description	Type	Reset
31:0	BCH_RESULT_2	BCH ECC result (bits 64 to 95)	RW	0x00000000

9.1.7.2.31 GPMC_BCH_RESULT3_i

Table 9-59. GPMC_BCH_RESULT3_i

Address Offset	0x0000 024C + (0x0000 0010 * I)	**Index**	I = 0 to 7
Physical Address	0x6E00 024C + (0x0000 0010 * I)	**Instance**	GPMC
Description	BCH ECC result (bits 96 to 103)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								BCH_RESULT_3							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Write 0s for future compatibility. Read returns 0s.	R	0x000000
7:0	BCH_RESULT_3	BCH ECC result (bits 96 to 103)	RW	0x00

9.1.7.2.32 GPMC_BCH_SWDATA

Table 9-60. GPMC_BCH_SWDATA

Address Offset	0x0000 02D0		
Physical Address	0x6E00 02D0	**Instance**	GPMC
Description	This register is used to directly pass data to the BCH ECC calculator without accessing the actual NAND flash interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																BCH_DATA															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility. Read returns 0s.	R	0x0000
15:0	BCH_DATA	Data to be included in the BCH calculation. Only bits 0 to 7 are taken into account if the calculator is configured to use 8 bits data (GPMC_ECC_CONFIG[7] ECC16B = 0)	RW	0x0000

9.2 SDRAM Controller (SDRC) Subsystem

This section describes the SDRAM controller (SDRC) subsystem.

9.2.1 SDRC Subsystem Overview

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

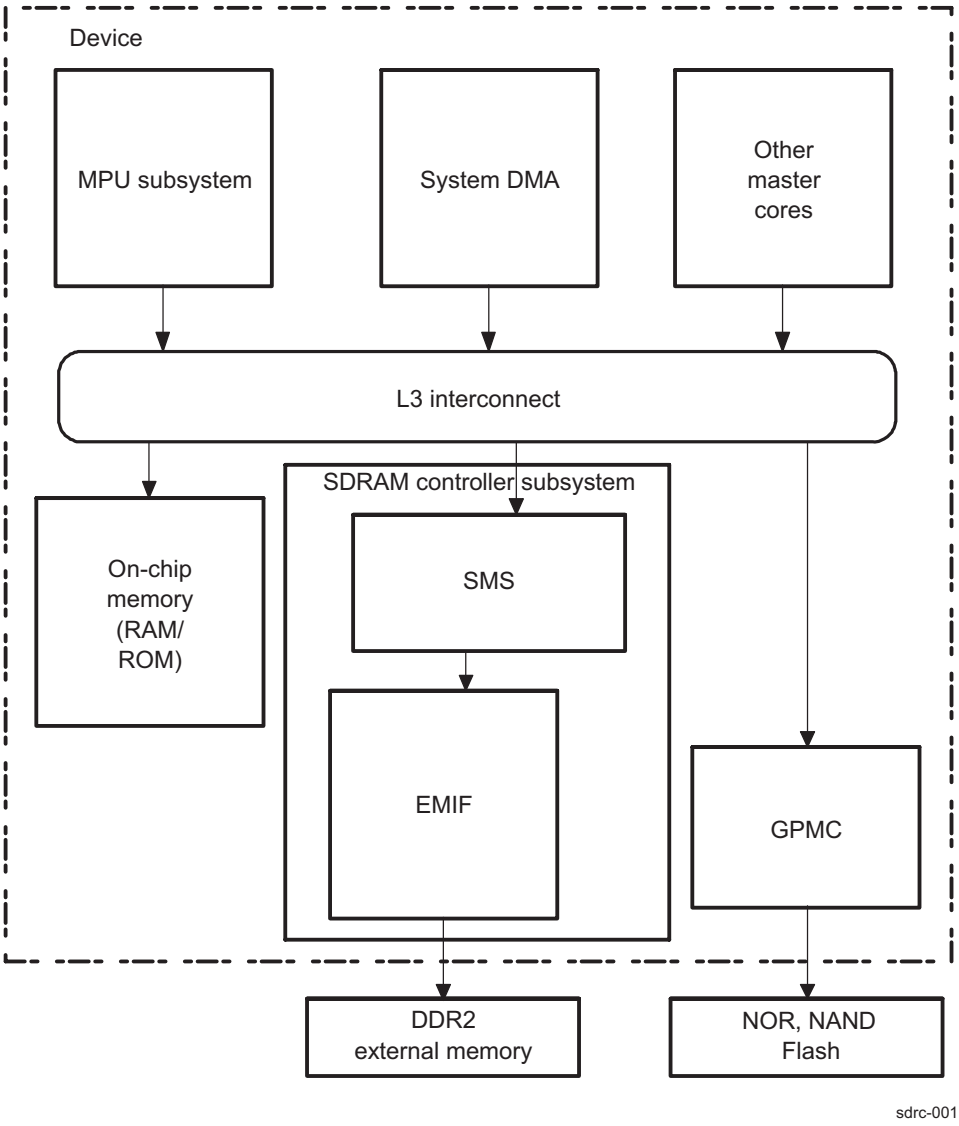
The SDRC subsystem module provides connectivity between the processor and external discrete DDR SDRAM.

The SDRC subsystem provides a high-performance interface to a variety of fast memory devices. It comprises two submodules:

- The SDRAM Memory Scheduler (SMS), consisting of scheduler, security firewall, and virtual rotated frame-buffer (VRFB) modules.
- EMIF (External Memory Interface).

Figure 9-41 shows the SDRC subsystem environment.

Figure 9-41. SDRC Subsystem Environment



9.2.1.1 Features

The main features of the SDRC subsystem module are:

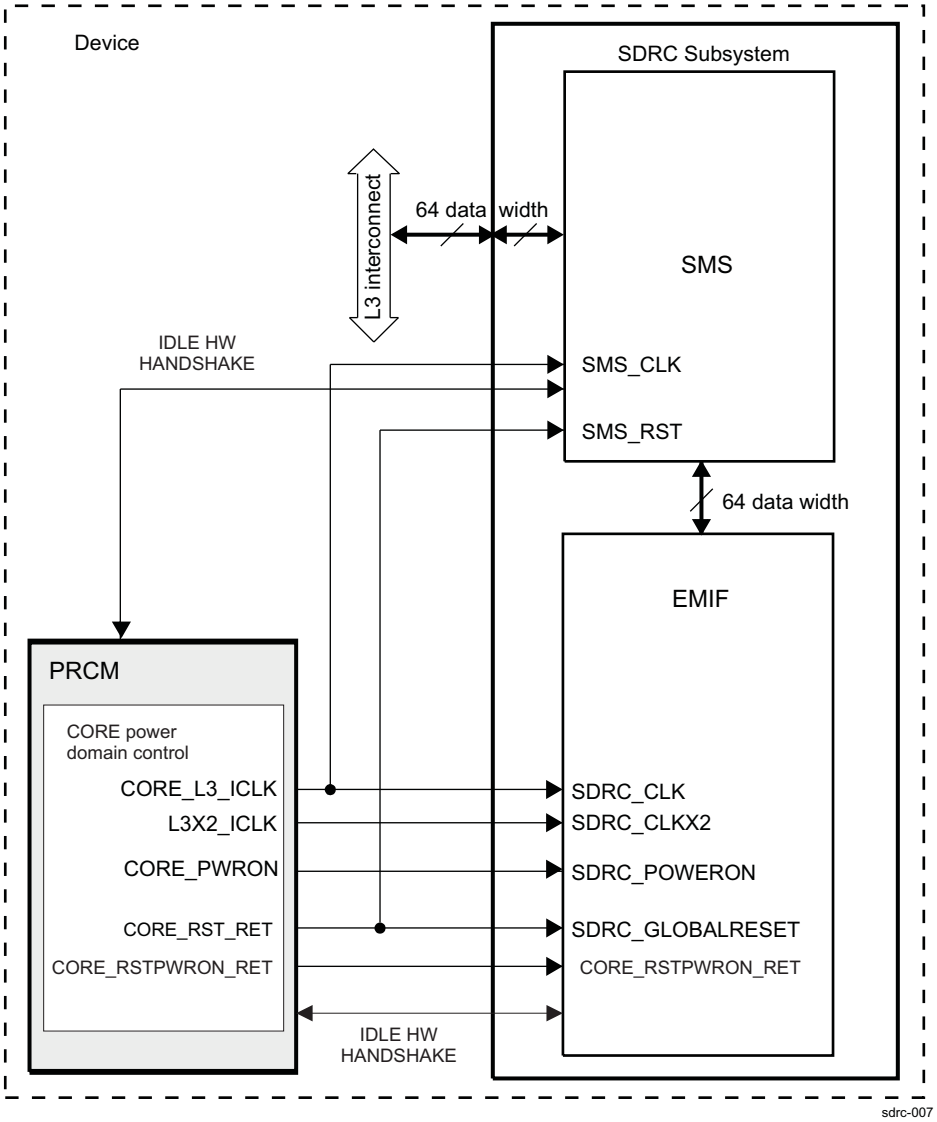
- Security firewall
  - Eight regions can be defined.
  - Independently programmable read/write access control
  - Independently programmable security control
- VRFB module
  - Minimizes SDRAM page-miss penalty when accessing graphics buffer in nonnatural raster-scan order
  - Supports rotations of 0, 90, 180, and 270 degrees
  - Transparent to software applications
  - 12 concurrent rotation contexts
- Memory-access scheduler
  - Optimizes latency and bandwidth usage between initiators
  - Supports secure transactions controlled by security firewall
  - Per-system initiator group quality-of-service (QoS) control
  - 8 × 8 × 64 request queue FIFO for optimal scheduling
  - Programmable arbitration scheme
  - Focus on real-time memory processes (LCD display, camera interface)
  - Focus on MPU memory latency
  - Fair arbitration between other system initiators (DMAs, video subsystem, SGX accelerator)
  - Exclusive read-write transaction support
  - Region-based security firewall in accordance with processor security management

For EMIF features see Section 9.2.3.4.1.1, *Features Supported*.

9.2.2 SDRC Subsystem Integration

Figure 9-42 shows how the SMS and EMIF modules are integrated into the processor and how they interact with the PRCM module.

Figure 9-42. SDRC Integration to the Processor



9.2.2.1 Clocking, Reset, and Power Management Scheme

9.2.2.1.1 Clocking

9.2.2.1.1.1 SMS

The SMS is a single clock domain module. The same clock is used for the interconnect system interface, the EMIF interface, and all internal operations.

SMS_CLK comes internally from the PRCM module and runs at the L3 interconnect frequency. It is also used as a functional clock for the SMS module.

The source of the SMS_CLK is the PRCM CORE_L3_ICLK output: CORE_L3_ICLK belongs to the L3 interconnect clock domain.

9.2.2.1.1.2 EMIF

The EMIF is a single-clock domain module. The same clock is used for both the interconnect and the memory interface. The SDRC_CLK clock comes from the PRCM and runs at the L3 interconnect frequency. SDRC_CLK is also used as a functional clock for the EMIF .

The SDRC_CLK clock source is the PRCM CORE_L3_ICLK output: CORE_L3_ICLK belongs to the L3 interconnect clock domain.

An SDRC_CLKX2 clock is provided by the PRCM at a double frequency of the SDRC_CLK clock.

As a power-saving feature, when the EMIF no longer requires the clock domain, the software can disable it at the PRCM level by setting the EN_SDRC bit in the PRCM.CM_ICKLEN1_CORE[1] register.

9.2.2.1.2 Hardware Reset

Global reset of the EMIF is done by activating the CORE_RST signal in the CORE_RST domain (see the *Power, Reset, and Clock Management* chapter).

There is one global reset signal, SDRC_GLOBALRESET, which is qualified by the signal SDRC_POWERON. This qualification differentiates whether the signal is a cold reset or a warm reset.

- On a cold reset (that is, the power-on reset, when SDRC_POWERON = 0 and SDRC_GLOBALRESET is applied), all registers and state-machines within the EMIF are asynchronously reset.
- On a warm reset (that is, any other system reset condition under control of the chip top-level power manager, SDRC_POWERON = 1 when SDRC_GLOBALRESET is applied), the EMIF registers and the FSM are not reset, but the EMIF takes care of completing the ongoing access and putting the external memory into self-refresh mode.

9.2.2.1.3 Software Reset

The SMS modules can be reset under software control through the SMS.SMS_SYSCONFIG[1] SOFTRESET bit.

A software reset has the same action as a hardware cold reset, that is, the FSM and all registers are reset immediately and unconditionally.

9.2.3 SDRC Subsystem Functional Description

The SMS optimizes the SDRAM memory usage to provide:

- The QoS level required by each of the initiators in the system
- A security firewall that controls access protection to the memory
- A VRFB module (also called the 2D rotation engine) that minimizes the SDRAM page-miss penalty when accessing rotated (that is, nonsequentially addressed) lines in a graphic frame buffer

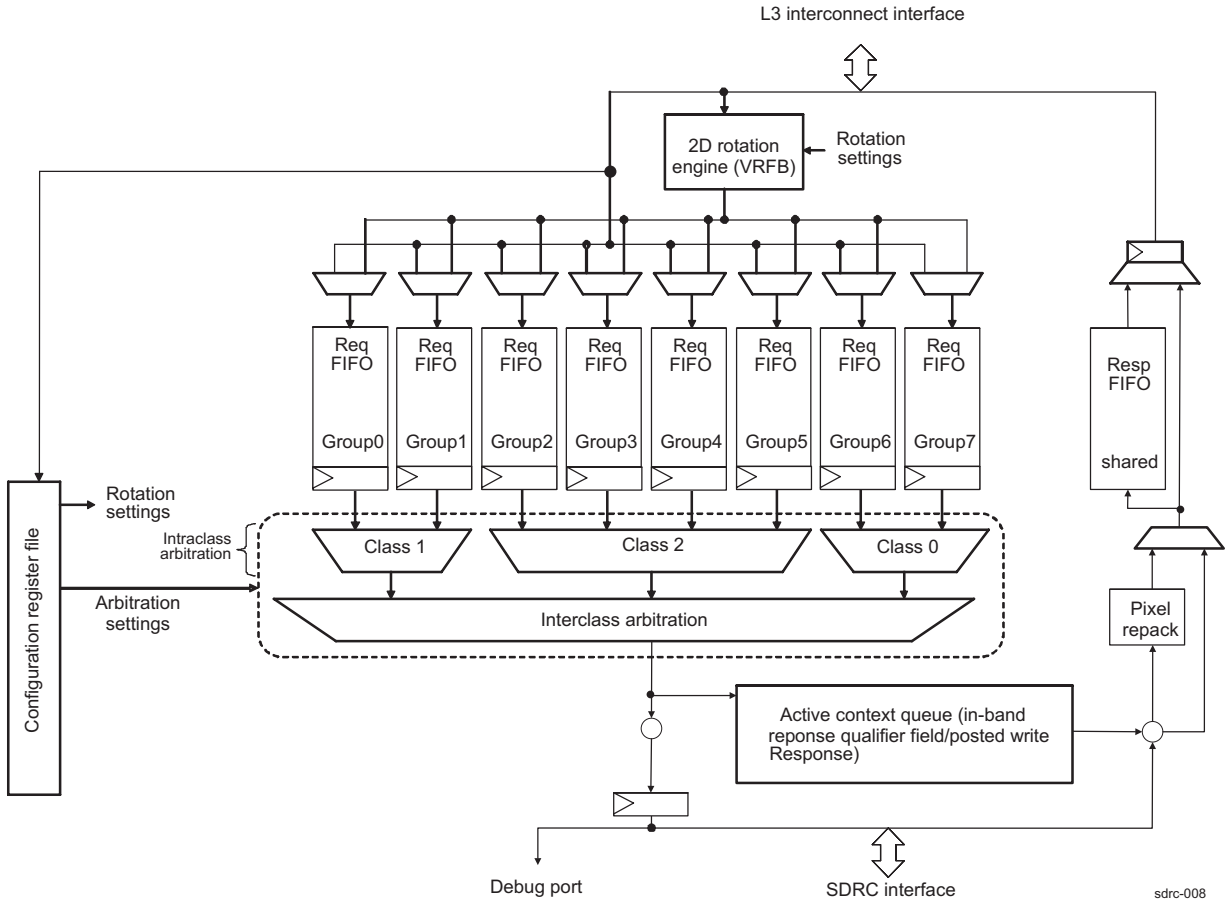
9.2.3.1 SDRAM Memory Scheduler

The SMS module is split into the following subsystems:

- L3 interconnect slave port
- VRFB: rotation engine (RE)
- Configuration register file
- Security check unit
- Request buffers
- Arbitration logic
- EMIF interface: master port
- Debug port
- Response buffer

Figure 9-43 shows the top-level diagram of the SMS.

Figure 9-43. SMS Top-Level Diagram



9.2.3.1.1 Memory Access Scheduling

NOTE: For a description of the SMS mode of operation and arbitration policy, see Section 9.2.5, *SDRC Use Cases and Tips*.

The SMS includes a set of FIFOs to queue the requests received from the system interconnect or processed by the RE. The FIFO entry contains a complete interconnect request, plus internal qualifiers added by the RE which are required to correctly process the requests modified (or generated) by the RE.

The SMS also includes a response FIFO, which is shared by all response threads. This FIFO provides full support for the response flow-control interconnect extension (handshake protocol).

When a memory transaction request arrives in the memory-access scheduler after being processed by the VRFB module, the request is sent to one of eight FIFO queues. Security firewall comes after the FIFOs.

The allocation to a particular queue is fixed and depends on the source of the request. To prioritize concurrent transactions and optimize memory usage, each FIFO queue (and hence the memory-access request source) is categorized into one of these three arbitration classes:

- Class 0 (highest priority): For bandwidth-sensitive devices with severe real-time constraints. The system fails when the bandwidth requirement is not met. LCD or video display cores belong to this category.
- Class 1: For latency-sensitive devices where system performance degrades severely when the average memory-access latency increases. These initiators also generally have some significant bandwidth requirements. All CPU cores are class 1 initiators.
- Class 2: For all other devices, possibly with high-bandwidth requirements but without being too latency-sensitive. Associated initiators may also have significant requirements in terms of bandwidth, but if the bandwidth budget requirement cannot be serviced, the system performance degrades, but remains functional (the system does not fail). General-purpose system DMA, multimedia accelerators (graphics, imaging, video) belong to this category.

The arbitration between Class 1 and Class 2 is programmable.

Table 9-61 shows the mapping of FIFO queues and memory-access request sources to arbitration classes.

Table 9-61. Arbitration Class Allocation

Class	FIFO Queue	Source Device
0	7	Display
1	0	MPU subsystem (instruction and data access)
	1	
2	2	sDMA WR
	3	SGX
	4	USB (HS + FS), DAP, CCDC, USB-OTG, EMAC
	5	sDMA (read)

A 2-level arbitration scheme determines the FIFO queue from which the next memory transaction is granted.

NOTE: In the register description, a group represents a FIFO queue of requests from the initiator.

9.2.3.1.2 Arbitration Policy

A 2-level arbitration scheme is implemented to grant access to the EMIF . The arbitration uses fully combinatorial logic, and the granted request is clocked before being issued on the interface master port of the EMIF .

- Intra-class arbitration: A first level of arbitration is done in parallel within each class between the different thread groups (request FIFOs).

- Interclass arbitration: A second level of arbitration is done among the three winners of the in-class arbitration.

9.2.3.1.2.1 Arbitration Policy within a Burst and at a Burst Boundary

Arbitration is performed on the transaction boundary. The transaction can be either a single transaction or a burst transaction.

- Within a burst, if the thread cannot provide the subsequent request of the burst, a mechanism provides a wait for one idle cycle before moving the arbitration grant to the next thread. If only one idle cycle appears on one thread, an arbitration grant must not move (the next request served must be from the same thread). One idle cycle is then inserted in the EMIF request path.
- If the thread still cannot provide the request in the next cycle, the slot can be awarded to another initiator to avoid locking the SDRAM resource, if a thread cannot supply a subsequent request within the burst. In this case, there must not be a second idle cycle insertion in the EMIF request path. A burst with fewer than two idle cycles cannot be interrupted by a higher priority request.
- On burst boundary, the arbitration does not wait one idle cycle before moving the arbitration grant. As soon as the thread cannot request one transaction at a burst boundary, the arbitration grant can move to the next thread.

This is also valid within the ExtendedGrant and NOfServices windows; as soon as the thread cannot request a transaction, the arbitration grant can move to the next thread. For more information, see the following descriptions of the ExtendedGrant and NOfServices features.

9.2.3.1.2.2 Burst-Complete Feature (BURST-COMPLETE Field in SMS_CLASS_ARBITER0, SMS_CLASS_ARBITER1, SMS_CLASS_ARBITER2)

A burst request can be submitted to the arbitration either as soon as the first request of the burst is received by the SMS, or when at least one complete burst is buffered into the FIFO. The behavior is programmable on a per-group basis using the BURST-COMPLETE field of the SMS_CLASS_ARBITER0, SMS_CLASS_ARBITER1, and SMS_CLASS_ARBITER2 registers. A per-FIFO counter tracks the number of complete bursts in a FIFO.

9.2.3.1.2.3 ExtendedGrant Feature (EXTENDEDGRANT Field in SMS_CLASS_ARBITER0, SMS_CLASS_ARBITER1, SMS_CLASS_ARBITER2)

EXTENDEDGRANT is a programmable control field that allows a group to be granted for N consecutive transactions (single or burst), assuming the group is still requesting service (FIFO is not empty). EXTENDEDGRANT is applicable on a single/burst boundary. This multiple-service grant is intended to take advantage of the high probability of two consecutive transactions within a group accessing consecutive memory addresses (that is, in the same SDRAM page). This mechanism does not apply to consecutive transactions that have been split by the RE. The maximum number of consecutive grants is given in the EXTENDEDGRANT field of the SMS_CLASS_ARBITER0, SMS_CLASS_ARBITER1, and SMS_CLASS_ARBITER2 registers. The allowed range is 1 to 3.

The ExtendedGrant logic is in the internal class arbitration but must be propagated to the interclass arbitration. The flag qualifying a second-service request is provided to the interclass arbiter so the extended grant scheme can be applied at the second level of arbitration.

- Class 0 requests can still override the ExtendedGrant scheme of class 1/class 2. Within an ExtendedGrant window, hand-over to another class/thread (granting another thread) can occur as soon as one idle cycle appears in the thread at burst or single boundary.
- The PWM counter of the interclass arbitration obeys the ExtendedGrant completion before handing priority to the other class.
- When a split transaction from the RE is interleaved within an ExtendedGrant window, completion of the ExtendedGrant window starts with the last ExtendedGrant counter value (not the initial value), because there are two independent counters for ExtendedGrant and NOfServices. The ExtendedGrant counter is reloaded to its programmed value when it reaches 0.

9.2.3.1.2.4 NOfServices Feature (SMS.SMS_CLASS_ROTATIONm[4:0] NOFSERVICES Field)

NOFSERVICES is a programmable control field that allows consecutive transactions (single or burst) coming from the VRFB (with an RE_split qualifier) to be granted consecutively, assuming the group is still requesting service (FIFO is not empty). NOfServices is applicable to the RE single/burst boundary. The maximum number of consecutive grants is given by the NOFSERVICES field of the SMS.SMS_CLASS_ROTATIONm registers. The allowable range is 1 to 31.

The NOfServices logic is in the internal class arbitration, but must be propagated to the interclass arbitration. The flag qualifying a second-service request is provided to the interclass arbiter so the extended grant scheme can be applied at the second level of arbitration.

- Class 0 requests can still override the NOfServices scheme of class 1/class 2. Within a NOfService window, hand-over to another class/thread (granting another thread) can occur as soon as one idle cycle appears in the thread at burst or single boundary. A burst with fewer than two idle cycles cannot be interrupted by a class 0 request.
- The PWM counter of the interclass arbitration obeys the NOfServices completion before handing priority to the other class.
- When a split transaction from the RE is interleaved within an ExtendedGrant window, completion of the NOfservices window starts with the last NOfServices counter value (not the initial value), because there are two independent counters for ExtendedGrant and NOfServices.

If a transaction split by the VRFB follows a nonsplit transaction currently being executed on the EMIF side, an arbitration slot occurs on the nonsplit transaction boundary, regardless of the status of the ExtendedGrant counter. This is because the chances of a nonsplit transaction and a split transaction accessing the same SDRAM page are low. Similar behavior would be observed for a split transaction followed by a nonsplit transaction. The NOfServices counter is reloaded with its programmed value when it reaches 0.

9.2.3.1.3 Internal Class Arbitration

Class 0, class 1, and class 2 internal arbitrations are performed according to the following rules:

- Within a class, a standard least-recently-used (LRU) policy is applied. The LRU thread, if not empty, is granted when an arbitration decision occurs. LRU is applied taking into account the ExtendedGrant/NOfServices counter status. Grant is given to another nonempty LRU thread only if the current thread is serviced for ExtendedGrant/NOfServices times.
- On top of the LRU policy, a high-priority group can be defined through the SMS.SMS_CLASS_ARBITER0[7:6] HIGHPRIOVECTOR field. The high-priority group is unique and programmable. If a high-priority thread in a class, which was previously empty, starts requesting service, the current thread that has been given grant is completely serviced as per ExtendedGrant/NOfServices strategy and then the grant is given to the high-priority thread.

9.2.3.1.3.1 Interclass Arbitration

The interclass arbitration is managed using a time-varying policy driven by the following rules:

- The interclass arbitration is a PWM-like (Pulse Width Modulation) logic that defines two request-based windows:
  - Class 1 has higher priority than class 2 during M requests of class 1, where M is defined by the CLASS1PRIO parameter.
  - Class 2 has higher priority than class 1 during the next N requests of class 2, where N is defined by the CLASS2PRIO parameter. For more information on priority between classes, see Section 9.2.5.2.2.3.
- The PWM counter is decremented each time the class is processing a single 64-bit request in its high-priority window.
- A class 0 request always has the highest priority. The PWM counter is frozen while class 0 requests are being serviced.
- A class 1 transaction can be serviced during the class 2 high-priority window if class 2 is not requesting service (conversely, a class 2 transaction can be serviced during the class 1 high-priority window if class 1 is not requesting service). The PWM counter is frozen when the grant is given to the

thread that is not in its high-priority window.

- NOfServices/ExtendedGrant have higher priority than the PWM counter.
- The PWM counter is reloaded with M and N when it reaches 1 and an arbitration decision must be made.
- The priority order is as follows:
  - Current burst service lock (assuming subsequent burst requests available when required)
  - Class 0
  - ExtendedGrant and NOfServices atomicity (assuming subsequent burst requests available when required)
  - Class 1 if PWM priority is to class 1; class 2 if PWM priority is to class 2
  - Class 2 if PWM priority is to class 1; class 1 if PWM priority is to class 2

9.2.3.1.4 Security Firewall

Access permissions can be defined in the target memory address space on a per-initiator basis. Initiators are differentiated using the interconnect ConnID extension.

Permissions are allocated to the various initiators on a per-region basis. The memory regions are programmable using a start address and an end address that are defined with 64 Kbytes granularity. Up to seven distinct regions can be defined; the software must ensure that they do not overlap.

The remaining memory space (total memory space minus the protected areas) is defined as region 0. This region has security attributes programmable in the same manner as the other regions.

Depending on whether the access is a read or a write, and depending on the in-band request qualifiers, a region may be given specific access permissions. When an access is received by the SMS, the access checked against the access attributes.

- The read permission is initiator-based and is controlled using the SMS.SMS_RG_RDPERMi register.
- The write permission is initiator-based and is controlled using the SMS.SMS_RG_WRPERMi register.
- The REQINFO bits taken into account are the incoming MReqInfo attributes: Security, debug, privilege, and attribute, along with the host parameter decoded in the SMS module (see the SMS.SMS_RG_ATTi[31:0] REQINFO field). For the SMS firewall, the host parameter is set for the MPU initiator and the sDMA initiator. The decoding of the host parameter, based on the MPU ConnID and sDMAConnID generic parameters (defined at design time), is done inside the SMS module.
- Whether the access is accepted (there is one valid bit for each ReqInfo pattern) can be specified for each ReqInfo pattern. ReqInfo permission is controlled using the region attributes register SMS.SMS_RG_ATTi[31:0] REQINFO field.

Table 9-62 lists the security ReqInfo parameters ordering.

Table 9-62. Security ReqInfo Parameters Ordering

Host	Privilege	Security	Debug	Type	Req Info	SMS.SMS_RG_ATTi[31:0] REQINFO Field
0: Nonhost 1: Host	**0: User 1: Supervisor**	**0: Public 1: Secure**	**0: Functional 1: Debug**	**0: Data Transfer 1: Opcode Fetch**		
N/A [1]						0b0…000000000
0	0	0	0	0	0	0b0…000000001
0	0	0	0	0	1	0b0…000000010
0	0	0	1	0	2	0b0…000000100
					…	
0	1	1	1	0	14	0b0…000001…00
0	1	1	1	1	15	0b0…00001…000
1	0	0	0	0	16	0b0…0001…0000
1	0	0	0	1	17	0b0…001…00000
					…	
1	1	1	0	1	29	0b0010…000000

(1) Access to the region is not allowed

Table 9-62. Security ReqInfo Parameters Ordering (continued)

Host	Privilege	Security	Debug	Type	Req Info	SMS.SMS_RG_ATTi[31:0] REQINFO Field
0: Nonhost 1: Host	0: User 1: Supervisor	0: Public 1: Secure	0: Functional 1: Debug	0: Data Transfer 1: Opcode Fetch		
1	1	1	1	0	30	0b0100…000000
1	1	1	1	1	31	0b1000…000000

When all SMS_RG_ATTi[31:0] REQINFO bits are set to 0 the access to the region is not allowed.

Set the REQINFO[0] bit to 1 when NonHost-User-Public-Functional-Data accesses are permitted in this region.

Set the REQINFO[1] bit to 1 when NonHost-User-Public-Debug-Data accesses are permitted in this region.

Set the REQINFO[31] bit to 1 when Host-Supervisor-Secure-Debug-Opcode accesses are permitted in this region.

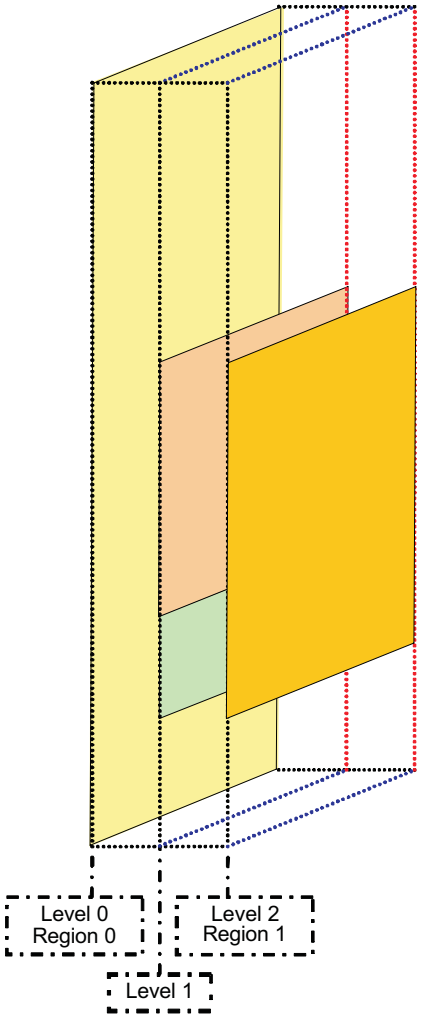
Any bit of the SMS_RG_ATTi[31:0] REQINFO field corresponds to a particular combination of the five attribute bits. When all SMS_RG_ATTi[31:0] REQINFO bits are set to 1 the region is full-access allowed.

The security unit performs the following checks to authorize or reject an access to the external memory:

- Compute the Region ID based on the transaction address.
- Generate a violation if overlapping of security regions is detected.
- From the Region ID, get the attributes for the region that has been hit.
- Check the transaction REQINFO attributes with respect to the region attributes (security, debug, privilege, type, and host).
- Reject the transaction if the attributes are not compatible.
- From the Region ID and the transaction qualifiers (MCmd, ConnID), check the initiator permissions for performing the access (read, write).
- Reject the transaction if there is no permission.

The protection regions are organized in security priority levels (from Level 0 to Level 2) to prevent problems with overlapping region and corner cases associated with them; the lowest level has the lowest priority (see Figure 9-44).

Figure 9-44. Security Region Organization



Region 0 (default region containing the whole memory space): Level 0

Region 1 (allows dynamic reprogramming of regions): Level 2

Regions 2–7 (protection regions): Level 1

Region 1 has the highest priority to perform dynamic masking of other already-programmed regions when they are reprogrammed.

Overlapping regions of the same priority level is forbidden and results in a security violation when an access to the concerned region occurs. This security violation is reported in the error-log register. Priority-level handling is done in the hardware; it does not involve any specific software development.

All transactions are checked, including those the RE has processed.

When detecting a security violation on an incoming request, the SMS respects the response ordering within the faulty thread.

A violation flag is raised internally and the MThreadID field is logged. Generation of the interconnect error response is then local to the SMS; the response buffer must be used to manage the potential response collision with regular EMIF responses.

To program a rotation context, set the security level per context through the SMS.SMS_SECURITY_CONTROL ROTCTXTiLOCK bits (i = [0:11]). That is, on context i: are all transactions allowed or is access restricted to secure transactions only?

9.2.3.1.5 Rotation Engine

Applications must often perform image rotation between the orientation of images stored in external memory and the orientation with which they must be displayed. The device offers a hardware mechanism that allows rotation tasks to be implemented efficiently, transparent to software applications, thereby keeping the MPU free for other tasks. For a description of the RE mechanism, see Section 9.2.5, *SDRC Use Cases and Tips*.

The SMS includes address processing support for rotated displays (90, 180, and 270 degrees). This function is realized by the VRFB submodule, also called the rotation engine (RE) in this document.

The primary goal of the VRFB is to eliminate the SDRAM page-miss penalty when reading graphics data in nonnatural raster scan order (that is, top to bottom, bottom to top, right to left).

The 2D-rotation module (the VRFB) is included as a black box that intercepts incoming requests when addressed to the virtual frame buffer. If the address of the request targets the VRFB address space, the request is sent to the RE. It processes the address and reinserts the modified request in the SMS request path. The SMS translates the virtual address into physical SDRAM addresses and reinserts a request or multiple requests in the SMS request path to the EMIF .

NOTE: The use of the word *virtual* does not refer to the usual CPU-related MMU concept; the word is used in a more general context of the address remapping feature, which decouples the system from the actual storage physical organization of the graphics data in the external memory.

The VRFB can be abstracted as a 3-port module:

- Interconnect input port
- Interconnect output port
- Configuration port; all programmable control registers are part of the SMS register file, which is described in Section 9.2.3.1.5.3, *VRFB Configuration Port*.

9.2.3.1.5.1 VRFB Input Port

The VRFB receives a 29-bit address, and two decoding signals, from the SMS module. The 29-bit address is decoded to determine the context and the rotation view of the request.

The VRFB address space is a 768 Mbytes address space split into two non-contiguous virtual address spaces:

- Address space 0: 256 MB in quarter Q1 (start address: 0x7000 0000, end address: 0x7FFF FFFF)
- Address space 1: 512 MB in quarter Q3 (start address: 0xE000 0000, end address: 0xFFFF FFFF)

It can manage up to 12 concurrent rotation contexts. Table 9-63 details the address space of each context.

Table 9-63. VRFB Contexts Virtual Address Spaces vs Rotation Angle

Context Number	0°	90°	180°	270°
0	0x7000 0000	0x7100 0000	0x7200 0000	0x7300 0000
1	0x7400 0000	0x7500 0000	0x7600 0000	0x7700 0000
2	0x7800 0000	0x7900 0000	0x7A00 0000	0x7B00 0000
3	0x7C00 0000	0x7D00 0000	0x7E00 0000	0x7F00 0000
4	0xE000 0000	0xE100 0000	0xE200 0000	0xE300 0000
5	0xE400 0000	0xE500 0000	0xE600 0000	0xE700 0000
6	0xE800 0000	0xE900 0000	0xEA00 0000	0xEB00 0000
7	0xEC00 0000	0xED00 0000	0xEE00 0000	0xEF00 0000
8	0xF000 0000	0xF100 0000	0xF200 0000	0xF300 0000
9	0xF400 0000	0xF500 0000	0xF600 0000	0xF700 0000
10	0xF800 0000	0xF900 0000	0xFA00 0000	0xFB00 0000

Table 9-63. VRFB Contexts Virtual Address Spaces vs Rotation Angle (continued)

Context Number	0°	90°	180°	270°
11	0xFC00 0000	0xFD00 0000	0xFE00 0000	0xFF00 0000

CAUTION

There is no protection in hardware for accesses outside the image resolution. Users can access the full virtual address range for a given context (there is no error reporting). Accessing outside of the image resolution creates aliasing within the image and is also translated as an additional physical memory space requirement on the external memory (some data outside of the image range can be overwritten by mistake). The following formula can be used to calculate the extra memory that will be accessed:
extra_mem_size = (2048-image_width_roundedup) x page_heigth x pixel_size with image_width_roundedup = [ROUNDUP(image_width x pixel_size/page_width)] x page_width/pixel_size.

9.2.3.1.5.2 VRFB Output Port

The VRFB output port can be a delayed copy of the input port, with the address field transformed. Depending on the rotated view that is active, an incoming transaction can also be split into several internal transactions.

The pipeline delay between the input and the output ports is three clock periods.

9.2.3.1.5.3 VRFB Configuration Port

The configuration port allows 12 concurrent rotation settings.

The VRFB address space is a 768 Mbytes (256 + 512) address space split into two noncontiguous spaces. It can manage up to 12 concurrent rotation settings.

The VRFB configuration port includes all the settings required to control the 12 rotation contexts. This is an input-only port. All settings are provided from the SMS control register file.

For each of the 12 contexts, the buffer physical base address, image height and width, and pixel size can be configured through the following registers (where n is the context number, from 0 to 11):

- SMS.SMS_ROT_PHYSICAL_BAn[30:0] PHYSICALBA field
- SMS.SMS_ROT_SIZEn[26:16] IMAGEHEIGHT and SMS.SMS_ROT_SIZEn[10:0] IMAGEWIDTH fields
- SMS.SMS_ROT_CONTROLn[1:0] PS field

The memory arrangement for the pixels of a frame buffer accessed through the RE is tile-based. A tile is a rectangular array of pixels. The tile size should match the page size of the SDRAM component attached to the controller for optimal performance.

The tile size is defined using the SMS.SMS_ROT_CONTROLn[10:8] page height (PH) and [6:4] page width (PW) fields.

The image height and image width (SMS.SMS_ROT_SIZEn[26:16] IMAGEHEIGHT and [10:0] IMAGEWIDTH fields, expressed in bytes) must be multiples of the tile height and width, respectively. Some padding is required if the image size does not fit this requirement.

For a configuration example, see Section 9.2.4.2 , *VRFB Context Configuration*.

9.2.3.1.6 Register Security

For the following register groups, use the SMS.SMS_SECURITY_CONTROL register to independently restrict permission to write to the SMS registers to secure accesses:

- VRFB context configuration registers (independently for each context)

- Memory-access scheduler arbitration registers
- Firewall registers, soft reset bit, error registers, and the security control register itself

Writing restrictions to the security firewall memory region registers (including SMS.SMS_SECURITY_CONTROL) depend on the control module configuration.

If an illegal access is attempted (for example, a nonsecure write access to a register configured as secure), a security violation is indicated to the security module. The address where the security violation occurs and the type of violation can be read from the SMS.SMS_ERR_ADDR and SMS.SMS_ERR_TYPE registers.

To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

9.2.3.1.7 Security Violation Reporting

The SMS firewall can detect security violations and report them to the overall system by using the following qualifiers:

- The in-band error response to a non-authorized access
- The out-of-band error signal generation using two out-of-band error signals

Based on the fact that there is no way to prevent the debugger from generating firewall violations during debug, and that users cannot stop checking for functional violations, two out-of-band security violation signals are set when:

- A functional violation is detected
- A debug violation is detected

These signals are asserted on each error detection from a read, write, or posted write faulty access: they are deasserted when the software clears the error bit in the SMS_ERR_TYPE register.

9.2.3.2 Module Power Saving

Power-saving is managed through the SMS.SMS_SYSCONFIG register.

The SMS_SYSCONFIG[4:3] SIDLEMODE field defines the power management strategy (Force Idle mode, No Idle mode, or Smart Idle mode). See Section 9.2.3.3 for more details on the system power management.

By default, the internal interface clock gating strategy is enabled as the SMS_SYSCONFIG[0] AUTOIDLE bit is set to 0x1 after reset. When all FIFO queues are empty and no ongoing transactions remain, the L3 interconnect clock is disabled inside the SMS thus reducing power consumption. The L3 interconnect clock can be disabled after a programmable delay defined in the SMS_POW_CTRL[7:0] IDLEDELAY bit field.

When there is new activity on the interconnect interface, the interconnect clock is restarted without any latency penalty. It is recommended to enable this mode to reduce power consumption.

9.2.3.3 System Power Management

The SMS can be configured through the SMS.SMS_SYSCONFIG register to be in one of these idle modes:

- No-idle mode (the SMS.SMS_SYSCONFIG[4:3] SIDLEMODE field is set to 0x1): The module never goes into idle state.
- Force-idle mode (the SMS.SMS_SYSCONFIG[4:3] SIDLEMODE field is set to 0x0): The module goes into idle state immediately after receiving the request from the PRCM.
- Smart-idle mode (the SMS.SMS_SYSCONFIG[4:3] SIDLEMODE field is set to 0x2): SidleAck is asserted once the module has confirmed there are no more outstanding transactions with the EMIF .

9.2.3.4 External Memory Interface Module (EMIF)

9.2.3.4.1 EMIF Overview

This document is intended to provide programmers with a functional presentation of the External memory Interface module. It provides a register description. In this specification, the External memory Interface will be named EMIF.

9.2.3.4.1.1 Features Supported

The main features of the controller are:

- Open Core Protocol 2.2 (OCP) compliant [7].
- Supports JEDEC standard compliant DDR2 [2] and LPDDR1 [4] devices.
  - SDRAM address range over 2 chip selects.
  - Supports following data bus widths:

OCP Data Bus Width	SDRAM Data Bus Width
64 and 128-Bit	16, 32, and 64-Bit

  - Supports following CAS latencies:

SDRAM Type	CAS Latencies
DDR2	2, 3, 4, 5, and 6
LPDDR1	2 and 3

  - Supports following number of internal banks:

SDRAM Type	Internal Banks
DDR2	1, 2, 4, and 8
LPDDR1	1, 2, and 4

  - Supports 256, 512, 1024, and 2048-word page sizes.
  - Supports following burst lengths:

SDRAM Type	Burst Length
DDR2	8 (4 not supported)
LPDDR1	8 (2 and 4 not supported)

  - Supports sequential burst type.
  - SDRAM auto initialization from reset or configuration change.
  - Supports Bank Interleaving across both the chip selects.
  - Supports Clock Stop mode for LPDDR1 for low power.
  - Supports Self Refresh and Precharge Power-Down modes for low power.
  - Supports Partial Array Self Refresh and Temperature Controlled Self Refresh modes for low power in LPDDR1.
  - Temperature Controlled Self Refresh is only supported for mobile SDRAM having on-chip temperature sensor.
  - Supports ODT on DDR2.
  - Supports prioritized refresh.
  - Programmable SDRAM refresh rate and backlog counter.
  - Programmable SDRAM timing parameters.
  - Supports only little endian.

9.2.3.4.1.2 Features Not Supported

- Interleave burst type not supported.
- Auto Precharge not supported for better Bank Interleaving performance.
- OCD calibration not supported for DDR2.
- Active Power-Down is not supported.

9.2.3.4.2 Functional Description

9.2.3.4.2.1 Functional Overview

The External Memory Interface (EMIF) is a TI developed re-usable IP component targeted for SOC designs, "4" is the major version number and "A" is the configuration type. The EMIF is an OCP slave peripheral providing an interface to a wide variety of DDR SDRAM. This memory controller is a soft macro and must be used with the DDR PHY hard macros to interface to the DDR SDRAM.

Figure 9-45. EMIF4 Top Level Block Diagram

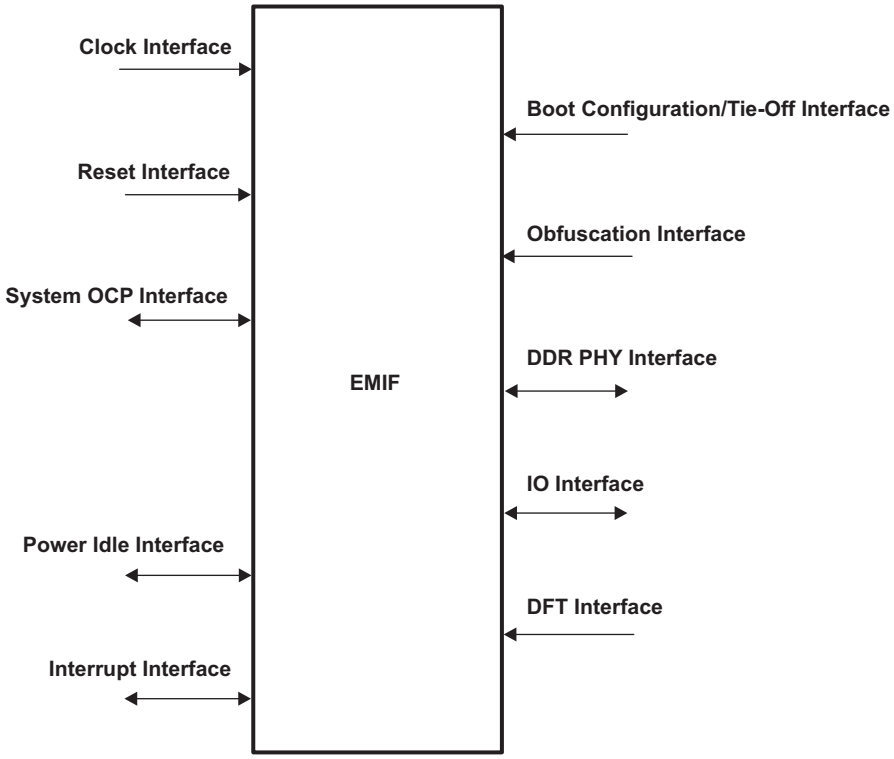


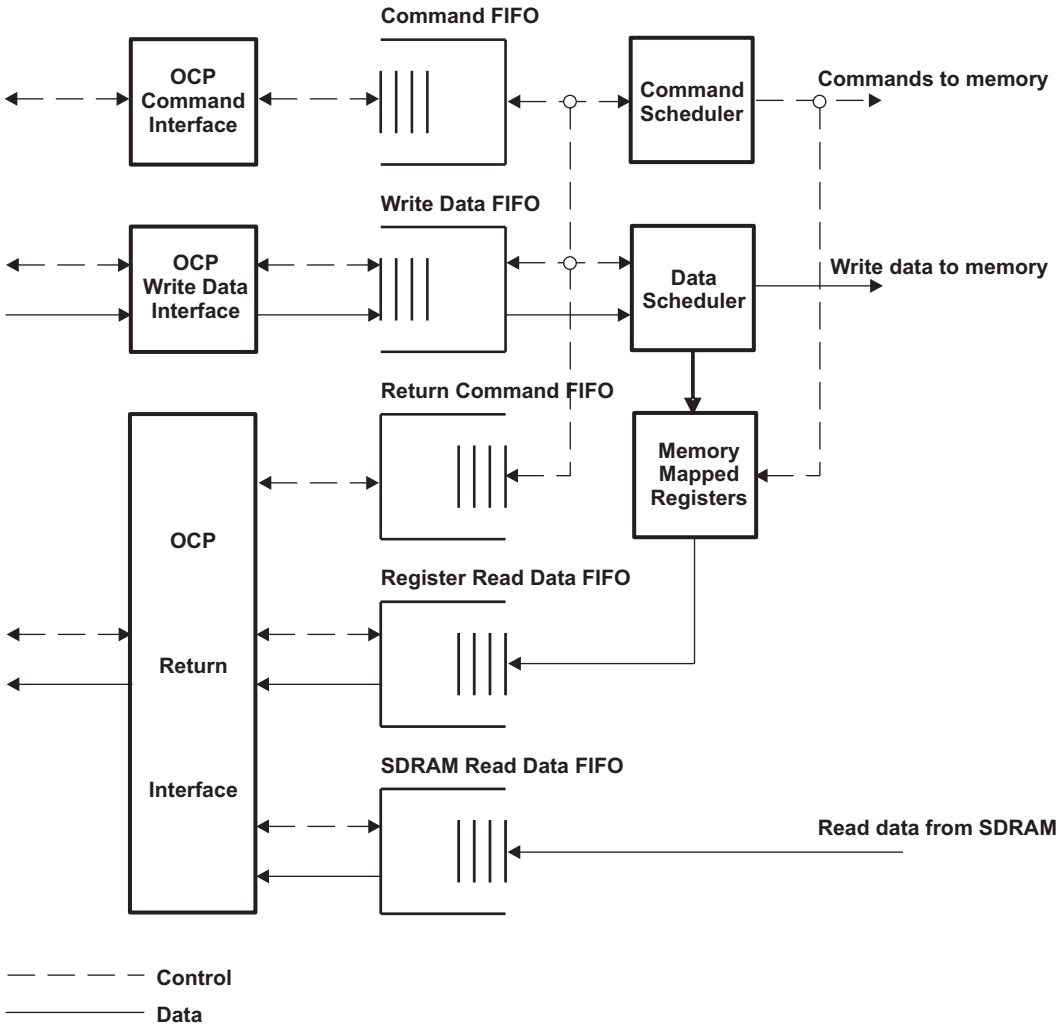
Table 9-64 shows the FIFO configurations of the EMIF.

Table 9-64. EMIF4A Configuration

Parameter	Value
OCP and Memory Clock Domains	Asynchronous
Command FIFO Entries	7
Write Data FIFO Entries	14
Return Command FIFO Entries	22
SDRAM Read Data FIFO Entries	22
Register Read Data FIFO Entries	2

9.2.3.4.2.2 EMIF4 Architecture

Figure 9-46. EMIF4 Block Diagram



The EMIF contains a Command FIFO, a Write Data FIFO, a Return Command FIFO, and two Read Data FIFOs. These FIFOs are built using flip-flops.

The Command FIFO stores all the commands coming in on the OCP command interface. If the Write Data FIFO is not full, SCmdAccept is always asserted. If the EMIF is busy and the Write Data FIFO is full, the EMIF will de-assert SCmdAccept.

The Write Data FIFO stores the write data for all the write transactions coming in on the OCP write data interface. If the Write Data FIFO is not full, SDataAccept is always asserted. If the EMIF is busy and the Write Data FIFO is full, the EMIF will de-assert SDataAccept. The Write Data FIFO stores the write data for all the write transactions coming in on the OCP write data interface. If the Write Data FIFO is not full, SDataAccept is always asserted. If the EMIF is busy and the Write Data FIFO is full, the EMIF will de-assert SDataAccept.

The Return Command FIFO stores all the return transactions that are to be issued to the OCP return interface. These include the write status return and the read data return commands.

There are two Read Data FIFOs that store the read data to be sent to the OCP return interface. One Read Data FIFO stores read data from the memory mapped registers. The other Read Data FIFO stores read data from external memory.

A write command is executed only when four OCP words of the corresponding data is received in the Write Data FIFO, or MDataLast is set, and if there is space in the Return Command FIFO. The EMIF will schedule a read only if the results can fit into both the Return Command FIFO and the corresponding Read Data FIFOs.

9.2.3.4.2.3 OCP Interface

The EMIF supports one OCP interface that connects to the system interconnect. This interface is used to request all external memory device accesses, to access the EMIF registers, and to transfer all data to/from the EMIF. Table 9-65 shows the MAddrSpace mapping to different chip selects.

Table 9-65. MAddrSpace Mapping to Chip Selects

MAddrSpace	Chip Select	Description
0x0	pad_cs_o_n[1:0]	SDRAM
0x1	N/A	Reserved
0x2	N/A	Reserved
0x3	N/A	Internal registers

The EMIF only supports Idle, Write, Read, and WriteNonPost command types (as indicated by MCmd). If an access request for an unsupported command type is received, the EMIF will send out an interrupt and treat WriteConditional and Broadcast as a Write commands, and ReadEx and ReadLinked as a Read commands. In this case, the EMIF will perform writes on the memory bus with pad_dqm_o_n set to all ones. It will not perform writes to memory mapped registers. It will perform reads on the memory bus as well as memory mapped registers, but will return all zeros for read data.

The EMIF only supports incrementing, wrapping, and 2-dimensional block addressing modes (as indicated by MBurstSeq). If an access request for an unsupported addressing mode is received, the EMIF will send out an interrupt and treat the request as an incrementing request. In this case, the EMIF will perform writes on the memory bus with pad_dqm_o_n set to all ones. It will not perform writes to memory mapped registers. It will perform reads on the memory bus as well as memory mapped registers, but will return all zeros for read data.

If an access request to an unsupported MAddrSpace is received, the EMIF will send out an interrupt. In this case, the EMIF will not perform any writes or reads. However, it will return all zeros for read data.

For all of the above errored transactions, the EMIF will report an error (SResp = 3) on the OCP return interface. The EMIF will also log the MConnID, MCmd, MBurstSeq, and MAddrSpace for the first errored transaction in the OCP Error Log register.

The EMIF supports command priorities through the MReqPriority sideband signal. It supports 8 levels of priority. A value of 0 indicates highest priority and a value of 7 indicates lowest priority.

9.2.3.4.2.4 Arbitration

The EMIF looks at all the commands stored in the Command FIFO to schedule commands to the external memory. All commands with the same MTagID will complete in order, regardless of the MAddrSpace and MReqPriority. The EMIF does not guarantee ordering between commands with different MTagID. However, the EMIF will maintain data coherency. Therefore, the EMIF will block a command, regardless of priority, if that command is to the same block address (2048 bytes) as an older command.

Thus, the EMIF might have one pending read or write for each MTagID. Among all pending reads, the EMIF selects all reads that have their corresponding SDRAM banks already open. Similarly, among all pending writes, the EMIF selects all writes that have their corresponding SDRAM banks already open. Accesses to memory mapped registers are treated as accesses that have open banks.

As a result of the above reordering, the EMIF might now have several pending reads and writes that have their corresponding banks open. The EMIF then selects the highest priority read from pending reads, and the highest priority write from pending writes. If two or more commands have the highest priority, the EMIF selects the oldest command. As a result, the EMIF might now have a final read and a final write command. If the Return Command FIFO and the Read Data FIFO has space and the external bus conflict

is resolved, the EMIF performs the final read command before the final write command. If the Return Command FIFO has space but the Read Data FIFO is full and the external bus conflict is resolved, the EMIF performs the final write command before the final read command. Resolution of external bus conflict means all the SDRAM command to command counters are satisfied and the read to write or write to read turn around time is met.

The EMIF does not support tag interleaving (tag_interleave_size = 0). In other words, the EMIF completes executing an OCP command before it switches to another command.

The data coherency inside the EMIF is only guaranteed in a single level of OCP infrastructure. For example, if a write from a secondary OCP bus segment is blocked by a bridge element, the read from a tertiary bus can still beat the write to the EMIF. In such a case, 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 status from the EMIF before indicating to master B that the data is ready to be read. If master A does not use the OCP wait status, it must perform the following sequence of operations:

1. Perform the required write.
2. Perform a dummy write to the EMIF Module ID and Revision register.
3. Perform a dummy read to the EMIF Module ID and Revision register.
4. Indicate to the other master that the data is ready to be read after completion of read in step 3.

The completion of read in step 3 ensures that the previous writes were done.

Apart from reads and writes the EMIF also needs to open and close SDRAM banks, and maintain the refresh counts for an SDRAM. The priority of SDRAM commands with respect to refresh levels are as follows:

1. (Highest priority) SDRAM refresh request due to Refresh Must level of refresh urgency reached.
2. OCP request for a read without a higher priority write.
3. SDRAM refresh request due to Refresh Need level of refresh urgency reached.
4. OCP request for a write.
5. SDRAM Activate commands.
6. SDRAM Deactivate commands.
7. SDRAM Deep Power-Down request.
8. SDRAM clock stop request.
9. SDRAM refresh request due to Refresh May or Release level of refresh urgency reached.
10. (Lowest priority) SDRAM self-refresh or SDRAM Power-Down request.

While performing the above listed scheduling algorithm, the EMIF might run into following two conditions:

1. A continuous stream of high priority commands can block lower priority commands.
2. A continuous stream of SDRAM commands to a row in an open bank can block commands to another row in the same bank.

To avoid a continuous blocking effect, the EMIF momentarily raises the priority of the oldest command over all other commands when the number or memory accesses equals to the value programmed in the reg_pr_old_count field of the OCP Configuration register.

On top of the above scheduling, the highest priority condition is a rising edge on the rst_por_arst_n or rst_main_arst_n port (i.e. removal of reset). If this condition occurs, the EMIF abandons whatever it is currently doing and commences its start-up sequence. In this case, commands and data stored in the FIFOs are lost. The EMIF also commences its start-up sequence whenever the SDRAM Configuration register is written and reg_initref_dis field in SDRAM Refresh Control register is set to 0. In this case, commands and data stored in the FIFOs are not lost. The EMIF will ensure that in-flight read or write transactions to the SDRAM are complete before starting the initialization sequence.

All the accesses to an SDRAM are pipelined to maximize the external bus utilization. In other words accesses to an SDRAM are issued back to back such that there are minimum idle cycles between any two accesses. This includes the scheduling listed above to minimize the overhead of opening and closing of SDRAM banks. All of these is done while fulfilling the access timing requirements of an SDRAM.

9.2.3.4.2.5 SDRAM Refresh Scheduling

The EMIF uses two counters to schedule AUTO REFRESH commands: a 13-bit decrementing refresh interval counter and a 4-bit refresh backlog counter. The interval counter is loaded with the reg_refresh_rate field value at reset. The interval counter decrements by one each cycle until it reaches zero at which point it reloads from reg_refresh_rate and restarts decrementing. The counter also reloads and restarts decrementing whenever the reg_refresh_rate field is updated.

The refresh backlog counter records the number of AUTO REFRESH commands the EMIF currently has outstanding. The backlog counter increments by one each time the interval counter reloads (unless it has reached its maximum value of 15). The backlog counter decrements by one each time the EMIF issues an AUTO REFRESH command (unless it is already at zero). For the range of values that the backlog counter can take, there are three levels of urgency with which the EMIF should perform an auto refresh cycle (in which it issues AUTO REFRESH commands), as follows:

- Refresh May level is reached whenever the backlog count is greater than 0, to indicate that there is a refresh backlog, so if the EMIF is not busy and none of the SDRAM banks are open, it should perform an auto refresh cycle.
- Refresh Release level is reached whenever the backlog count is greater than 4, to indicate that the refresh backlog is getting high, so if the EMIF is not busy it should perform an auto refresh cycle even if any banks are open.
- Refresh Need level is reached whenever the backlog count is greater than 5, to indicate that the refresh backlog is getting high and the EMIF should raise the priority of performing an auto refresh cycle above that of servicing new memory write requests. The EMIF starts servicing new memory accesses after Refresh Release level is cleared.
- Refresh Must level is reached whenever the backlog count is greater than 7, to indicate that the refresh backlog is getting excessive and the EMIF should perform an auto refresh cycle before servicing any new memory access requests. The EMIF starts servicing new memory accesses after Refresh Release level is cleared.

The refresh counters do not operate when SDRAM has been put into self-refresh mode. Also, the refresh counters start tracking the missed refreshes only after initialization is complete.

The EMIF issues AUTO REFRESH commands within auto refresh cycles. An auto refresh cycle consists of issuing an AUTO REFRESH command and waiting t_rfc (refer to the SDRAM Timing 3 register) cycles before re-checking the refresh levels. If the Refresh Release level is not reached, the EMIF starts another auto refresh cycle, otherwise it returns to the idle state where it can issue any command.

9.2.3.4.2.6 SDRAM Initialization

9.2.3.4.2.6.1 LPDDR1 Initialization

On coming out of reset if the reg_sdram_type field in the SDRAM Config register is equal to 1 and the reg_initref_dis bit in the SDRAM Refresh Control register is set to 0, the EMIF performs a LPDDR1 SDRAM initialization sequence as follows:

1. Drives pad_cke_o high and starts to continuously issues NOP commands.
2. After 16 SDRAM refresh rate intervals, issues a PRECHARGE command with pad_a_o[10] held high to indicate all banks. The SDRAM refresh rate is as defined in the reg_refresh_rate field description (see description of SDRAM Refresh Control register).
3. After 2 AUTO REFRESH commands, issues a LOAD MODE REGISTER command to the mode register (pad_ba_o[2:0] = 0x0) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:7]	0x0	Normal operation.
pad_a_o[6:4]	reg_cl[2:0]	CAS latency from SDRAM Config register.
pad_a_o[3]	0x0	Sequential burst type.
pad_a_o[2:0]	0x3	Burst length of 8.

4. Issues a LOAD MODE REGISTER command to the extended mode register (pad_ba_o[2:0] = 0x2) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:7]	0x0	Reserved.
pad_a_o[6:5]	reg_sdram_drive	Drive strength from SDRAM Config register.
pad_a_o[4:3]	0x0	Internal temperature compensated self-refresh.
pad_a_o[2:0]	reg_pasr	Partial array self-refresh from SDRAM Refresh Control register.

5. The EMIF performs an auto refresh cycle (see Refresh Controller section) after which it enters its idle state.

The EMIF also performs the initialization sequence whenever either of the SDRAM Configuration register is written. But in this case, the EMIF starts at step 3.

The LOAD MODE REGISTER command may be referred to as MODE REGISTER SET command in some LPDDR1 data sheets. The EMIF does not perform any transactions until the LPDDR1 initialization sequence is complete.

The reg_refresh_rate value at reset is config_refresh_def_val port value. When the EMIF comes out of reset, the delay time in step 2 resulting from the 16 refresh rate intervals + 8 cycles is approximately 16 * reg_refresh_rate / input frequency. It is up to the user to tie off the config_refresh_def_val port with a correct value to meet the typical LPDDR1 device specified delay time of 200us between power-up and the application of the PRECHARGE all command.

9.2.3.4.2.6.2 DDR2 Initialization

On coming out of reset if the reg_sdram_type field in the SDRAM Configuration register is equal to 2 and the reg_initref_dis bit in the SDRAM Refresh Control register is set to 0, the EMIF performs a DDR2 SDRAM initialization sequence as follows:

1. Drives pad_cke_o low.
2. After 16 SDRAM refresh rate intervals, issues a NOP command with pad_cke_o held high. The SDRAM refresh rate is as defined in the reg_refresh_rate field description (see description of SDRAM Refresh Control register).
3. After 1 SDRAM refresh rate interval, issues a PRECHARGE command with pad_a_o[10] held high to indicate all banks.
4. Issues a LOAD MODE REGISTER command to the extended mode register 2 (pad_ba_o[2:0] = 0x2) with pad_a_o[15:0] = 0x0.
5. Issues a LOAD MODE REGISTER command to the extended mode register 3 (pad_ba_o[2:0] = 0x3) with pad_a_o[15:0] = 0x0.
6. Issues a LOAD MODE REGISTER command to the extended mode register 1 (pad_ba_o[2:0] = 0x1) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:13]	0x0	Reserved
pad_a_o[12]	0x0	Output buffer enabled
pad_a_o[11]	0x0	RDQS disable
pad_a_o[10]	!reg_ddr2_ddqs	Differential DQS enable value from SDRAM Configuration register
pad_a_o[9:7]	0x0	Exit OCD calibration mode
pad_a_o[6]	reg_ddr_term[1]	DDR2 termination resistor value from SDRAM Configuration register
pad_a_o[5:3]	0x0	Additive latency = 0
pad_a_o[2]	reg_ddr_term[0]	DDR2 termination resistor value from SDRAM Configuration register
pad_a_o[1]	reg_sdram_drive	SDRAM drive strength from SDRAM Configuration register
pad_a_o[0]	0x0	Enable DLL

7. Issues a LOAD MODE REGISTER command to the mode register (pad_ba_o[2:0] = 0x0) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:13]	0x0	Reserved
pad_a_o[12]	0x0	Fast exit active power-down exit time
pad_a_o[11:9]	reg_t_wr	Write recovery for auto precharge from SDRAM Timing 1 register
pad_a_o[8]	0x1	DLL reset
pad_a_o[7]	0x0	Normal mode
pad_a_o[6:4]	reg_cl[2:0]	CAS latency from SDRAM Configuration register
pad_a_o[3]	0x0	Sequential burst type
pad_a_o[2:0]	0x3	Burst length of 8

8. After 200 clock cycles, issues a PRECHARGE command with pad_a_o[10] held high to indicate all banks.
9. After 2 AUTO REFRESH commands, issues a LOAD MODE REGISTER command to the mode register (pad_ba_o[2:0] = 0x0) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:9]	Equal to step 7	
pad_a_o[8]	0x0	DLL not reset
pad_a_o[7:0]	Equal to step 7	

10. Issues a LOAD MODE REGISTER command to the extended mode register 1 (pad_ba_o[2:0] = 0x1) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:10]	Equal to step 6	
pad_a_o[9:7]	0x7	Default OCD calibration
pad_a_o[6:0]	Equal to step 6	

11. Issues a LOAD MODE REGISTER command to the extended mode register 1 (pad_ba_o[2:0] = 0x1) with the pad_a_o bits equal to step 6.
12. If the reg_ddr_disable_dll bit in the SDRAM Configuration register is 1, issues a LOAD MODE REGISTER command to the extended mode register 1 (pad_ba_o[2:0] = 0x1) with the pad_a_o bits, set as follows:

Bits	Value	Description
pad_a_o[15:1]	Equal to step 6	
pad_a_o[0]	0x1	Disable DLL

13. The EMIF performs an auto refresh cycle (see Refresh Controller section) after which it enters its idle state.

The EMIF also performs the initialization sequence whenever either of the SDRAM Configuration register is written. But in this case, the EMIF starts at step 3.

The LOAD MODE REGISTER command may be referred to as MODE REGISTER SET command in some DDR2 datasheets. The EMIF does not perform any transactions until the DDR2 initialization sequence is complete.

The reg_refresh_rate value at reset is config_refresh_def_val port value. When the EMIF comes out of reset, the delay time in step 2 resulting from the 16 refresh rate intervals + 8 cycles is approximately 16 * reg_refresh_rate / input frequency. It is up to the user to tie off the config_refresh_def_val port with a correct value to meet the typical DDR2 device specified delay time of 200us between power-up and the application of the PRECHARGE all command.

9.2.3.4.2.7 EMIF Access Cycles

By default the EMIF keeps SDRAM chip selects active. To direct a command to just one of the SDRAMs it deasserts the chip select to the other SDRAM for the duration of the command. If reg_ebank field in the SDRAM Configuration register is set to 0, chip select 1 will always be driven high except during initialization and for refresh, Power Down, Self Refresh, and Deep Power Down commands.

The EMIF always performs burst accesses to SDRAM. Multiple SDRAM bursts may be needed to service a single OCP burst request. Figure 9-47 to Figure 9-51 show a few examples on how EMIF accesses SDRAM for a linear incrementing transaction type. T0, T1, etc. are clock cycles. R0 is read starting at column 0, R8 is read starting at column 8, and R16 is read starting at column 16. D0-1 is the data from column 0 and 1, D2-3 is the data from column 2 and 3, and so on.

Figure 9-47. 64-Byte Linear Read Starting at Address 0x0

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
R0				R8							
				D0-1	D2-3	D4-5	D6-7	D8-9	D10-11	D12-13	D14-15

Figure 9-48. 64-Byte Linear Read Starting at Address 0x8 (LPDDR1)

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
R2			R8				R16							
				D2-3	D4-5	D6-7	D8-9	D10-11	D12-13	D14-15	D16-17	Unused	Unused	Unused

Figure 9-49. 64-Byte Linear Read Starting at Address 0x8

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
R2		R4		R8				R16							
				D2-3	Unused	D4-5	D6-7	D8-9	D10-11	D12-13	D14-15	D16-17	Unused	Unused	Unused

Figure 9-50. 64-Byte Linear Read Starting at Address 0x10

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
R4		R8				R16							
				D4-5	D6-7	D8-9	D10-11	D12-13	D14-15	D16-17	D18-19	Unused	Unused

Figure 9-51. 64-Byte Linear Read Starting at Address 0x18

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
R6		R8				R16							
				D6-7	Unused	D8-9	D10-11	D12-13	D14-15	D16-17	D18-19	D20-21	Unused

EMIF will use the unused data phases in the above figure by issuing successive read commands if there are reads to open banks pending in the command FIFO.

The write data conversion from single data rate to double data rate must be done outside the EMIF. The SDRAM_DQS signal generation (with right timing) from the phy_dqs_en signal must also be done outside the EMIF.

9.2.3.4.2.8 Turn Around Time

Table 9-66 shows the turn around time that the EMIF introduces on the data bus for various back-to-back accesses. Note that the EMIF takes advantage of the CAS latencies and packs the commands as close as possible on the control bus to introduce the following turn around time on the data bus.

Table 9-66. Turn Around Time

Previous Access	Next Access	Turn Around Time (# of m_clk cycles)
SDRAM read	SDRAM read to same chip select	0
SDRAM read	SDRAM read to different chip select	2

Table 9-66. Turn Around Time (continued)

Previous Access	Next Access	Turn Around Time (# of m_clk cycles)
SDRAM read	SDRAM write	2
SDRAM write	SDRAM write	0
SDRAM write	SDRAM read	2

9.2.3.4.2.9 SDRAM Address Mapping

The SDRAM controller interleaves the internal banks for SDRAM connected to both the chip selects. From the system point of view, the external SDRAM is seen as one block of SDRAM. If smaller devices are used, the memory is seen to rollover. If two external 64 MB devices are used, a 128 MB memory block is observed. If two external 32 MB devices are used, a 64 MB block is observed.

When addressing SDRAM, if the reg_ibank_pos field in the SDRAM Configuration register is set to 0, the EMIF uses the 3 fields, reg_ibank, reg_ebank and reg_pagesize in the SDRAM Configuration register to determine the mapping from source address to SDRAM row, column, bank, and chip select. If the reg_ibank_pos field in the SDRAM Configuration register is set to 1, 2, or 3, the EMIF uses the 4 fields - reg_ibank, reg_ebank, reg_pagesize, and reg_rowsize in the SDRAM Configuration register to determine the mapping from source address to SDRAM row, column, bank, and chip select. In all cases the EMIF considers its SDRAM address space to be a single logical block regardless of the number of physical devices or whether the devices are mapped across 1 or 2 EMIF chip selects. For reg_ibank_pos = 0, Table 9-67 and Table 9-68 show which source address bits, MAddr, map to the SDRAM row, column, bank, and chip select bits for all combinations of reg_ibank, reg_ebank, and reg_pagesize. For reg_ibank_pos != 0, Table 9-69 to Table 9-71 shows which source address bits, MAddr, map to the SDRAM row, column, bank, and chip select bits for all combinations of reg_ibank, reg_ebank, reg_pagesize, and reg_rowsize. The tables also give the maximum size of the resulting SDRAM space.

Table 9-67. Address to SDRAM Address Mapping for 16-Bit SDRAM (reg_ibank_pos=0)

Reach (Mbytes)	reg_ebank	reg_ibank	reg_page size	MAddr															
				31	30	29	28	27	26	25	24:16	15	14	13	12	11	10	9	8:01
32	0	0	0	-							row								col
64	1	0	0	-							row							cs	col
64	0	1	0	-							row							bank	col
128	1	1	0	-							row						cs	bank	col
128	0	2	0	-							row						bank		col
256	1	2	0	-						row						cs	bank		col
256	0	3	0	-						row						bank			col
512	1	3	0	-					row						cs	bank			col
64	0	0	1	-							row							col	
128	1	0	1	-							row						cs	col	
128	0	1	1	-							row						bank	col	
256	1	1	1	-						row						cs	bank	col	
256	0	2	1	-						row						bank		col	
512	1	2	1	-					row						cs	bank		col	
512	0	3	1	-					row						bank			col	
1024	1	3	1	-				row						cs	bank			col	
128	0	0	2	-							row							col	
256	1	0	2	-						row						cs		col	
256	0	1	2	-						row						bank		col	
512	1	1	2	-					row						cs	bank		col	
512	0	2	2	-					row						bank			col	
1024	1	2	2	-				row						cs	bank			col	
1024	0	3	2	-				row						bank				col	
2048	1	3	2	-			row						cs	bank				col	
256	0	0	3	-						row						col			
512	1	0	3	-					row						cs	col			
512	0	1	3	-					row						bank	col			
1024	1	1	3	-				row						cs	bank	col			
1024	0	2	3	-				row						bank		col			
2048	1	2	3	-			row						cs	bank		col			
2048	0	3	3	-			row						bank			col			
4096	1	3	3	row								cs	bank			col			

Table 9-68. OCP Address to SDRAM Address Mapping for 32-Bit SDRAM (reg_ibank_pos=0)

Reach (Mbytes)	reg_ebank	reg_ibank	reg_page size	MAddr														
				31	30	29	28	27	26	25:17	16	15	14	13	12	11	10	9:02
64	0	0	0	-						row								col
128	1	0	0	-						row							cs	col
128	0	1	0	-						row							bank	col
256	1	1	0	-					row							cs	bank	col
256	0	2	0	-					row							bank		col
512	1	2	0	-				row							cs	bank		col
512	0	3	0	-				row							bank			col
1024	1	3	0	-			row							cs	bank			col
128	0	0	1	-						row							col	
256	1	0	1	-					row							cs	col	
256	0	1	1	-					row							bank	col	
512	1	1	1	-				row							cs	bank	col	
512	0	2	1	-				row							bank		col	
1024	1	2	1	-			row							cs	bank		col	
1024	0	3	1	-			row							bank			col	
2048	1	3	1	-		row							cs	bank			col	
256	0	0	2	-					row							col		
512	1	0	2	-				row							cs	col		
512	0	1	2	-				row							bank	col		
1024	1	1	2	-			row							cs	bank	col		
1024	0	2	2	-			row							bank		col		
2048	1	2	2	-		row							cs	bank		col		
2048	0	3	2	-		row							bank			col		
4096	1	3	2	row								cs	bank			col		
512	0	0	3	-				row							col			
1024	1	0	3	-			row							cs	col			
1024	0	1	3	-			row							bank	col			
2048	1	1	3	-		row							cs	bank	col			
2048	0	2	3	-		row							bank		col			
4096	1	2	3	row								cs	bank		col			
4096	0	3	3	row									bank		col			
4096	1	3	3	row								cs	bank		col			

For reg_ibank_pos = 0, the effect of the address-mapping scheme is that as the source address increments across SDRAM page boundaries, the EMIF moves onto the same page in the next bank in the current device (pad_cs_o_n[0]). This movement along the banks of the current device continues until the page has been accessed in all banks in the current device. The EMIF then proceeds to the same page in the next device (if reg_ebank = 1, pad_cs_o_n[1]) and proceeds through the same page in all its banks before moving over to the next page in the first device (pad_cs_o_n[0]). The EMIF exploits this traversal across internal banks and chip selects while remaining on the same page to maximize the number of open SDRAM banks within the overall SDRAM space.

Thus, the EMIF can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, and can interleave among all of them.

Table 9-69. OCP Address to SDRAM Address Mapping for reg_ibank_pos=1

MAddr[31:N][1]				
Bank Address[2]	**Row Address**	**Chip Select**	**Bank Address[1:0]**	**Column Address**
# of bits defined by reg_ibank	**# of bits defined by reg_rowsize**	**# of bits defined by reg_ebank**	**# of bits defined by reg_ibank**	**# of bits defined by reg_pagesize**
reg_ibank=0 => 0 bits	reg_rowsize=0 => 9 bits	reg_ebank=0 => 0 bits	reg_ibank=0 => 0 bits	reg_pagesize=0 => 8 bits
reg_ibank=1 => 0 bits	reg_rowsize=1 => 10 bits	reg_ebank=1 => 1 bit	reg_ibank=1 => 1 bit	reg_pagesize=1 => 9 bits
reg_ibank=2 => 0 bits	reg_rowsize=2 => 11 bits		reg_ibank=2 => 2 bits	reg_pagesize=2 => 10 bits
reg_ibank=3 => 1 bit	reg_rowsize=3 => 12 bits		reg_ibank=3 => 2 bits	reg_pagesize=3 => 11 bits
	reg_rowsize=4 => 13 bits			
	reg_rowsize=5 => 14 bits			
	reg_rowsize=6 => 15 bits			
	reg_rowsize=7 => 16 bits			

(1) N=1 for 16-bit SDRAM and N=2 for 32-bit SDRAM.

For reg_ibank_pos = 1, the EMIF interleaves banks the same as reg_ibank_pos = 0, but the interleaving of banks within a device (per chip select) is limited to 4 banks. Thus, the EMIF can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 8 of them.

Table 9-70. OCP Address to SDRAM Address Mapping for reg_ibank_pos=2

MAddr[31:N][1]				
Bank Address[2:1]	**Row Address**	**Chip Select**	**Bank Address[0]**	**Column Address**
# of bits defined by reg_ibank	**# of bits defined by reg_rowsize**	**# of bits defined by reg_ebank**	**# of bits defined by reg_ibank**	**# of bits defined by reg_pagesize**
reg_ibank=0 => 0 bits	reg_rowsize=0 => 9 bits	reg_ebank=0 => 0 bits	reg_ibank=0 => 0 bits	reg_pagesize=0 => 8 bits
reg_ibank=1 => 0 bits	reg_rowsize=1 => 10 bits	reg_ebank=1 => 1 bit	reg_ibank=1 => 1 bit	reg_pagesize=1 => 9 bits
reg_ibank=2 => 1 bit	reg_rowsize=2 => 11 bits		reg_ibank=2 => 1 bit	reg_pagesize=2 => 10 bits
reg_ibank=3 => 2 bits	reg_rowsize=3 => 12 bits		reg_ibank=3 => 1 bit	reg_pagesize=3 => 11 bits
	reg_rowsize=4 => 13 bits			
	reg_rowsize=5 => 14 bits			
	reg_rowsize=6 => 15 bits			
	reg_rowsize=7 => 16 bits			

(1) N=1 for 16-bit SDRAM and N=2 for 32-bit SDRAM.

For reg_ibank_pos = 2, the EMIF interleaves banks the same as reg_ibank_pos = 0, but the interleaving of banks within a device (per chip select) is limited to 2 banks. Thus, the EMIF can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 4 of them.

Table 9-71. OCP Address to SDRAM Address Mapping for ibank_pos=3

MAddr[31:N][1]			
Bank Address	**Row Address**	**Chip Select**	**Column Address**
# of bits defined by reg_ibank	**# of bits defined by reg_rowsize**	**# of bits defined by reg_ebank**	**# of bits defined by reg_pagesize**
reg_ibank=0 => 0 bits	reg_rowsize=0 => 9 bits	reg_ebank=0 => 0 bits	reg_pagesize=0 => 8 bits
reg_ibank=1 => 1 bit	reg_rowsize=1 => 10 bits	reg_ebank=1 => 1 bit	reg_pagesize=1 => 9 bits
reg_ibank=2 => 2 bits	reg_rowsize=2 => 11 bits		reg_pagesize=2 => 10 bits
reg_ibank=3 => 3 bits	reg_rowsize=3 => 12 bits		reg_pagesize=3 => 11 bits
	reg_rowsize=4 => 13 bits		
	reg_rowsize=5 => 14 bits		
	reg_rowsize=6 => 15 bits		
	reg_rowsize=7 => 16 bits		

(1) N=1 for 16-bit SDRAM and N=2 for 32-bit SDRAM.

For reg_ibank_pos = 3, the EMIF cannot interleave banks within a device (per chip select). However, it can still interleave banks between the two chip selects. Thus, the EMIF can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 2 of them.

Since the EMIF interleave among less number of banks when reg_ibank_pos != 0, these cases are lower in performance than the reg_ibank_pos = 0 case. Thus these cases are only recommended to be used along with partial array self-refresh where performance can be traded off for power savings.

9.2.3.4.2.10 Endian Support

The order in which bytes on the OCP port data bus is written to or read from devices that are not as wide as OCP port data bus is determined by the setting of the config_big_endian pin at the time of an access. The EMIF maintains the natural order of endian operations. That is, if a byte stream of data starting at any address n within any endian environment will always be accessed in the correct or incrementing byte order. The EMIF will always access address n prior to n+1 in all endian modes and in any byte width.

Since the endian mode must be known prior to the CPU booting, the endian mode pin on the EMIF and the booting CPU are normally registered by a boot options register at reset time. The setting of the config_big_endian pin is reflected in the be bit of the Status register.

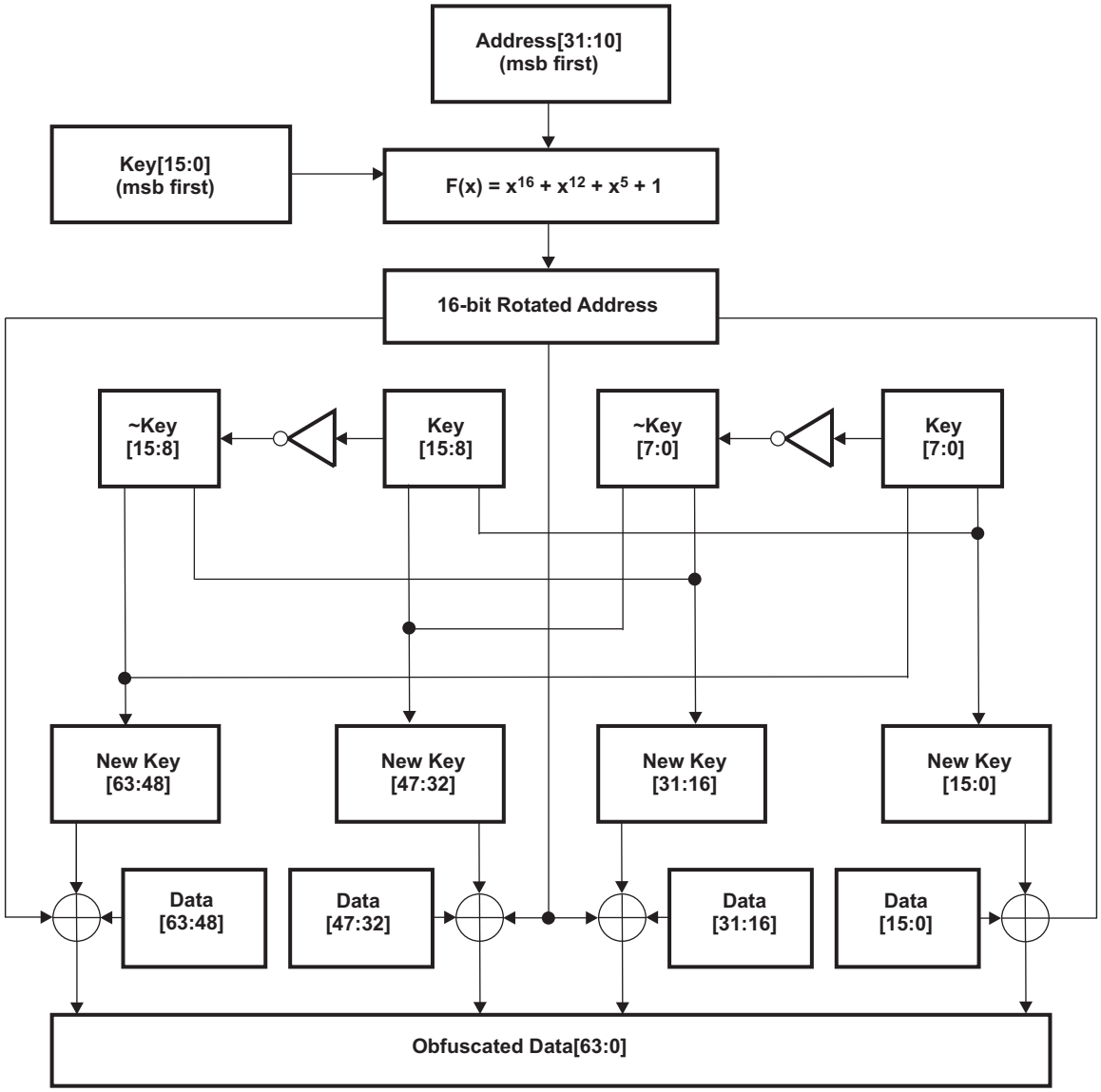
9.2.3.4.2.11 Data Bus Obfuscation

For security, the EMIF supports obfuscation for data written to the SDRAM. The obfuscation scheme for data written to the SDRAM is shown in Figure 9-52. The 16-bit key for obfuscation is sourced from the obfuscation_key port, The key can be a tie-off, or can come from a separate module at chip level, or can also come from efuse. Before it is used, the key is further modified, as shown in Figure 9-52, to create a 64-bit new key. Data will be obfuscated going out of the EMIF to the DDR PHY, and will be de-obfuscated going back to the master.

To ensure that the obfuscation is different for different regions of the memory, the upper address bits, MAddr[31:10], are taken through a LFSR function to generate a 16-bit rotated address. The obfuscation_key is used as a seed for the LFSR.

The data is then XOR'd with four copies of the 16-bit rotated address and the 64-bit new key. This ensures that the obfuscation will be different for each 1 KB page of the memory. It also helps EMIF to keep the obfuscation transparent to a master that reads or writes the memory in different transaction sizes.

Figure 9-52. Data Bus Obfuscation



9.2.3.4.2.12 Pipeline Latency

Following is the information on read latency through the EMIF and DDR PHY sub-system. EMIF is assumed to be idle, i.e., not performing any operations on the external interface and the Command FIFO is empty. Also, the read is assumed to be to an open row.

1 ocp_clk cycle - EMIF captures OCP command

1 ocp_clk cycle - SDRAM address calculation

3 m_clk cycles - synchronizer delay to memory clock (EMIF uses a 2 clock synchronizer plus 1 clock for the flop after the synchronizer)

1 m_clk cycle - command to external interface

1 m_clk cycle - SDRAM registers command

CL m_clk cycles - CAS latency

PL m_clk cycles - DDR PHY latency (minimum 1 to maximum 8 cycles depending on system round trip delay)

DC m_clk cycles - data capture/reassembly to Read Data FIFO

3 ocp_clk cycles - synchronizer delay to memory clock (EMIF uses a 2 clock synchronizer plus 1 clock for the flop after the synchronizer)

1 ocp_clk cycle - read request with data/status to OCP interface

((6 ocp_clk cycles) + ((6+CL+PL+DC) m_clk cycles)) total from request rising to first data produced on the OCP interface

SV and SM = 0 if ocp_clk and m_clk are synchronous

DC = 1 for OCP to SDRAM data width ratio of 1:2

DC = 2 for OCP to SDRAM data width ratio of 1:4

9.2.3.4.2.13 Special Operation for Integration with SMS

The EMIF supports a single bit sys_MReqMisc signal to enable a special operation for integration with SMS. This special mechanism is required by SMS to improve the overall system performance for high priority commands.

The EMIF will deassert sys_SCmdAccept output whenever the Command FIFO inside the EMIF has 4 commands that have the sys_MReqMisc signal set high. A fifth command with sys_MReqMisc high might already be in the pipe in front of the Command FIFO. In other words, either 4 or 5 command that have sys_MReqMisc signal high will be present inside the EMIF when EMIF drives sys_SCmdAccept low.

The SMS will ensure that the sys_MReqMisc signal is asserted high only for the commands required to be counted by the EMIF to apply back pressure. All other commands will have sys_MReqMisc set to 0. For systems without the SMS, the sys_MReqMisc signal must be tied low.

9.2.3.4.3 EMIF Registers

The following section describes the register map and register definition in ascending order. The state of each register after reset is shown.

For an EMIF with OCP data width greater than 32 bits, the 32-bit words are swapped internally depending on the endianness to keep the register memory map same for both little and big endian modes. This allows the same boot code to be run regardless of endianness. For example, for a 64-bit OCP data width, see Table 9-72.

Table 9-72. 64-bit OCP Data Width

Endian	Bits 63:32	Bits 31:0
Little	SDRAM Status Register	EMIF Module ID and Revision Register
Big	EMIF Module ID and Revision Register	SDRAM Status Register

9.2.3.4.3.1 EMIF Register Mapping Summary

Table 9-73. EMIF Register Mapping Summary

Offset	Acronym	Register Description	Section
0x00	EMIF_MOD_ID_REV	EMIF Module ID and Revision Register	Section 9.2.3.4.3.2.1
0x04	STATUS	SDRAM Status Register	Section 9.2.3.4.3.2.2
0x08	SDRAM_CONFIG	SDRAM Configuration Register	Section 9.2.3.4.3.2.3
0x10	SDRAM_REF_CTRL	SDRAM Refresh Control Register	Section 9.2.3.4.3.2.4
0x14	SDRAM_REF_CTRL_SHDW	SDRAM Refresh Control Shadow Register	Section 9.2.3.4.3.2.5
0x18	SDRAM_TIM_1	SDRAM Timing 1 Register	Section 9.2.3.4.3.2.6
0x1C	SDRAM_TIM_1_SHDW	SDRAM Timing 1 Shadow Register	Section 9.2.3.4.3.2.7
0x20	SDRAM_TIM_2	SDRAM Timing 2 Register	Section 9.2.3.4.3.2.8
0x24	SDRAM_TIM_2_SHDW	SDRAM Timing 2 Shadow Register	Section 9.2.3.4.3.2.9
0x28	SDRAM_TIM_3	SDRAM Timing 3 Register	Section 9.2.3.4.3.2.10
0x2C	SDRAM_TIM_3_SHDW	SDRAM Timing 3 Shadow Register	Section 9.2.3.4.3.2.11
0x38	PWR_MGMT_CTRL	Power Management Control Register	Section 9.2.3.4.3.2.12
0x3C	PWR_MGMT_CTRL_SHDW	Power Management Control Shadow Register	Section 9.2.3.4.3.2.13
0x54	OCP_CONFIG	OCP Configuration Register	Section 9.2.3.4.3.2.14
0x58	OCP_CFG_VAL_1	OCP Configuration Value 1 Register	Section 9.2.3.4.3.2.15
0x5C	OCP_CFG_VAL_2	OCP Configuration Value 2 Register	Section 9.2.3.4.3.2.16
0x60	IODFT_TLGC	IODFT Test Logic Global Control Register	Section 9.2.3.4.3.2.17
0x64	IODFT_CTRL_MISR_RSLT	IODFT Test Logic Control MISR Result Register	Section 9.2.3.4.3.2.18
0x68	IODFT_ADDR_MISR_RSLT	IODFT Test Logic Address MISR Result Register	Section 9.2.3.4.3.2.19
0x6C	IODFT_DATA_MISR_RSLT_1	IODFT Test Logic Data MISR Result 1 Register	Section 9.2.3.4.3.2.20
0x70	IODFT_DATA_MISR_RSLT_2	IODFT Test Logic Data MISR Result 2 Register	Section 9.2.3.4.3.2.21
0x74	IODFT_DATA_MISR_RSLT_3	IODFT Test Logic Data MISR Result 3 Register	Section 9.2.3.4.3.2.22
0x80	PERF_CNT_1	Performance Counter 1 Register	Section 9.2.3.4.3.2.23

Table 9-73. EMIF Register Mapping Summary (continued)

Offset	Acronym	Register Description	Section
0x84	PERF_CNT_2	Performance Counter 2 Register	Section 9.2.3.4.3.2.24
0x88	PERF_CNT_CFG	Performance Counter Configuration Register	Section 9.2.3.4.3.2.25
0x8C	PERF_CNT_SEL	Performance Counter Master Region Select Register	Section 9.2.3.4.3.2.26
0x90	PERF_CNT_TIM	Performance Counter Time Register	Section 9.2.3.4.3.2.27
0xA0	IRQ_EOI	End of Interrupt Register	Section 9.2.3.4.3.2.28
0xA4	IRQSTATUS_RAW_SYS	System OCP Interrupt Raw Status Register	Section 9.2.3.4.3.2.29
0xAC	IRQSTATUS_SYS	System OCP Interrupt Status Register	Section 9.2.3.4.3.2.30
0xB4	IRQENABLE_SET_SYS	System OCP Interrupt Enable Set Register	Section 9.2.3.4.3.2.31
0xBC	IRQENABLE_CLR_SYS	System OCP Interrupt Enable Clear Register	Section 9.2.3.4.3.2.32
0xD0	OCP_ERR_LOG	OCP Error Log Register	Section 9.2.3.4.3.2.33
0xE4	DDR_PHY_CTRL_1	DDR PHY Control 1 Register	Section 9.2.3.4.3.2.34
0xE8	DDR_PHY_CTRL_1_SHDW	DDR PHY Control 1 Shadow Register	Section 9.2.3.4.3.2.35
0xEC	DDR_PHY_CTRL_2	DDR PHY Control 2 Register	Section 9.2.3.4.3.2.36

9.2.3.4.3.2 Register Descriptions

9.2.3.4.3.2.1 EMIF Module ID and Revision Register (EMIF_MOD_ID_REV)

The EMIF Module ID and Revision Register (EMIF_MOD_ID_REV) is shown in Figure 9-53 and described in Table 9-74.

Figure 9-53. EMIF Module ID and Revision Register (EMIF_MOD_ID_REV)

31	30	29	28	27	24
REG_SCHEME		Reserved		REG_MODULE_ID	
R-0		R-0		R-0x41	

23	16
REG_MODULE_ID	
R-0x41	

15	11	8
REG_RTL_VERSION	REG_MAJOR_REVISION	
R-0	R-0x4	

7	6	5	0
Reserved		REG_MINOR_REVISION	
R-0		R-1	

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

Table 9-74. EMIF Module ID and Revision Register (EMIF_MOD_ID_REV) Field Descriptions

Bit	Field	Value	Description
31-30	REG_SCHEME	0-3h	Used to distinguish between old and current revision schemes. 0 means old scheme. 1 means new scheme. The present register format is compliant to the new scheme. Old scheme had a different format for this register.
29-28	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
27-16	REG_MODULE_ID	0-FFFh	EMIF module ID.
15-11	REG_RTL_VERSION	0-Fh	RTL Version.
10-8	REG_MAJOR_REVISION	0-7h	Major Revision.
7-6	Reserved	0	Reserved.
5-0	REG_MINOR_REVISION	0-3Fh	Minor Revision.

9.2.3.4.3.2.2 SDRAM Status Register (STATUS)

The SDRAM Status Register (STATUS) is shown in Figure 9-54 and described in Table 9-75.

Figure 9-54. SDRAM Status Register (STATUS)

31	30	29	28	24
REG_BE	REG_DUAL_CLK_MODE	REG_FAST_INIT	Reserved	
R-0	R-0	R-0	R-0	

23	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	3	2	1	0
Reserved		REG_PHY_DLL_READY	Reserved	
R-0		R-0	R-0	

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

Table 9-75. SDRAM Status Register (STATUS) Field Descriptions

Bit	Field	Value	Description
31	REG_BE		Big Endian. Reflects the value on the config_big_endian port that defines whether the EMIF is in big or little endian mode.
30	REG_DUAL_CLK_MODE		Dual Clock mode. Reflects the value on the config_dual_clk_mode port that defines whether the ocp_clk and m_clk are asynchronous.
29	REG_FAST_INIT		Fast Initialization. Reflects the value on the config_fast_init port that defines whether the EMIF fast initialization mode has been enabled.
28-3	Reserved	0	Reserved.
2	REG_PHY_DLL_READY		DDR PHY Ready. Reflects the value on the phy_ready port (active high) that defines whether the DDR PHY is ready for normal operation.
1-0	Reserved	0	Reserved.

9.2.3.4.3.2.3 SDRAM Configuration Register (SDRAM_CONFIG)

The SDRAM Configuration Register (SDRAM_CONFIG) is shown in Figure 9-55 and described in Table 9-76.

NOTE:

1. This register can only be written if lock_config_ctrl port is set to 0.
2. A write to this register will cause the EMIF to start the SDRAM initialization sequence for DDR2.
3. If reg_cl field is changed, the read latency in the PHY Control 1 register must also be reprogrammed to the correct value.

Figure 9-55. SDRAM Configuration Register (SDRAM_CONFIG)

31	29	28	27	26	24
REG_SDRAM_TYPE		REG_IBANK_POS		REG_DDR_TERM	
R/W-		R/W-		R/W-	

23	22	21	20	19	18	17	16
REG_DDR2_DDQS	Reserved		REG_DDR_DISABLE_DLL	REG_SDRAM_DRIVE		Reserved	
R/W-	R-0		R/W-	R/W-		R-0	

15	14	13	10	9	8
REG_NARROW_MODE		REG_CL		REG_ROWSIZE	
R/W-		R/W-		R/W-	

7	6	4	3	2	0
REG_ROWSIZE	REG_IBANK		REG_EBANK	REG_PAGESIZE	
R/W-	R/W-		R/W-	R/W-	

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

Table 9-76. SDRAM Configuration Register (SDRAM_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-29	REG_SDRAM_TYPE	0-7h	SDRAM Type selection. Set to 0 for DDR1, set to 1 for LPDDR1, set to 2 for DDR2. All other values are reserved.
28-27	REG_IBANK_POS	0-3h	Internal bank position. Set to 0 to assign internal bank address bits from OCP address as shown in Table 4, 5, and 6. Set to 1, 2, or 3 to assign internal bank address bits from OCP address as shown in Table 7, 8, or 9 respectively.
26-24	REG_DDR_TERM	0-7h	DDR2 termination resistor value. Set to 0 to disable termination. Set to 1 for 75 ohm, set to 2 for 150 ohm, and set to 3 for 50 ohm. All other values are reserved.
23	REG_DDR2_DDQS	0-1h	DDR2 differential DQS enable. Set to 0 for single ended DQS. Set to 1 for differential DQS.
22-21	RESERVED	0	Reserved.
20	REG_DDR_DISABLE_DLL	0-1h	Disable DLL select. Set to 1 to disable DLL inside SDRAM.
19-18	REG_SDRAM_DRIVE	0-3h	SDRAM drive strength. For DDR1, set to 0 for normal, and set to 1 for weak drive strength. For DDR2, set to 0 for full, and set to 1 for reduced drive strength. For LPDDR1, set to 0 for full, set to 1 for 1/2, set to 2 for 1/4, and set to 3 for 1/8 drive strength. All other values are reserved.
17-16	RESERVED	0	Reserved.
15-14	REG_NARROW_MODE	0-3h	SDRAM data bus width. Set to 0 for 32-bit and set to 1 for 16-bit. All other values are reserved.
13-10	REG_CL	0-Fh	CAS Latency. The value of this field defines the CAS latency to be used when accessing connected SDRAM devices. Value of 2, 3, 5, and 6 (CAS latency of 2, 3, 1.5, and 2.5) are supported for DDR1. Value of 2, 3, 4, 5, and 6 (CAS latency of 2, 3, 4, 5, and 6) are supported for DDR2. Value of 2 and 3 (CAS latency of 2 and 3) are supported for LPDDR1. All other values are reserved.

Table 9-76. SDRAM Configuration Register (SDRAM_CONFIG) Field Descriptions (continued)

Bit	Field	Value	Description
9-7	REG_ROWSIZE	0-7h	Row Size. Defines the number of row address bits of connected SDRAM devices. Set to 0 for 9 row bits, set to 1 for 10 row bits, set to 2 for 11 row bits, set to 3 for 12 row bits, set to 4 for 13 row bits, set to 5 for 14 row bits, set to 6 for 15 row bits, and set to 7 for 16 row bits. This field is only used when reg_ibank_pos field in SDRAM Config register is set to 1, 2, or 3.
6-4	REG_IBANK	0-7h	Internal Bank setup. Defines number of banks inside connected SDRAM devices. Set to 0 for 1 bank, set to 1 for 2 banks, set to 2 for 4 banks, and set to 3 for 8 banks. All other values are reserved.
3	REG_EBANK	0-1h	External chip select setup. Defines whether SDRAM accesses will use 1 or 2 chip select lines. Set to 0 to use pad_cs_o_n[0] only. Set to 1 to use pad_cs_o_n[1:0].
2-0	REG_PAGESIZE	0-7h	Page Size. Defines the internal page size of connected SDRAM devices. Set to 0 for 256-word page (8 column bits), set to 1 for 512-word page (9 column bits), set to 2 for 1024-word page (10 column bits), and set to 3 for 2048-word page (11 column bits). All other values are reserved.

9.2.3.4.3.2.4 SDRAM Refresh Control Register (SDRAM_REF_CTRL)

The SDRAM Refresh Control Register (SDRAM_REF_CTRL) is shown in Figure 9-56 and described in Table 9-77.

NOTE: The reg_refresh_rate is the number of clock cycles required to cause a refresh interval of 7.8 us or 15.7 us for the appropriate SDRAM used. This field is not byte writable, i.e., all 16 bits of this field need to be written simultaneously. A 125 MHz memory system requires a 15.7 * 125 = 1962 or 0x7AA value to be written to the SDRAM Refresh Control register. If the SDRAM requires a 7.8 us refresh rate instead of a 15.7 us, then one-half the value should be written.

Figure 9-56. SDRAM Refresh Control Register (SDRAM_REF_CTRL)

31	30	27	26	24
REG_INITREF_DIS	Reserved		REG_PASR	
R/W-	R-0			

23	16
Reserved	
R-0	

15	8
REG_REFRESH_RATE	
R/W-	

7	0
REG_REFRESH_RATE	
R/W-	

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

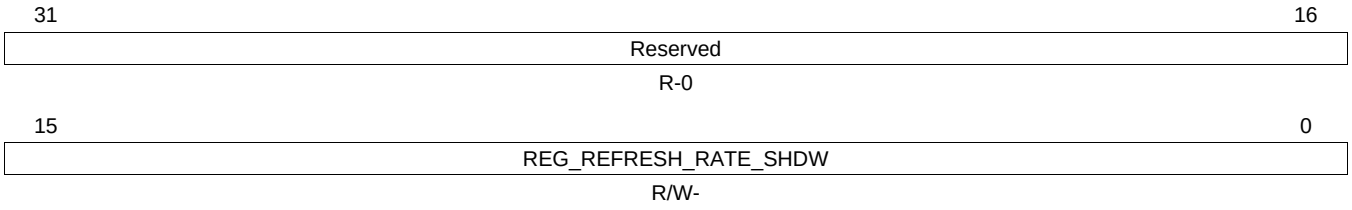
Table 9-77. SDRAM Refresh Control Register (SDRAM_REF_CTRL) Field Descriptions

Bit	Field	Value	Description
31	REG_INITREF_DIS	0-1h	Initialization and Refresh disable. When set to 1, EMIF will disable SDRAM initialization and refreshes, but will carry out SDRAM write/read transactions.
30-27	Reserved	0	Reserved.
26-24	REG_PASR	0-7h	Partial Array Self Refresh. These bits get loaded into the Extended Mode Register of an LPDDR1 during initialization. Set to 0 for full array, set to 1 for 1/2 array, set to 2 for 1/4 array, set to 5 for 1/8 array, and set to 6 for 1/16 array to be refreshed. All other values are reserved. A write to this field will cause the EMIF to start the SDRAM initialization sequence.
23-16	Reserved	0	Reserved.
15-0	REG_REFRESH_RATE	0-FFFFh	Refresh Rate. Value in this field is used to define the rate at which connected SDRAM devices will be refreshed. SDRAM refresh rate = EMIF rate / reg_refresh_rate where EMIF rate is equal to m_clk rate. If this value is less than 0x80 and the reg_t_rfc value in SDRAM Timing 3 register is less than 0x40, this field will be loaded with 0x80. If this value is less than reg_t_rfc times 2, than it is loaded with reg_t_rfc times 2.

9.2.3.4.3.2.5 SDRAM Refresh Control Shadow Register (SDRAM_REF_CTRL_SHDW)

The SDRAM Refresh Control Shadow Register (SDRAM_REF_CTRL_SHDW) is shown in Figure 9-57 and described in Table 9-78.

Figure 9-57. SDRAM Refresh Control Shadow Register (SDRAM_REF_CTRL_SHDW)



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

Table 9-78. SDRAM Refresh Control Shadow Register (SDRAM_REF_CTRL_SHDW) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-0	REG_REFRESH_RATE_SHDW	0-FFFFh	Shadow field for reg_refresh_rate. This field is loaded into reg_refresh_rate field in SDRAM Refresh Control register when SIdleAck is asserted.

9.2.3.4.3.2.6 SDRAM Timing 1 Register (SDRAM_TIM_1)

The SDRAM Timing 1 Register (SDRAM_TIM_1) is shown in Figure 9-58 and described in Table 9-79.

NOTE: If this register is byte written, care must be taken that all the fields are written before performing any accesses to the SDRAM.

Figure 9-58. SDRAM Timing 1 Register (SDRAM_TIM_1)

31 29	28 25	24 21	20 17	16
Reserved	REG_T_RP	REG_T_RCD	REG_T_WR	REG_T_RAS
R-0	R/W-	R/W-	R/W-	R/W-

15 12	11 6	5 3	2 0
REG_T_RAS	REG_T_RC	REG_T_RRD	REG_T_WTR
R/W-	R/W-	R/W-	R/W-

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

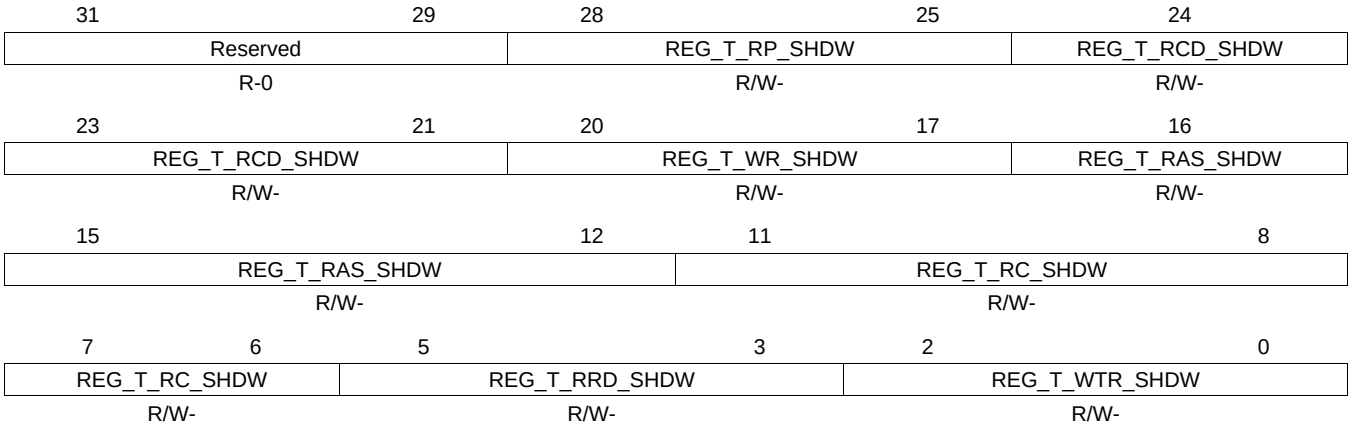
Table 9-79. SDRAM Timing 1 Register (SDRAM_TIM_1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved.
28-25	REG_T_RP	0-Fh	Minimum number of m_clk cycles from Precharge to Activate or Refresh, minus one.
24-21	REG_T_RCD	0-Fh	Minimum number of m_clk cycles from Activate to Read or Write, minus one.
20-17	REG_T_WR	0-Fh	Minimum number of m_clk cycles from last Write transfer to Pre-charge, minus one. The SDRAM initialization sequence will be started when the value of this field is changed from the previous value and the EMIF is in DDR2 mode.
16-12	REG_T_RAS	0-1Fh	Minimum number of m_clk cycles from Activate to Pre-charge, minus one. reg_t_ras >= reg_t_rcd.
11-6	REG_T_RC	0-3Fh	Minimum number of m_clk cycles from Activate to Activate, minus one.
5-3	REG_T_RRD	0-7h	Minimum number of m_clk cycles from Activate to Activate for a different bank, minus one. For an 8-bank DDR2, this field must be equal to ((tFAW/(4*tCK))-1).
2-0	REG_T_WTR	0-7h	Minimum number of m_clk cycles from last Write to Read, minus one.

9.2.3.4.3.2.7 SDRAM Timing 1 Shadow Register (SDRAM_TIM_1_SHDW)

The SDRAM Timing 1 Shadow Register (SDRAM_TIM_1_SHDW) is shown in Figure 9-59 and described in Table 9-80.

Figure 9-59. SDRAM Timing 1 Shadow Register (SDRAM_TIM_1_SHDW)



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

Table 9-80. SDRAM Timing 1 Shadow Register (SDRAM_TIM_1_SHDW) Field Descriptions

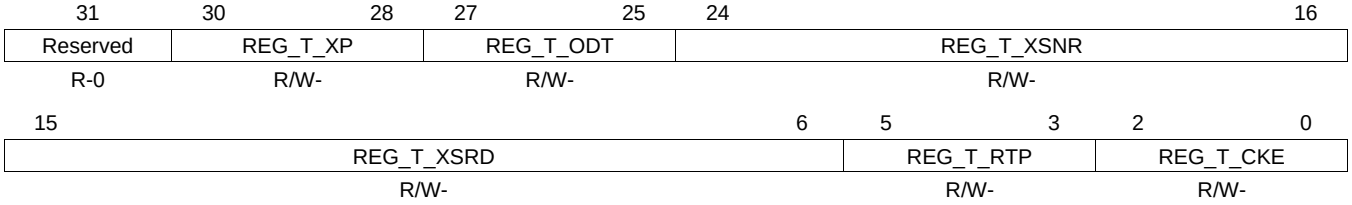
Bit	Field	Value	Description
31-29	Reserved	0	Reserved.
28-25	REG_T_RP_SHDW	0-Fh	Shadow field for reg_t_rp. This field is loaded into reg_t_rp field in SDRAM Timing 1 register when SIdleAck is asserted.
24-21	REG_T_RCD_SHDW	0-Fh	Shadow field for reg_t_rcd. This field is loaded into reg_t_rcd field in SDRAM Timing 1 register when SIdleAck is asserted.
20-17	REG_T_WR_SHDW	0-Fh	Shadow field for reg_t_wr. This field is loaded into reg_t_wr field in SDRAM Timing 1 register when SIdleAck is asserted. Initialization sequence will be started when the value of this field is changed from the previous value and the EMIF is in DDR2 mode.
16-12	REG_T_RAS_SHDW	0-1Fh	Shadow field for reg_t_ras. This field is loaded into reg_t_ras field in SDRAM Timing 1 register when SIdleAck is asserted.
11-6	REG_T_RC_SHDW	0-3Fh	Shadow field for reg_t_rc. This field is loaded into reg_t_rc field in SDRAM Timing 1 register when SIdleAck is asserted.
5-3	REG_T_RRD_SHDW	0-7h	Shadow field for reg_t_rrd. This field is loaded into reg_t_rrd field in SDRAM Timing 1 register when SIdleAck is asserted.
2-0	REG_T_WTR_SHDW	0-7h	Shadow field for reg_t_wtr. This field is loaded into reg_t_wtr field in SDRAM Timing 1 register when SIdleAck is asserted.

9.2.3.4.3.2.8 SDRAM Timing 2 Register (SDRAM_TIM_2)

The SDRAM Timing 2 Register (SDRAM_TIM_2) is shown in Figure 9-60 and described in Table 9-81.

NOTE: If this register is byte written, care must be taken that all the fields are written before performing any accesses to the SDRAM.

Figure 9-60. SDRAM Timing 2 Register (SDRAM_TIM_2)



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

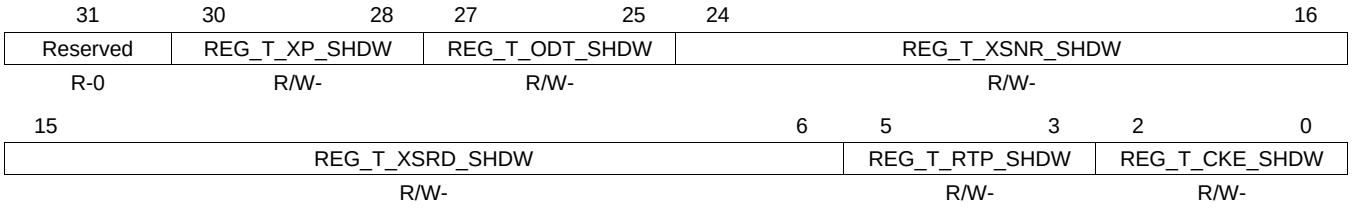
Table 9-81. SDRAM Timing 2 Register (SDRAM_TIM_2) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved.
30-28	REG_T_XP	0-7h	Minimum number of m_clk cycles from Power-Down exit to any command other than aRead command, minus one. For DDR2 and LPDDR1, this field must satisfy greater of tXP or tCKE.
27-25	REG_T_ODT	0-7h	Minimum number of m_clk cycles from ODT enable to write data driven for DDR2.reg_t_odt must be equal to tAOND.
24-16	REG_T_XSNR	0-1FFh	Minimum number of m_clk cycles from Self-Refresh exit to any command other than aRead command, minus one.
15-6	REG_T_XSRD	0-3FFh	Minimum number of m_clk cycles from Self-Refresh exit to a Read command, minus one.
5-3	REG_T_RTP	0-7h	Minimum number of m_clk cycles from the last Read command to a Pre-chargecommand for DDR2, minus one.
2-0	REG_T_CKE	0-7h	Minimum number of m_clk cycles between pad_cke_o changes, minus one.

9.2.3.4.3.2.9 SDRAM Timing 2 Shadow Register (SDRAM_TIM_2_SHDW)

The SDRAM Timing 2 Shadow Register (SDRAM_TIM_2_SHDW) is shown in Figure 9-61 and described in Table 9-82.

Figure 9-61. SDRAM Timing 2 Shadow Register (SDRAM_TIM_2_SHDW)



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

Table 9-82. SDRAM Timing 2 Shadow Register (SDRAM_TIM_2_SHDW) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved.
30-28	REG_T_XP_SHDW	0-7h	Shadow field for reg_t_xp. This field is loaded into reg_t_xp field in SDRAM Timing 2 register when SIdleAck is asserted.
27-25	REG_T_ODT_SHDW	0-7h	Shadow field for reg_t_odt. This field is loaded into reg_t_odt field in SDRAM Timing 2 register when SIdleAck is asserted.
24-16	REG_T_XSNR_SHDW	0-1FFh	Shadow field for reg_t_xsnr. This field is loaded into reg_t_xsnr field in SDRAM Timing 2 register when SIdleAck is asserted.
15-6	REG_T_XSRD_SHDW	0-3FFh	Shadow field for reg_t_xsrd. This field is loaded into reg_t_xsrd field in SDRAM Timing 2 register when SIdleAck is asserted.
5-3	REG_T_RTP_SHDW	0-7h	Shadow field for reg_t_rtp. This field is loaded into reg_t_rtp field in SDRAM Timing 2 register when SIdleAck is asserted.
2-0	REG_T_CKE_SHDW	0-7h	Shadow field for reg_t_cke. This field is loaded into reg_t_cke field in SDRAM Timing 2 register when SIdleAck is asserted.

9.2.3.4.3.2.10 SDRAM Timing 3 Register (SDRAM_TIM_3)

The SDRAM Timing 3 Register (SDRAM_TIM_3) is shown in Figure 9-62 and described in Table 9-83.

NOTE:

1. If this register is byte written, care must be taken that all the fields are written before performing any accesses to the SDRAM.
2. Value for reg_t_ras_max can be calculated as follows:
   - o If tRASmax = 120 us and tREFI = 15.7 us, then reg_t_ras_max = ((120/15.7)-1) = 6.64.
   - o Round down to the next lower integer. Therefore, the programmed value must be 6.

Figure 9-62. SDRAM Timing 3 Register (SDRAM_TIM_3)

31	13	12	4	3	0
Reserved		REG_T_RFC		REG_T_RAS_MAX	
R-0					

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

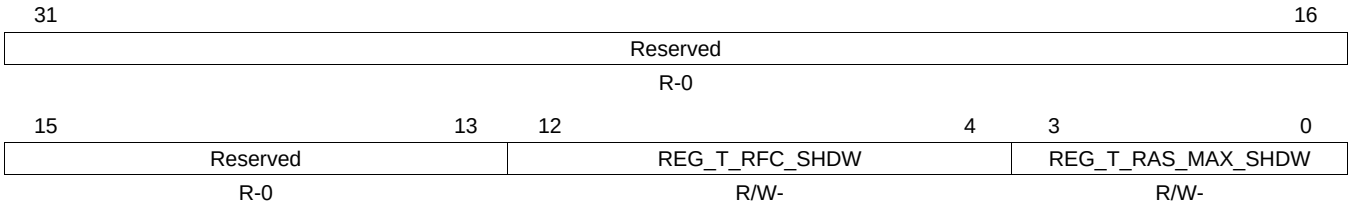
Table 9-83. SDRAM Timing 3 Register (SDRAM_TIM_3) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved.
12-4	REG_T_RFC	0-1FFh	Minimum number of m_clk cycles from Refresh or Load Mode to Refresh or Activate, minus one.
3-0	REG_T_RAS_MAX	0-Fh	Maximum number of reg_refresh_rate intervals from Activate to Precharge command. This field must be equal to ((tRASmax / tREFI)-1) rounded down to the next lower integer.

9.2.3.4.3.2.11 SDRAM Timing 3 Shadow Register (SDRAM_TIM_3_SHDW)

The SDRAM Timing 3 Shadow Register (SDRAM_TIM_3_SHDW) is shown in Figure 9-63 and described in Table 9-84.

Figure 9-63. SDRAM Timing 3 Shadow Register (SDRAM_TIM_3_SHDW)



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

Table 9-84. ‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘

Bit	Field	Value	Description
31-13	Reserved	0	Reserved.
12-4	REG_T_RFC_SHDW	0-1FFh	Shadow field for reg_t_rfc. This field is loaded into reg_t_rfc field in SDRAM Timing 3 register when SIdleAck is asserted.
3-0	REG_T_RAS_MAX_SHDW	0-Fh	Shadow field for reg_t_ras_max. This field is loaded into reg_t_ras_max field in SDRAM Timing 3 register when SIdleAck is asserted.

9.2.3.4.3.2.12 Power Management Control Register (PWR_MGMT_CTRL)

The Power Management Control Register (PWR_MGMT_CTRL) is shown in Figure 9-64 and described in Table 9-85.

Figure 9-64. Power Management Control Register (PWR_MGMT_CTRL)

31 30	29 16
REG_IDLEMODE	Reserved
R/W-0x2	R-0

15 11	10	9 8	7 4	3 0
Reserved	REG_DPD_EN	REG_LP_MODE	Reserved	REG_PM_TIM
R-0	R/W-0	R/W-0	R-0	R/W-0

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

Table 9-85. Power Management Control Register (PWR_MGMT_CTRL) Field Descriptions

Bit	Field	Value	Description
31-30	REG_IDLEMODE	0-3h	Power-Idle IP Generic mode. Set to 0 for force-idle mo Power-Idle IP Generic mode. Set to 0 for force-idle mode, where EMIF acknowledges the system's idle requests unconditionally, i.e. regardless of the EMIF's internal requirements. Backup mode, for debug only. Set to 1 for no-idle mode, where EMIF will never enter idle state. Backup mode, for debug only. Set to 2 for smart-idle mode, where EMIF's idle state eventually follows (acknowledges) the system's idle requests, depending on EMIF's internal requirements. Setting of 3 (smart-idle wakeup-capable mode) is not supported.
31-11	Reserved	0	Reserved.
10	REG_DPD_EN	0-1h	Deep Power-Down enable. Set to 0 for normal operation. Set to 1 to enter Deep Power-Down mode. This mode will override the reg_lp_mode field setting.
9-8	REG_LP_MODE	0-3h	Automatic Power Management enable. Set to 1 for Clock Stop, set to 2 for Self Refresh, and set to 3 for Power-Down. Set to 0 to disable automatic power management. Reserved for future use.
7-4	Reserved	0	Reserved.
3-0	REG_PM_TIM	0-Fh	Power Management timer. The EMIF will put the external SDRAM in a power saving mode after the EMIF is idle for these number of m_clk cycles. The power saving mode is chosen by the reg_lp_mode field. Set to 0 to immediately enter the power savings mode. Set to 1 for 16 clocks, set to 2 for 32 clocks, set to 3 for 64 clocks, set to 4 for 128 clocks, set to 5 for 256 clocks, set to 6 for 512 clocks, set to 7 for 1024 clocks, set to 8 for 2048 clocks, set to 9 for 4096 clocks, set to 10 for 8192 clocks, set to 11 for 16384 clocks, set to 12 for 32768 clocks, set to 13 for 65536 clocks, set to 14 for 131072 clocks, and set to 15 for 262144 clocks.

9.2.3.4.3.2.13 Power Management Control Shadow Register (PWR_MGMT_CTRL_SHDW)

The Power Management Control Shadow Register (PWR_MGMT_CTRL_SHDW) is shown in Figure 9-65 and described in Table 9-86.

Figure 9-65. Power Management Control Shadow Register (PWR_MGMT_CTRL_SHDW)

31	4	3	0
Reserved		REG_PM_TIM_SHDW	
R-0		R/W-0	

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

Table 9-86. Power Management Control Shadow Register (PWR_MGMT_CTRL_SHDW) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3-0	REG_PM_TIM_SHDW	0-Fh	Shadow field for reg_pm_tim. This field is loaded into reg_pm_tim field in Power Management Control register when SIdleAck is asserted.

9.2.3.4.3.2.14 OCP Configuration Register (OCP_CONFIG)

The OCP Configuration Register (OCP_CONFIG) is shown in Figure 9-66 and described in Table 9-87.

Figure 9-66. OCP Configuration Register (OCP_CONFIG)

31	8	7	0
Reserved		REG_PR_OLD_COUNT	
R-0		R/W-0xFF	

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

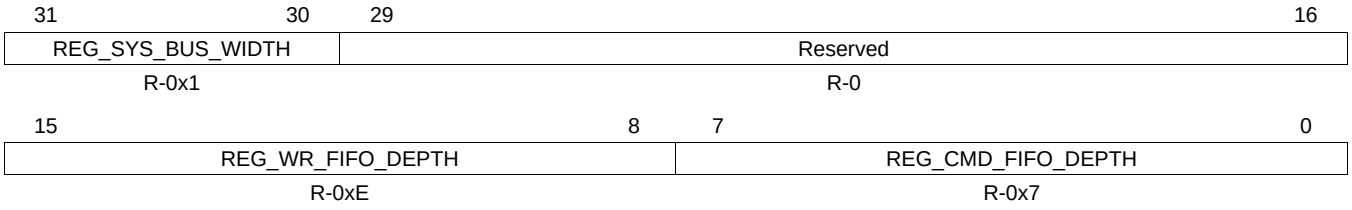
Table 9-87. OCP Configuration Register (OCP_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	REG_PR_OLD_COUNT	0-FFh	Priority Raise Old Counter. Number of memory transfers after which the EMIF momentarily raises the priority of old commands in the OCP Command FIFO.

9.2.3.4.3.2.15 OCP Configuration Value 1 Register (OCP_CFG_VAL_1)

The CP Configuration Value 1 Register (OCP_CFG_VAL_1) is shown in Figure 9-67 and described in Table 9-88.

Figure 9-67. OCP Configuration Value 1 Register (OCP_CFG_VAL_1)



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

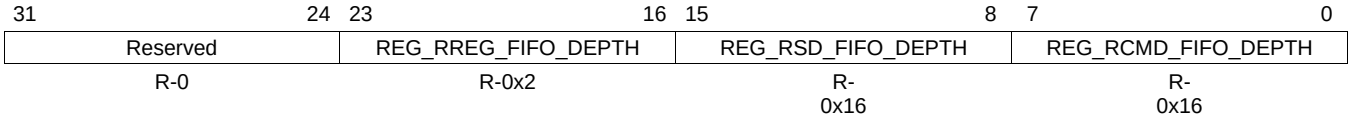
Table 9-88. OCP Configuration Value 1 Register (OCP_CFG_VAL_1) Field Descriptions

Bit	Field	Value	Description
31-30	REG_SYS_BUS_WIDTH	0-3h	L3 OCP data bus width for a particular configuration. 0 = 32-bit wide, 1 = 64-bit wide, 2 = 128-bit wide, and 3 = Reserved
29-16	Reserved	0	Reserved
15-8	REG_WR_FIFO_DEPTH	0-FFh	Write Data FIFO depth for a particular configuration.
7-0	REG_CMD_FIFO_DEPTH	0-FFh	Command FIFO depth for a particular configuration.

9.2.3.4.3.2.16 OCP Configuration Value 2 Register (OCP_CFG_VAL_2)

The OCP Configuration Value 2 Register (OCP_CFG_VAL_2) is shown in Figure 9-68 and described in Table 9-89.

Figure 9-68. OCP Configuration Value 2 Register (OCP_CFG_VAL_2)



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

Table 9-89. OCP Configuration Value 2 Register (OCP_CFG_VAL_2) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	REG_RREG_FIFO_DEPTH	0-FFh	Register Read Data FIFO depth for a particular configuration.
15-8	REG_RSD_FIFO_DEPTH	0-FFh	SDRAM Read Data FIFO depth for a particular configuration.
7-0	REG_RCMD_FIFO_DEPTH	0-FFh	Read Command FIFO depth for a particular configuration.

9.2.3.4.3.2.17 IODFT Test Logic Global Control Register (IODFT_TLGC)

The IODFT Test Logic Global Control Register (IODFT_TLGC) is shown in Figure 9-69 and described in Table 9-90.

Figure 9-69. IODFT Test Logic Global Control Register (IODFT_TLGC)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_TLEC															
R/W-0															

15	14	13	12	11	10	9	8
Reserved	REG_MT	REG_ACT_CAP_EN	REG_OPG_LD	Reserved	REG_RESET_PHY	Reserved	REG_MMS
R-0	R/W-0	R/W-1	R/W-0	R-0	R/W-0	R-0	R/W-0

7	6	5	4	3	2	1	0
Reserved		REG_MC		REG_PC			REG_TM
R-0		R/W-1		R/W-0			R/W-1

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

Table 9-90. IODFT Test Logic Global Control Register (IODFT_TLGC) Field Descriptions

Bit	Field	Value	Description
31-16	REG_TLEC	0-FFFFh	IODFT Test Logic Execution Counter. Contains the number of cycle that the MISR signature will be accumulated. Upon the expiration of the counter the MISR capture will be turned off.
15	Reserved	0	Reserved
14	REG_MT	0-1h	MISR on/off trigger command. 0x0 = inactive/no affect. 0x1 = MISR capture start on the first write or read command to the memory and continues to update the signature until reg_tlec expires, when mc = 0x3. 0x1 = pattern generator starts on the first write or read command to the memory and continues to update the signature until reg_tlec expires, when pc = 0x1, 0x2, 0x3, 0x5, 0x6, or 0x7. These bits are cleared when reg_tlec expires.
13	REG_ACT_CAP_EN	0-1h	Active cycles capture enable. If set to a 1 the MISRs and pattern generators will shift only during active cycles. If set to a 0 the MISRs and pattern generators will shift every clock cycle.
12	REG_OPG_LD	0-1h	Load pattern generators' initial value. Set to 1 to load an initial value in the pattern generators from reg_tlec.
11	Reserved	0	Reserved.
10	REG_RESET_PHY	0-1h	Reset DDR PHY. Writing a 1 to this bit will reset the DDR PHY. This bit will self clear to 0.
9	Reserved	0	Reserved.
8	REG_MMS	0-1h	Chooses the source of the MISR input. Set to 0 for output register, and set to 1 for input capture.
7-6	Reserved	0	Reserved.
5-4	REG_MC	0-3h	MISR state. Set to 0 to download results. Set to 1 to hold current value. Set to 2 to load initial value from pc bits. Set to 3 to enable MISR to capture signature.
3-1	REG_PC	0-7h	Pattern code. Defines the type of pattern that is selected for the pattern generators. Set to 0 for functional mode and set to 4 to hold current register value. Set to 5 for random XOR, set to 6 for random XNOR, and set to 7 for an 8 bit shifter. All other values are reserved.
0	REG_TM	0-1h	Functional mode enable. Set to 1 for functional mode, and set to 0 for IODFT mode.

9.2.3.4.3.2.18 IODFT Test Logic Control MISR Result Register (IODFT_CTRL_MISR_RSLT)

The IODFT Test Logic Control MISR Result Register is shown in Figure 9-70 and described in Table 9-91.

Figure 9-70. IODFT Test Logic Control MISR Result Register (IODFT_CTRL_MISR_RSLT)

31 26	25 16	15 11	10 0
Reserved	REG_DQM_TLMR	Reserved	REG_CTL_TLMR
R-0	R-0	R-0	R-0

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

Table 9-91. IODFT Test Logic Control MISR Result Register (IODFT_CTRL_MISR_RSLT) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved.
25-16	REG_DQM_TLMR	0-3FFh	This contains the MISR result signature of a given test after the download function is executed. This result is for the control signals. Reserved for future use.
15-11	Reserved	0	Reserved.
10-0	REG_CTL_TLMR	0-7FFh	This contains the MISR result signature of a given test after the download function is executed. This result is for the control signals.

9.2.3.4.3.2.19 IODFT Test Logic Address MISR Result Register (IODFT_ADDR_MISR_RSLT)

The IODFT Test Logic Address MISR Result Register (IODFT_ADDR_MISR_RSLT) is shown in Figure 9-71 and described in Table 9-92.

Figure 9-71. IODFT Test Logic Address MISR Result Register (IODFT_ADDR_MISR_RSLT)

31 21	20 0
Reserved	REG_ADDR_TLMR
R-0	R-0

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

Table 9-92. IODFT Test Logic Address MISR Result Register (IODFT_ADDR_MISR_RSLT) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved.
20-0	REG_ADDR_TLMR	0-1F FFFFh	This contains the MISR result signature of a given test after the download function is executed. This result is for the address signals.

9.2.3.4.3.2.20 IODFT Test Logic Data MISR Result 1 Register (IODFT_DATA_MISR_RSLT_1)

The IODFT Test Logic Data MISR Result 1 Register (IODFT_DATA_MISR_RSLT_1) is shown in Figure 9-72 and described in Table 9-93.

Figure 9-72. IODFT Test Logic Data MISR Result 1 Register (IODFT_DATA_MISR_RSLT_1)

31	0
REG_DATA_TLMR_31_0	
R-0	

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

Table 9-93. IODFT Test Logic Data MISR Result 1 Register (IODFT_DATA_MISR_RSLT_1) Field Descriptions

Bit	Field	Value	Description
31-0	REG_DATA_TLMR_31_0	0-FFFF FFFFh	This contains the least significant bits of the MISR result signature of a given test after the download function is executed. This result is for data bus.

9.2.3.4.3.2.21 IODFT Test Logic Data MISR Result 2 Register (IODFT_DATA_MISR_RSLT_2)

The IODFT Test Logic Data MISR Result 2 Register (IODFT_DATA_MISR_RSLT_2) is shown in Figure 9-73 and described in Table 9-94.

Figure 9-73. IODFT Test Logic Data MISR Result 2 Register (IODFT_DATA_MISR_RSLT_2)

31	0
REG_DATA_TLMR_63_32	
R-0	

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

Table 9-94. IODFT Test Logic Data MISR Result 2 Register (IODFT_DATA_MISR_RSLT_2) Field Descriptions

Bit	Field	Value	Description
31-0	REG_DATA_TLMR_63_32	0-FFFF FFFFh	This contains the middle bits of the MISR result signature of a given test after the download function is executed. This result is for data bus.

9.2.3.4.3.2.22 IODFT Test Logic Data MISR Result 3 Register (IODFT_DATA_MISR_RSLT_3)

The IODFT Test Logic Data MISR Result 3 Register (IODFT_DATA_MISR_RSLT_3) is shown in Figure 9-74 and described in Table 9-95.

Figure 9-74. IODFT Test Logic Data MISR Result 3 Register (IODFT_DATA_MISR_RSLT_3)

31	3	2	0
Reserved		REG_DATA_TLMR_66_64	
R-0		R-0	

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

Table 9-95. IODFT Test Logic Data MISR Result 3 Register (IODFT_DATA_MISR_RSLT_3) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2-0	REG_DATA_TLMR_66_64	0-7h	This contains the most significant bits of the MISR result signature of a given test after the download function is executed. This result is for data bus.

9.2.3.4.3.2.23 Performance Counter 1 Register (PERF_CNT_1)

The Performance Counter 1 Register (PERF_CNT_1) is shown in Figure 9-75 and described in Table 9-96.

Figure 9-75. Performance Counter 1 Register (PERF_CNT_1)

31	0
REG_COUNTER1	
R-0	

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

Table 9-96. Performance Counter 1 Register (PERF_CNT_1) Field Descriptions

Bit	Field	Value	Description
31-0	REG_COUNTER1	0-FFFF FFFFh	32-bit counter that can be configured as specified in the Performance Counter Configuration Register and Performance Counter Master Region Select Register.

9.2.3.4.3.2.24 Performance Counter 2 Register (PERF_CNT_2)

The Performance Counter 2 Register (PERF_CNT_2) is shown in Figure 9-76 and described in Table 9-97.

Figure 9-76. Performance Counter 2 Register (PERF_CNT_2)

31	0
REG_COUNTER2	
R-0	

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

Table 9-97. Performance Counter 2 Register (PERF_CNT_2) Field Descriptions

Bit	Field	Value	Description
31-0	REG_COUNTER2	0-FFFF FFFFh	32-bit counter that can be configured as specified in the Performance Counter Configuration Register and Performance Counter Master Region Select Register.

9.2.3.4.3.2.25 Performance Counter Configuration Register (PERF_CNT_CFG)

The Performance Counter Configuration Register (PERF_CNT_CFG) is shown in Figure 9-77 and described in Table 9-98.

Figure 9-77. Performance Counter Configuration Register (PERF_CNT_CFG)

31	30	29 24
REG_CNTR2_MCONNID_EN	REG_CNTR2_REGION_EN	Reserved
R/W-0	R/W-0	R-0

23 20	19 16
Reserved	REG_CNTR2_CFG
R-0	R/W-0

15	14	13 8
REG_CNTR1_MCONNID_EN	REG_CNTR1_REGION_EN	Reserved
R/W-0	R/W-0	R-0

7 4	3 0
Reserved	REG_CNTR1_CFG
R-0	R/W-0

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

Table 9-98. Performance Counter Configuration Register (PERF_CNT_CFG) Field Descriptions

Bit	Field	Value	Description
31	REG_CNTR2_MCONNID_EN	0-1h	MConnID filter enable for Performance Counter 2 register.
30	REG_CNTR2_REGION_EN	0-1h	Chip Select filter enable for Performance Counter 2 register.
29-20	Reserved	0	Reserved.
19-16	REG_CNTR2_CFG	0-Fh	Filter configuration for Performance Counter 2. Refer to Table 9-99 for details.
15	REG_CNTR1_MCONNID_EN	0-1h	MConnID filter enable for Performance Counter 1 register.
14	REG_CNTR1_REGION_EN	0-1h	Chip Select filter enable for Performance Counter 1 register.
13-4	Reserved	0	Reserved.
3-0	REG_CNTR1_CFG	0-Fh	Filter configuration for Performance Counter 1. Refer to Table 9-99 for details.

Table 9-99 shows the possible filter configurations for the two performance counters. These filter configurations can be used in conjunction with an OCP master ID and/or an external chip select to obtain performance statistics for a particular OCP master and/or an external chip select.

Table 9-99. Performance Counter Filter Configuration

cntrN_cfg	cntrN_region_en	cntrN_mconnid_en	Description
0x0	0x0	0x0 or 0x1	Count total SDRAM accesses.
0x1	0x0	0x0 or 0x1	Count total SDRAM activates.
0x2	0x0 or 0x1	0x0 or 0x1	Count total reads.
0x3	0x0 or 0x1	0x0 or 0x1	Count total writes.
0x4	0x0	0x0	Count number of m_clk cycles OCP Command FIFO is full.
0x5	0x0	0x0	Count number of m_clk cycles OCP Write Data FIFO is full.
0x6	0x0	0x0	Count number of m_clk cycles OCP Read Data FIFO is full.
0x7	0x0	0x0	Count number of m_clk cycles OCP Return Command FIFO is full.
0x8	0x0 or 0x1	0x0 or 0x1	Count number of priority elevations.
0x9	0x0	0x0	Count number of m_clk cycles that a command was pending.
0xA	0x0	0x0	Count number of m_clk cycles for which the memory data bus was transferring data.
0xB - 0xF	0x0	0x0	Reserved for future use.

NOTE: When MReqDebug is set to a 1 for a particular OCP command, the performance counters will not be incremented for that particular command if the cntrN_cfg values are equal 0x0, 0x1, 0x2, 0x3, or 0xA.

9.2.3.4.3.2.26 Performance Counter Master Region Select Register (PERF_CNT_SEL)

The Performance Counter Master Region Select Register (PERF_CNT_SEL) is shown in Figure 9-78 and described in Table 9-100.

Figure 9-78. Performance Counter Master Region Select Register (PERF_CNT_SEL)

31	24
REG_MCONNID2	
R/W-0	

23	18	17	16
Reserved		REG_REGION_SEL2	
R-0		R/W-0	

15	8
REG_MCONNID1	
R/W-0	

7	2	1	0
Reserved		REG_REGION_SEL1	
R-0		R/W-0	

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

Table 9-100. Performance Counter Master Region Select Register (PERF_CNT_SEL) Field Descriptions

Bit	Field	Value	Description
31-24	REG_MCONNID2	0-FFh	MConnID for Performance Counter 2 register.
23-18	Reserved	0	Reserved.
17-16	REG_REGION_SEL2	0-3h	MAddrSpace for Performance Counter 2 register.
15-8	REG_MCONNID1	0-FFh	MConnID for Performance Counter 1 register.
7-2	Reserved	0	Reserved.
1-0	REG_REGION_SEL1	0-3h	MAddrSpace for Performance Counter 1 register.

9.2.3.4.3.2.27 Performance Counter Time Register (PERF_CNT_TIM)

The Performance Counter Time Register (PERF_CNT_TIM) is shown in Figure 9-79 and described in Table 9-101.

NOTE: This is a free running counter and is not affected by MReqDebug value.

Figure 9-79. Performance Counter Time Register (PERF_CNT_TIM)

31	0
REG_TOTAL_TIME	
R-0	

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

Table 9-101. Performance Counter Time Register (PERF_CNT_TIM) Field Descriptions

Bit	Field	Value	Description
31-0	REG_TOTAL_TIME	0-FFFF FFFFh	32-bit counter that continuously counts number for m_clk cycles elapsed after EMIF is brought out of reset.

9.2.3.4.3.2.28 End of Interrupt Register (IRQ_EOI)

The End of Interrupt Register (IRQ_EOI) is shown in Figure 9-80 and described in Table 9-102.

Figure 9-80. End of Interrupt Register (IRQ_EOI)

31	1	0
Reserved		REG_EOI
R-0		R/W-0

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

Table 9-102. End of Interrupt Register (IRQ_EOI) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	REG_EOI		Software End Of Interrupt (EOI) control. Write 0 for system OCP interrupt. This field always reads 0 (no EOI memory).

9.2.3.4.3.2.29 System OCP Interrupt Raw Status Register (IRQSTATUS_RAW_SYS)

The System OCP Interrupt Raw Status Register (IRQSTATUS_RAW_SYS) is shown in Figure 9-81 and described in Table 9-103.

Figure 9-81. System OCP Interrupt Raw Status Register (IRQSTATUS_RAW_SYS)

31	1	0
Reserved		REG_RAW_SYS
R-0		WOS-0

LEGEND: WOS = Write one to set field; R = Read only; *-n* = value after reset

Table 9-103. System OCP Interrupt Raw Status Register (IRQSTATUS_RAW_SYS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	REG_RAW_SYS		Raw status of system OCP interrupt. Write 1 to set the (raw) status, mostly for debug. Writing a 0 has no effect.

9.2.3.4.3.2.30 System OCP Interrupt Status Register (IRQSTATUS_SYS)

The System OCP Interrupt Status Register (IRQSTATUS_SYS) is shown in Figure 9-82 and described in Table 9-104.

Figure 9-82. System OCP Interrupt Status Register (IRQSTATUS_SYS)

31	1	0
Reserved		REG_ENABLED_SYS
R-0		WOC-0

LEGEND: WOC = Write one to clear field; R = Read only; *-n* = value after reset

Table 9-104. System OCP Interrupt Status Register (IRQSTATUS_SYS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	REG_ENABLED_SYS		Enabled status of system OCP interrupt. Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, i.e. even if not enabled). Writing a 0 has no effect.

9.2.3.4.3.2.31 System OCP Interrupt Enable Set Register (IRQENABLE_SET_SYS)

The System OCP Interrupt Enable Set Register (IRQENABLE_SET_SYS) is shown in Figure 9-83 and described in Table 9-105.

Figure 9-83. System OCP Interrupt Enable Set Register (IRQENABLE_SET_SYS)

31	1	0
Reserved		REG_EN_SYS
R-0		WOS-0

LEGEND: WOS = Write one to set field; R = Read only; -*n* = value after reset

Table 9-105. System OCP Interrupt Enable Set Register (IRQENABLE_SET_SYS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	REG_EN_SYS		Enable set for system OCP interrupt. Writing a 1 will enable the interrupt, and set this bit as well as the corresponding Interrupt Enable Clear Register. Writing a 0 has no effect.

9.2.3.4.3.2.32 System OCP Interrupt Enable Clear Register (IRQENABLE_CLR_SYS)

The System OCP Interrupt Enable Clear Register (IRQENABLE_CLR_SYS) is shown in Figure 9-84 and described in Table 9-106.

Figure 9-84. System OCP Interrupt Enable Clear Register (IRQENABLE_CLR_SYS)

31	1	0
Reserved		REG_EN_SYS
R-0		WOC-0

LEGEND: WOC = Write one to clear field; R = Read only; -*n* = value after reset

Table 9-106. System OCP Interrupt Enable Clear Register (IRQENABLE_CLR_SYS) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved.
0	REG_EN_SYS		Enable clear for system OCP interrupt. Writing a 1 will disable the interrupt, and clear this bit as well as the corresponding Interrupt Enable Set Register. Writing a 0 has no effect.

9.2.3.4.3.2.33 OCP Error Log Register (OCP_ERR_LOG)

The OCP Error Log Register (OCP_ERR_LOG) is shown in Figure 9-85 and described in Table 9-107.

Figure 9-85. OCP Error Log Register (OCP_ERR_LOG)

31	16
Reserved	
R-0	

15	14	13	11	10	8
REG_MADDRSPACE		REG_MBURSTSEQ		REG_MCMD	
R-0		R-0		R-0	

7	0
REG_MCONNID	
R-0	

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

Table 9-107. OCP Error Log Register (OCP_ERR_LOG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved.
15-14	REG_MADDRSPACE	0-3h	Address space of the first errored transaction.
13-11	REG_MBURSTSEQ	0-7h	Addressing mode of the first errored transaction.
10-8	REG_MCMD	0-7h	Command type of the first errored transaction.
7-0	REG_MCONNID	0-FFh	Connection ID of the first errored transaction.

9.2.3.4.3.2.34 DDR PHY Control 1 Register (DDR_PHY_CTRL_1)

The DDR PHY Control 1 Register (DDR_PHY_CTRL_1) is shown in Figure 9-86 and described in Table 9-108.

Figure 9-86. DDR PHY Control 1 Register (DDR_PHY_CTRL_1)

31	24
Reserved	
R-0	

23	22	16
TESTIN_LB_CK_SELECT	Reserved	
R/W-0	R-0	

15	14 12	11 9	8
CONFIG_VTP_DYNAMIC_UPDATE	CONFIG_DLL_MODE	Reserved	DDR_16B_MODE_PWRSAVE
R/W-0	R/W-0	R-0	R/W-0

7	6	5 3	2 0
CONFIG_EXT_STRBEN	CONFIG_PWRDN_DISABLE	Reserved	READ_LATENCY
R/W-0	R/W-0	R-0	R/W-0

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

Table 9-108. DDR PHY Control 1 Register (DDR_PHY_CTRL_1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23	TESTIN_LB_CK_SELECT	0	This bit controls the value assigned to the TESTIN_LB_CK_SELECT input on the command macro.
22-16	Reserved	0	Reserved
15	CONFIG_VTP_DYNAMIC_UPDATE	0	This bit controls the value assigned to the CONFIG_VTP_DYNAMIC_UPDATE input on the command macro.
14-12	CONFIG_DLL_MODE	0	This bit controls the value assigned to the CONFIG_DLL_MODE input on the data macros.
11-9	Reserved	0	Reserved
8	DDR_16B_MODE_PWRSAVE	0	This bit can be set to gate the clock of second data macro when DDR is used in 16b mode.
7	CONFIG_EXT_STRBEN	1h	This bit controls the value assigned to the CONFIG_EXT_STRBEN input on the DDR PHY data macro.
6	CONFIG_PWRDN_DISABLE	0	This bit controls the value assigned to the CONFIG_PWRDNEN input on the DDR PHY data macro. 0: DDR receive IO buffer will be powered down (disabled) when there no read. 1: DDR receive IO buffer will be always enabled.
5-3	Reserved	0	Reserved
2-0	READ_LATENCY	3h	This field defines the latency for read data from DDR SDRAM in number of X1_CLK_OUT cycles. The value applied should be equal to the required value minus one. The maximum read latency supported by the DDR PHY is equal to CAS latency plus 2 X1_CLK_OUT CYCLES. The minimum read latency must be equal to CAS latency plus 1 X1_CLK_OUT CYCLES.

9.2.3.4.3.2.35 DDR PHY Control 1 Shadow Register (DDR_PHY_CTRL_1_SHDW)

The DDR PHY Control 1 Shadow Register (DDR_PHY_CTRL_1_SHDW) is shown in Figure 9-87 and described in Table 9-109.

Figure 9-87. DDR PHY Control 1 Shadow Register (DDR_PHY_CTRL_1_SHDW)

31	3	2	0
REG_DDR_PHY_CTRL_1_SHDW		REG_READ_LATENCY_SHDW	
R/W-0		R/W-0	

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

Table 9-109. DDR PHY Control 1 Shadow Register (DDR_PHY_CTRL_1_SHDW) Field Descriptions

Bit	Field	Value	Description
31-3	REG_DDR_PHY_CTRL_1_SHDW	0-1FFF FFFFh	Shadow field for REG_DDR_PHY_CTRL_1. This field is loaded into REG_DDR_PHY_CTRL_1 field in DDR PHY Control 1 register when SIdleAck is asserted.
2-0	REG_READ_LATENCY_SHDW	0-7h	Shadow field for REG_READ_LATENCY. This field is loaded into REG_READ_LATENCY field in DDR PHY Control 1 register when SIdleAck is asserted.

9.2.3.4.3.2.36 DDR PHY Control 2 Register (DDR_PHY_CTRL_2)

The DDR PHY Control 2 Register (DDR_PHY_CTRL_2) is shown in Figure 9-88 and described in Table 9-110.

Figure 9-88. DDR PHY Control 2 Register (DDR_PHY_CTRL_2)

31	2	1	0
Reserved		CONFIG_RX_DLL_BYPASS	CONFIG_TX_STRB_DATA_ALIGN
R-0		R/W-0	R/W-0

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

Table 9-110. DDR PHY Control 2 Register (DDR_PHY_CTRL_2) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	CONFIG_RX_DLL_BYPASS	0	This bit controls the value assigned to the CONFIG_RX_DLL_BYPASS input on the DDR PHY data macro.
0	CONFIG_TX_STRB_DATA_ALIGN	0	This bit controls the value assigned to the CONFIG_TX_STRB_DATA_ALIGN input on the DDR PHY data macro.

9.2.3.4.4 Interrupt Conditions

This section defines the module interrupt capabilities and requirements.

9.2.3.4.4.1 Interrupt Description

The EMIF sources one interrupt for the system OCP. The system OCP interrupts are sys_err_intr_req (pulse) and sys_err_intr_pend (level). The pulse and level interrupt signals for a particular interface are both sourced from the same raw interrupt for the corresponding interface internal to the module.

9.2.3.4.4.2 Interrupt Condition Control

The EMIF only supports Idle, Write, Read, and WriteNonPost command types (as indicated by MCmd). Also, the EMIF only supports incrementing, wrapping, and 2-dimensional block addressing modes (as indicated by MBurstSeq). If an access request for an unsupported command type or an unsupported addressing mode is received, the EMIF will set the reg_raw_sys bit in the Interrupt Raw Status register. The EMIF will also set the reg_raw_sys bit in Interrupt Raw Status register, if an access request to an unsupported MAddrSpace is received.

The EMIF will only output the interrupts on the interrupt lines if they are enabled by writing a 1 to the corresponding bits in the Interrupt Enable Set register. The interrupts can be disabled by writing a 1 to the corresponding bits in the Interrupt Enable Clear register.

When enabled, the corresponding bits in the Interrupt Status register will also be set if the above error condition occurs. The interrupts can be cleared once serviced by writing a 1 to the corresponding bits either in the Interrupt Raw Status or Interrupt Status register. The software must also write to the End of Interrupt register to indicate that the interrupt was serviced.

9.2.3.4.5 Power Management

This section defines the module Power Management capabilities and requirements.

9.2.3.4.5.1 SDRAM Self-Refresh Mode

The EMIF supports self-refresh mode for low power. The EMIF automatically puts the SDRAM into self-refresh after the EMIF is idle for reg_pm_tim number of m_clk cycles and the reg_lp_mode field is set to 2. The reg_lp_mode and reg_pm_tim fields can be programmed in the Power Management Control register. The EMIF will complete all pending refreshes before it puts the SDRAM into self-refresh. Therefore, after the expiration of reg_pm_tim, the EMIF will start issuing refreshes to complete the refresh backlog, and then issue a SELF REFRESH command to the SDRAM.

In self-refresh mode, the EMIF automatically stops the clocks to the SDRAM, by asserting the phy_sdramclkstop port. The EMIF maintains pad_cke_o low to maintain the self-refresh state.

When the SDRAM is in self-refresh, the EMIF services register accesses as normal.

If the reg_lp_mode field is set not equal to 2, or an SDRAM access is requested while it is in self-refresh, and reg_t_cke + 1 cycles have elapsed since the SELF-REFRESH command was issued, the EMIF will bring the SDRAM out of self-refresh. The value of reg_t_cke is taken from SDRAM Timing 2 register.

For DDR2 the EMIF:

- Drives phy_sdramclkstop low to enable clocks.
- Drives pad_cke_o high.
- Waits for reg_t_xsnr + 1 cycles. The value of reg_t_xsnr is taken from SDRAM Timing 2 register.
- If the reg_ddr_disable_dll bit in the SDRAM Configuration register is 1, issues a LOAD MODE REGISTER command to the extended mode register 1 (pad_ba_o[2:0] = 0x1) with the pad_a_o bits set as follows:

Bits	Value	Description
pad_a_o[15:13]	0x0	Reserved
pad_a_o[12]	0x0	Output buffer enabled
pad_a_o[11]	0x0	RDQS disable
pad_a_o[10]	!reg_ddr2_ddqs	Differential DQS enable value from SDRAM Configuration register
pad_a_o[9:7]	0x0	Exit OCD calibration mode
pad_a_o[6]	reg_ddr_term[1]	DDR2 termination resistor value from SDRAM Configuration register
pad_a_o[5:3]	0x0	Additive latency = 0
pad_a_o[2]	reg_ddr_term[0]	DDR2 termination resistor value from SDRAM Configuration register
pad_a_o[1]	reg_sdram_drive[0]	SDRAM drive strength from SDRAM Configuration register
pad_a_o[0]	0x1	Disable DLL

- Starts an auto-refresh cycle in the next cycle.
- Enters its idle state and can issue any other commands except a write or a read. A write or a read will only be issued after reg_t_xsrd + 1 clock cycles have elapsed since pad_cke_o is driven high. The value of reg_t_xsrd is taken from SDRAM Timing 2 register.

To use partial array self-refresh, reg_pasr bits in the SDRAM Refresh Control register must be appropriately programmed. The EMIF performs bank interleaving when reg_ibank_pos = 0 in SDRAM Configuration register. Since the SDRAM is partially refreshed during partial array self-refresh, for software ease, it is recommended that reg_ibank_pos be set to 1, 2, or 3 depending on the scheme used. If reg_ibank_pos is set to 0, it is the responsibility of software to move critical data into the banks that are going to be refreshed during partial array self-refresh.

For LPDDR1 the EMIF:

- Drives phy_sdramclkstop low to enable clocks.
- Drives pad_cke_o high.
- Waits for reg_t_xsnr + 1 cycles. The value of reg_t_xsnr is taken from SDRAM Timing 2 register.
- Starts an auto-refresh cycle in the next cycle.
- Enters its idle state and can issue any commands.

9.2.3.4.5.2 SDRAM Power-Down Mode

The EMIF also supports Power-Down mode for low power. The EMIF automatically puts the SDRAM into Power-Down after the EMIF is idle for reg_pm_tim number of m_clk cycles and the reg_lp_mode field is set to 3. The reg_lp_mode and reg_pm_tim fields can be programmed in the Power Management Control register. The EMIF will complete all pending refreshes before it puts the SDRAM into Power-Down. Therefore, after the expiration of reg_pm_tim, the EMIF will start issuing refreshes to complete the refresh backlog, and then issue a POWER-DOWN command to the SDRAM.

In Power-Down mode, the EMIF does not stop the clocks to the SDRAM. The EMIF maintains pad_cke_o low to maintain the Power-Down state.

When the SDRAM is in Power-Down, the EMIF services register accesses as normal.

If the reg_lp_mode field is set not equal to 3, or an SDRAM access is requested, or the Refresh Must level is reached while the SDRAM is in Power-Down, the EMIF will bring the SDRAM out of Power-Down.

For DDR2 the EMIF:

- Drives pad_cke_o high after reg_t_cke + 1 cycles have elapsed since the POWER-DOWN command was issued. The value of reg_t_cke is taken from SDRAM Timing 2 register.
- Waits for reg_t_xp + 1 cycles. The value of reg_t_xp is taken from SDRAM Timing 2 register.
- Enters its idle state and can issue any commands.

For LPDDR1 the EMIF:

- Drives pad_cke_o high.

- Waits for reg_t_xp + 1 cycles. The value of reg_t_xp is taken from SDRAM Timing 2 register.
- Enters its idle state and can issue any commands.

9.2.3.4.5.3 SDRAM Deep Power-Down Mode

For ultimate power savings, the EMIF supports Deep Power-Down mode for LPDDR1.

The SDRAM can be forced into Deep Power-Down through software by setting the reg_dpd_en field in the Power Management Control register to 1. In this case, the EMIF will continue normal operation until all SDRAM memory access requests have been serviced. At this point the EMIF will issue a DEEP POWER-DOWN command. The EMIF then maintains pad_cke_o low to maintain the Deep Power-Down state. In Deep Power-Down mode, the EMIF automatically stops the clocks to the SDRAM, by asserting the phy_sdramclkstop port.

Setting the reg_dpd_en field to 1 overrides the setting of reg_lp_mode field. Therefore, if the SDRAM is in Clock Stop, Self Refresh, or Power-Down mode, and reg_dpd_en field is set to 1, the EMIF will exit those modes and go into Deep Power-Down mode.

When the SDRAM is in Deep Power-Down, the EMIF services register accesses as normal. If the reg_dpd_en field is set to 0, or an SDRAM access is requested, the EMIF will bring the SDRAM out of Deep Power-Down.

The EMIF:

- Performs SDRAM initialization as specified in the LPDDR1 Initialization section.
- Enters its idle state and can issue any commands.

Since the EMIF performs initialization upon Deep Power-Down exit, the reg_refresh_rate field in the SDRAM Refresh Control register must be set appropriately to meet the 200us wait requirement for LPDDR1 during initialization.

9.2.3.4.5.4 Save and Restore

The EMIF supports save and restore mechanism to completely switch off power to the EMIF. The following sequence of operations is followed to put EMIF in off mode.

- An external master reads the following memory mapped registers using the system OCP interface and saves their value external to the EMIF.
  - SDRAM Configuration register
  - SDRAM Refresh Control register
  - SDRAM Refresh Control Shadow register
  - SDRAM Timing 1 register
  - SDRAM Timing 1 Shadow register
  - SDRAM Timing 2 register
  - SDRAM Timing 2 Shadow register
  - SDRAM Timing 3 register
  - SDRAM Timing 3 Shadow register
  - Power Management Control register
  - Power Management Control Shadow register
  - OCP Configuration register
  - System OCP Interrupt Enable Set Register
  - DDR PHY Control 1 register
  - DDR PHY Control 1 Shadow register
  - DDR PHY Control 2 register
- The pwr_SIdleReq port is asserted keeping the pwr_Fclken=0.
- EMIF requests OCP disconnect on both OCP interfaces through the sys_SConnect port.
- EMIF waits for the OCP disconnection confirmation on both OCP interfaces through the sys_MConnect

port.

- EMIF completes all pending transactions and drains all its FIFOs.
- EMIF puts the SDRAM in Self Refresh.
- EMIF copies all shadow memory mapped registers to its main registers. It is assumed that the shadow register always has the same value as its corresponding main register.
- EMIF waits for all interrupts to be serviced.
- EMIF gates internal ocp_clk and m_clk.
- EMIF acknowledges assertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The rst_por_arst_n reset is asserted.
- The clocks and power to the EMIF can now be switched off.

To restore power to the EMIF the following sequence of operations is followed:

- The power and clocks to the EMIF are switched on.
- The rst_por_arst_n reset is deasserted keeping prcm_deviceoff_wkup_core_rstactst=1 indicating to the EMIF that it is waking up from off mode. The prcm_deviceoff_wkup_core_rstactst port is synchronized inside the EMIF and therefore must be stable before rst_por_arst_n is deasserted.
- The EMIF does not perform SDRAM initialization and forces its state machine to be in Self Refresh.
- The pwr_SIdleReq port is deasserted.
- EMIF acknowledges deassertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The external master restores all of the above memory mapped registers.
- The system can now perform access to the external memory.
- The EMIF will exit Self Refresh state once it receives accesses to the external memory.

9.2.3.4.5.5 Clock Frequency Change

If dual_clk_mode port is set to 1, the ocp_clk and the m_clk are asynchronous to each other. In this case, the ocp_clk frequency can be changed at any time. However, the m_clk frequency can only be changed after putting the external SDRAM in Self Refresh.

If dual_clk_mode port is set to 0, the ocp_clk and the m_clk are synchronous to each other. In this case, the ocp_clk and the m_clk frequency can only be changed after putting the external SDRAM in Self Refresh. Thus the clock frequencies can be changed through software by putting the EMIF in Self Refresh.

In case software intervention is not required, EMIF supports two modes for clock frequency change. The first mode uses a similar sequence of operation as the Save and Restore mechanism. Following is the sequence of operations.

- All shadow memory mapped registers are programmed for target frequency values.
- The pwr_SIdleReq port is asserted keeping the pwr_Fclken=0.
- EMIF requests OCP disconnect on both OCP interfaces through the sys_SConnect port.
- EMIF waits for the OCP disconnection confirmation on both OCP interfaces through the sys_MConnect port.
- EMIF completes all pending transactions and drains all its FIFOs.
- EMIF puts the SDRAM in Self Refresh.
- EMIF copies all shadow memory mapped registers to its main registers.
- EMIF waits for all interrupts to be serviced.
- EMIF gates internal ocp_clk and m_clk.
- EMIF acknowledges assertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The frequency of clocks can now be changed.

The second mode allows lower latency by bypassing the OCP disconnect protocol. The config_freq_change port must be set to a 1 for this mode. This mode allows a use case where the system does not want to wait for the EMIF FIFOs to be drained to change the clock frequency. Following is the sequence of operations.

- All shadow memory mapped registers are programmed for target frequency values.
- The pwr_SIdleReq port is asserted keeping the pwr_Fclken=1.
- EMIF completes the on-going access and then puts the SDRAM in Self Refresh.
- EMIF copies all shadow memory mapped registers to its main registers.
- EMIF acknowledges assertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The frequency of clocks can now be changed.

To start performing access to SDRAM, the following sequence of operations is followed:

- The pwr_SIdleReq port is deasserted.
- EMIF acknowledges deassertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The system can now perform access to the external memory.
- The EMIF will exit Self Refresh state once it receives accesses to the external memory.

9.2.3.4.5.6 Retention Mode

EMIF supports full-retention mode, i.e., if retention is required, all flip-flops must be retention flip-flops. It does not support partial retention. The following sequence of operations is followed to put EMIF in retention.

- The pwr_SIdleReq port is asserted keeping the pwr_Fclken=0.
- EMIF requests OCP disconnect on both OCP interfaces through the sys_SConnect port.
- EMIF waits for the OCP disconnection confirmation on both OCP interfaces through the sys_MConnect port.
- EMIF completes all pending transactions and drains all its FIFOs.
- EMIF puts the SDRAM in Self Refresh.
- EMIF copies all shadow memory mapped registers to its main registers. It is assumed that the shadow register always has the same value as its corresponding main register.
- EMIF waits for all interrupts to be serviced.
- EMIF gates internal ocp_clk and m_clk.
- EMIF acknowledges assertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The clocks to the EMIF can now be switched off.
- The can now be put into retention.

To exit retention state the following sequence of operations is followed:

- The EMIF flip-flops value is restored from retention flops and clocks to the EMIF are switched on.
- The pwr_SIdleReq port is deasserted.
- EMIF acknowledges deassertion of pwr_SIdleReq through the pwr_SIdleAck port.
- The system can now perform access to the external memory.
- The EMIF will exit Self Refresh state once it receives accesses to the external memory.

9.2.3.4.5.7 Global Warm Reset

EMIF supports global warm reset mode where the EMIF will follow the following sequence of operations.

- The prcm_globalwarm_rstactst port is asserted indicating to the EMIF that it needs to enter global warm reset mode.
- EMIF completes the on-going access and then puts the SDRAM in Self Refresh.
- EMIF clears all its FIFO contents and deasserts sys_SCmdAccept and sys_SDataAccept ports.

To exit global warm reset state the following sequence of operations is followed:

- The prcm_globalwarm_rstactst port is deasserted indicating to the EMIF that global warm reset has finished.
- The system can now perform access to the external memory.
- The EMIF will exit Self Refresh state once it receives accesses to the external memory.

9.2.3.4.6 Programming/Usage Guide

9.2.3.4.6.1 Connecting to SDRAM

Figure 9-89 and Figure 9-90 show the ODT signal connections between the EMIF and DDR2.

Figure 9-89. Connecting Two DDR2 on One Chip Select

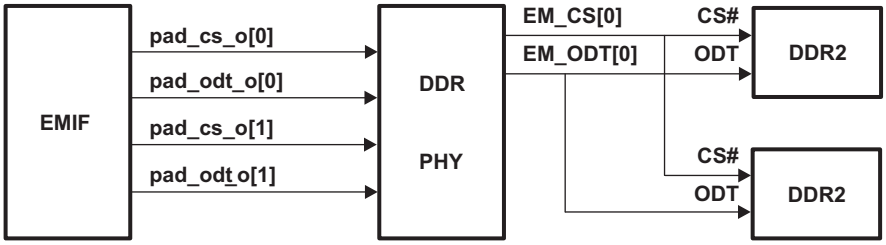
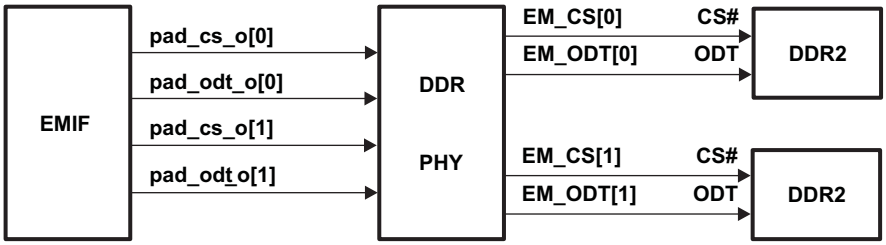


Figure 9-90. Connecting Two DDR2 on Two Chip Selects



9.2.4 SMS Basic Programming Model

9.2.4.1 SMS Firewall Usage

Use the SMS.SMS_SECURITY_CONTROL register to allow access to the registers.

Because region 0 always has level 0 priority, it can be masked by any protected region.

To configure a region:

1. Set the SMS.SMS_RG_ATTi register (where i is the region index from 0 to 7).
2. Set the SMS.SMS_RG_RDPERMi register.
3. Set the SMS.SMS_RG_WRPERMi register.
4. Set the SMS.SMS_RG_STARTj register (except for region 0. j is the region index from 1 to 7).
5. Set the SMS.SMS_RG_ENDj register (except for region 0. j is the region index from 1 to 7).

Region 0 is always active. There are no start and an end address for region 0. As soon as the SMS.SMS_RG_STARTj[30:16] STARTADDRESS and SMS.SMS_RG_ENDj[30:16] ENDADDRESS registers (where j = 1 to 7) are programmed, the firewall is activated. To prevent unexpected violations, ensure that the protection regions do not overlap.

Region 1 ensures the proper dynamic programming of protection for regions 2 through 7. For example, the protected region A is set and currently accessed, but it must be enlarged. To avoid deactivating region A and exposing its content to unwanted leakage, region 1 can be used to mask this whole area during reprogramming. When region A is correctly reprogrammed, region 1 can be deactivated.

If the secure system wants to restrict use of the firewall to secure mode, the security control register allows setting of a bit that locks the access to all registers to secure supervisor only.

On general-purpose (GP) devices, when all the required regions are programmed, the locking mechanism allows freezing the configuration, thus ensuring no further reprogramming.

To program the SMS firewall region in secure and supervisor privilege only, the SMS_SECURITY_CONTROL register allows any transaction or only secure and supervisor privilege accesses to:

- Rotation context registers (only secure privilege required)
- Arbitration control registers (only secure privilege required)
- Region 1 security firewall registers
- Error log registers (only secure privilege required)
- Security control register configuration lock bit
- Firewall lock bit
- Soft-reset lock bit (only secure privilege required)

If these fields are programmed to 0x1 (meaning that secure and supervisor privilege transaction is required), then nonsecure and supervisor privilege accesses to corresponding registers do not modify the register values and generate an ERRORSECREG error in the SMS_ERR_TYPE register.

9.2.4.2 VRFB Context Configuration

Using the RE requires several initialization programming steps. After an RE context is set up, an application can use it transparently, as if addressing a frame buffer object with a standard raster-based memory arrangement.

1. Define the page configuration. This operation usually depends only on the external memory device. The same page settings are then applied to any newly created rotated frame buffer.

   Consider an SDRAM device with 1024-byte pages. The page can be defined as a 16 × 64 array. The page is not necessarily a square (32 × 32 is also a suitable value). It is recommended that the longest side corresponds to the access direction requiring the maximum bandwidth.

   In terms of register settings, in any context that uses that page size:

   Page (page) width = 2^{pw} bytes (that is, pw = 6)

   Page (page) height = 2^{ph} rows (that is, ph = 4)

2. The application must allocate the appropriate amount of memory in the SDRAM address space, as required by the size of the frame buffer object.
   Example: Create a 400 × 300 frame buffer, 16 bits per pixel
   - Pixel size = 2^{ps} bytes (that is ps = 1)
   - Number of pages per line: 400 × 2/64 = 12.5, rounded up to 13
   - Number of pages per column: 300/16 = 18.75, rounded up to 19

   In terms of register settings, the image size parameters correspond to the enlarged image, and are programmed to:

   - Image width = w pixels (that is, w = 13 × 64 /2 = 416)
   - Image height = h pixels (that is, h = 19 × 16 = 304)

NOTE: In this example, the values obtained are not integer values, but are rounded up to the closest integer value. This results in a loss of physical memory, generally negligible compared to the total size of the image.

In terms of memory allocation (in the physical memory), this corresponds to a 416 ×304 × 2 = 252,928-byte buffer.

The physical base address of this buffer must be aligned on a page boundary (in that example, a 1024-byte boundary; that is, the 10 LSBs of the base address must be all zeros). This buffer must be allocated as a contiguous memory segment.

All these parameters, once determined, must be loaded in the registers of the chosen context (there are 12 VRFB contexts with 12 independent sets of registers).

Example, context 1:

- Configure the physical base address of the frame buffer. Example: 512 Mbytes, start of CS0, SMS.SMS_ROT_PHYSICAL_BAn[30:0] PHYSICALBA = 0x20000000 (where n = 1)

- Configure the image height and width:
  Image width (416): SMS.SMS_ROT_SIZEn[10:0] IMAGEWIDTH = 0x1A0 (where n = 1)
  Image height (304): SMS.SMS_ROT_SIZEn[26:16] IMAGEHEIGHT = 0x130
- Configure the control parameters:
  Pixel size (example: 2 bytes, 2^1 bytes): SMS.SMS_ROT_CONTROLn[1:0] PS = 0x1 (where n = 1)
  Page (page) size (example: 1024 bytes = 64 bytes × 16 bytes)
  Page height (example: 16 rows, 2^4 rows): SMS.SMS_ROT_CONTROLn[10:8] PH = 0x4
  Page width (example: 64 bytes, 2^6 bytes): SMS.SMS_ROT_CONTROLn[6:4] PW = 0x6

3. The frame buffer just created is now ready for use by the different system initiators (MPU, sDMA, LCD controller, etc.).

   From the perspective of these modules, the frame buffer object can be accessed at the following VRFB address-space memory locations:

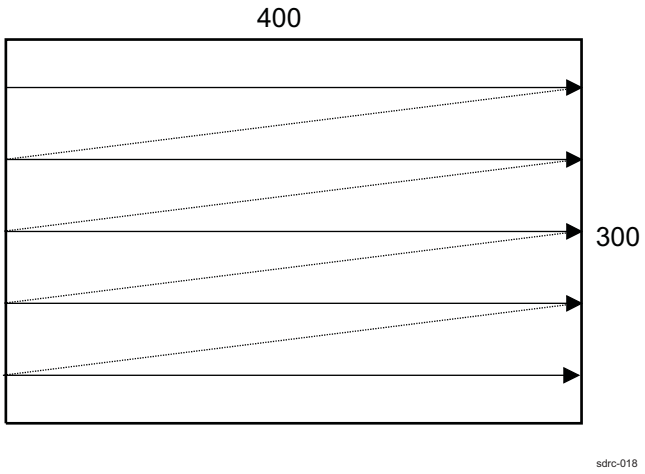
   - Context 1 0-degree view: 0x7400 0000
   - Context 1 90-degree view: 0x7500 0000
   - Context 1 180-degree view: 0x7600 0000
   - Context 1 270-degree view: 0x7700 0000

   The start address of the view corresponds to the logical origin of the image (x = 0, y = 0). The image pitch parameter, commonly defined as the distance between two vertically adjacent pixels, is fixed to 2048 pixels.

   Example of usage by the application:

   In a system using an 400 × 300 LCD panel that has a native landscape orientation, the natural scan order is as shown in Figure 9-91.

Figure 9-91. Natural Scan Order



   The display buffer is the one created in the example sequence.

   When the application is running and uses the portrait orientation for the display (typically, a PDA-type application):

   - The LCD controller accesses the frame buffer using the 0-degree view.
   - The processor and other initiators, such as 2D DMA or 3D accelerators, use the 90-degree view.

   When the application is running and uses the landscape orientation for the display (typically, a video recorder/player or gaming application):

   - The LCD controller still accesses the frame buffer using the 0-degree view.
   - The processor and other initiators, such as 2D DMA or 3D accelerators, also use the 0-degree view. See Section 9.2.5.1.1.

9.2.4.3 Memory-Access Scheduler Configuration

The memory-access scheduler is configured as follows:

- Register security settings
  - SMS.SMS_SECURITY_CONTROL register
    - Arbitration control registers security level (SMS.SMS_SECURITY_CONTROL[5] ARBITRATIONREGSLOCK bit)
- For each of the three classes, the arbitration parameters are:
  - SMS.SMS_CLASS_ARBITER0 through SMS.SMS_CLASS_ARBITER2
    - One high-priority FIFO queue in the class (HIGHPRIOVECTOR field)
    - Number of consecutive transactions to perform (EXTENDEDGRANT field)
    - Burst transaction submitted for arbitration immediately or after the burst has been buffered (BURST-COMPLETE field)

9.2.4.4 Error Logging

All data transfers in the SMS are full handshake. The SMS uses this capability to signal the system when a transaction error is detected.

The SMS captures the address of the faulty access in the SMS.SMS_ERR_ADDR register. The error type is logged in the SMS.SMS_ERR_TYPE register. Once a faulty access is logged and the SMS.SMS_ERR_TYPE[0] ERRORVALID bit is set, the next faulty accesses cannot be logged before clearing the ERRORVALID bit.

In the case of an interconnect transaction, an error response is generated if any of the following occur:

- An incoming request arrives after an idle request from the PRCM.
- An illegal command is received.
- A protection region overlap is detected.
- A nonsecure or non-supervisor write access to the security control register is detected.
- Protection errors.

It is assumed that the system interconnect on which the SMS is plugged is responsible for signaling the error event to the host MPU based on the interconnect response. The MPU error handler can then consult the error logging registers.

The security control register allows locking the SMS.SMS_ERR_ADDR and SMS.SMS_ERR_TYPE registers to secure- and supervisor-only accesses. This configuration is highly recommended for high-security devices. For GP devices, ErrorLog can remain public.

9.2.5 SDRC Use Cases and Tips

9.2.5.1 How to Program the VRFB

9.2.5.1.1 VRFB Rotation Mechanism

An inherent limitation of SDRAM technology is high-memory latency caused by page-miss penalties incurred when downloading to a memory cache. For example, switching from one page to another in external memory can cause a page-miss, indicating that the page accessed for the current pixel is different from that for the previous pixel.

A DMA engine is used to rotate pictures in external DRAM, but this rotation method increases the number of page misses.

The efficient way to rotate image data in external SDRAM is to use the VRFB module, which is an RE embedded in the SMS of the processor, as shown in Figure 9-92 . It is configured in the SMS registers.

Accessing images stored in external SDRAM in a non-natural order requires an address transaction: the virtual address of an image is translated into a physical address in the corresponding buffer. Address translation causes an image to be rotated in 0-, 90-, 180-, or 270-degree views. With multiple views of the image, the RE can change addresses and issue multiple requests to the EMIF so that a maximum of consecutive accesses is performed, thus decreasing the number of page-miss penalties. The RE cannot reorder requests to the SDRAM.

A VRFB context defines the configuration used to access a picture in external SDRAM. For each VRFB context, a set of registers in the SMS describes:

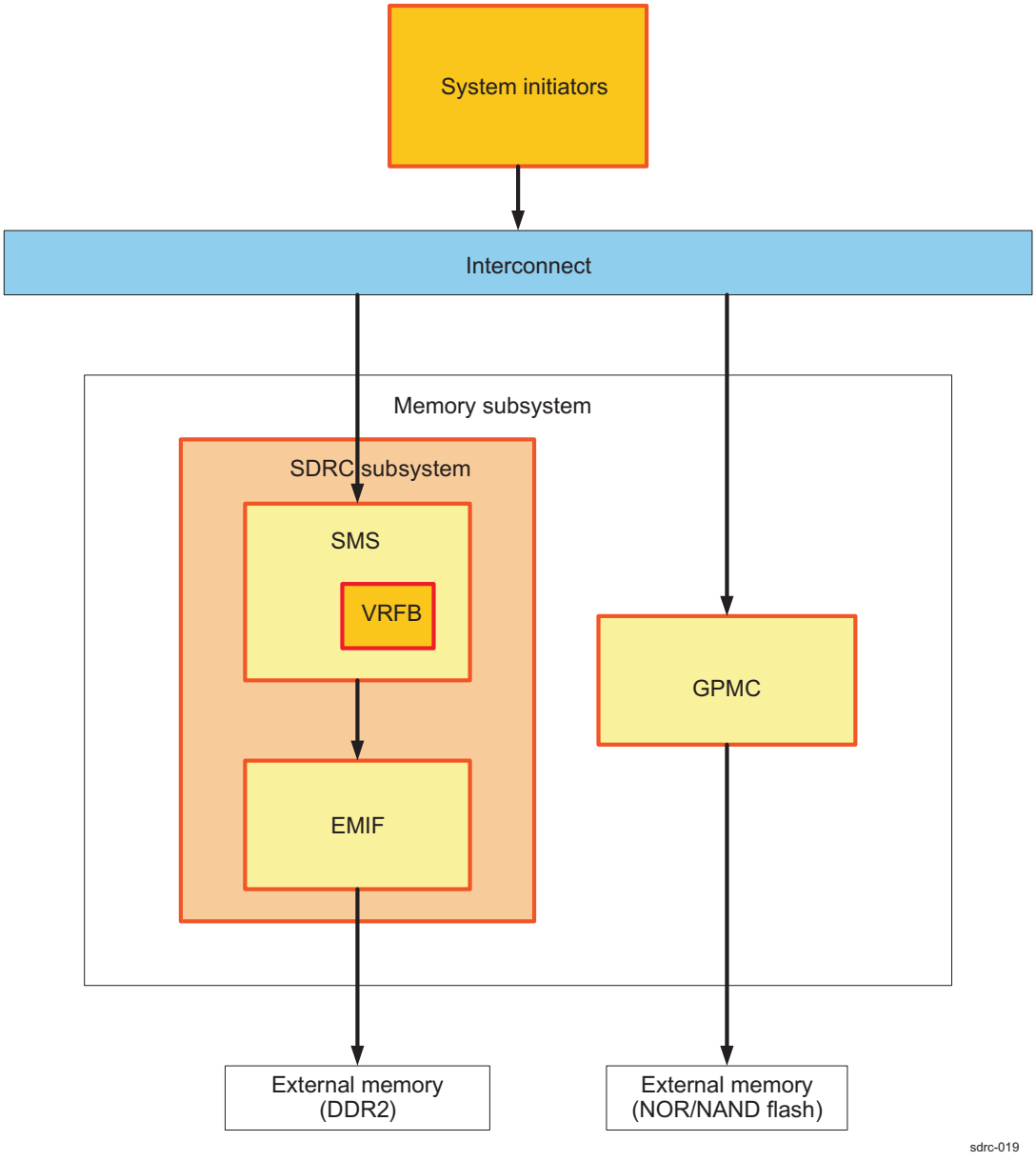
- The size of the page (height and width)
- The picture parameters before rotation (width, height, and pixel format)
- The rotation angle (0-, 90-, 180-, or 270-degree)
- The physical base address in external memory

Any initiator in the processor can access up to 12 images simultaneously, each image having an independent context. These 12 contexts is considered virtually addressed image buffers. Each context is assigned to a physical image buffer.

The virtual address space of each context is subdivided into four separate address regions pointing to the same physical image buffer but corresponding to the four possible rotations: 0/90/180/270.

After the VRFB context is configured, all data accesses to an address space are automatically translated when accessing SDRAM through the RE.

Figure 9-92. SDRC Subsystem Overview



9.2.5.1.2 Setting a VRFB Context

NOTE: YUV Format

The YUV standard shows a color space with three elements:

- Y for luminance
- U for chrominance
- V for chrominance

As shown in Figure 9-93, pixel format (not pixel size) used in the YUV standard is spread onto a 32-bit data structure:

- This 32-bit data structure represents a packet of two pixels: P0 and P1.
- Each element (Y0, Y1, U, and V) is coded in 8 bits.
- P0 = Y0; U, V, and P1 = Y1, U, V.

There are 24 bits of information per pixel (Y, U, and V); however, because the chrominance elements U and V are common to both P0 and P1 pixels, only 16 bits are used to store a pixel in YUV format. The YUV standard uses a 32-bit data structure to represent 2 pixels; the pixel format uses 32 bits (4 bytes).

Thus, when defining YUV image parameters, the image width must be set to one-half the number of pixels per row and the pixel format must be set to 4 bytes, because the YUV pixel data is spread onto a 32-bit word representing 2 pixels.

Figure 9-93 shows the pixel representation of the YUV format.

Figure 9-93. YUV Format: Pixel Representation

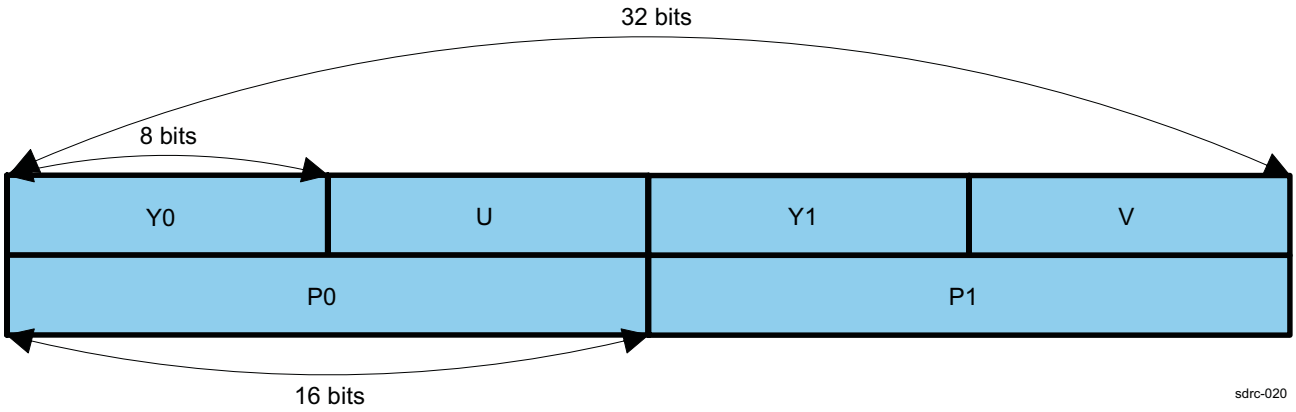


Figure 9-94 shows a generic way to configure a VRFB context to perform a rotation view on pixel data in external DRAM.

Figure 9-94. VRFB Context Configuration

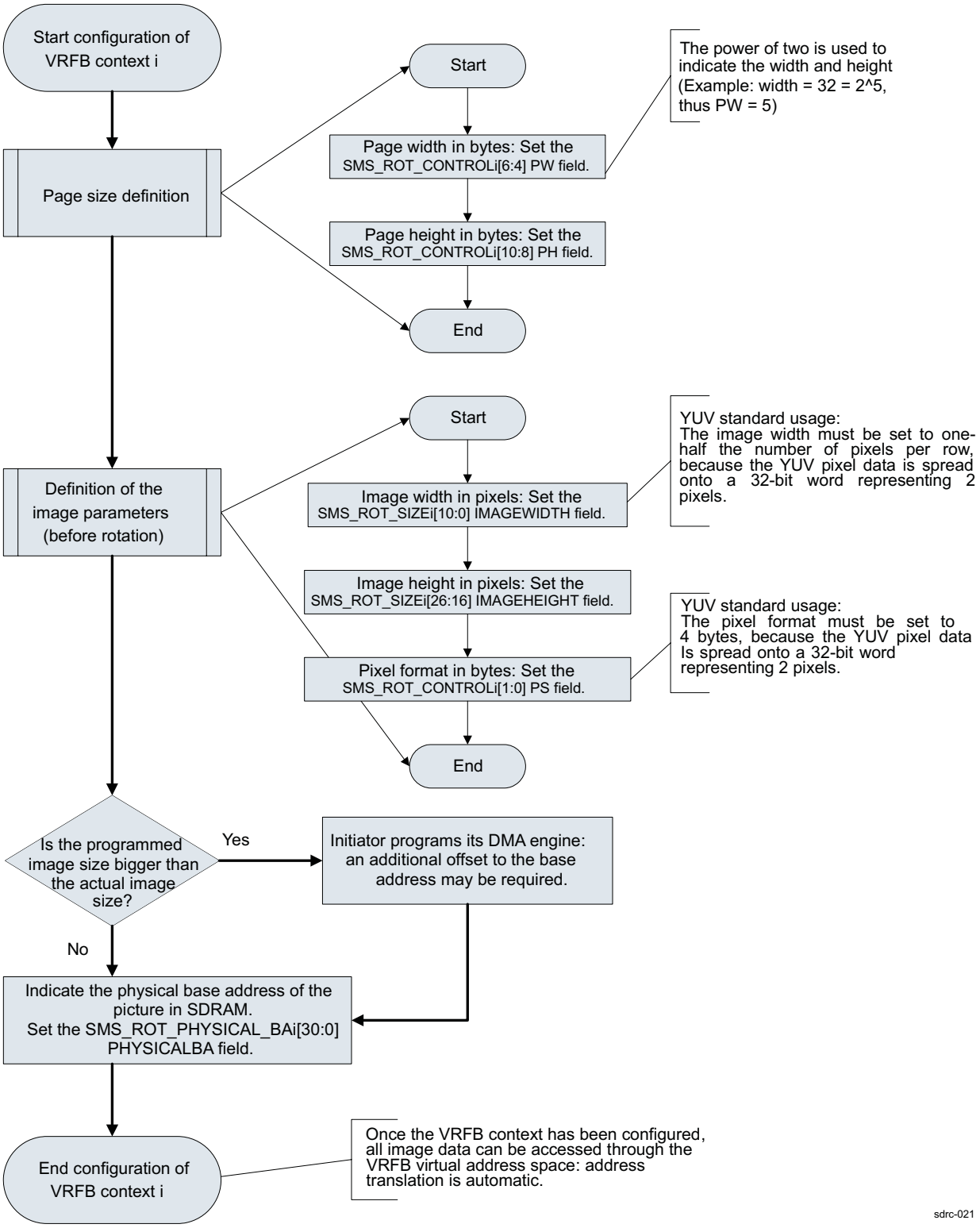


Figure 9-95 is an example of VRFB context configuration using both RGB and YUV image formats. This example represents the configuration of a 1024-byte page size (32 × 32 byte page) and a 240 × 240 image size using VRFB context 1.

Figure 9-95. Example of VRFB Context 1 Configuration

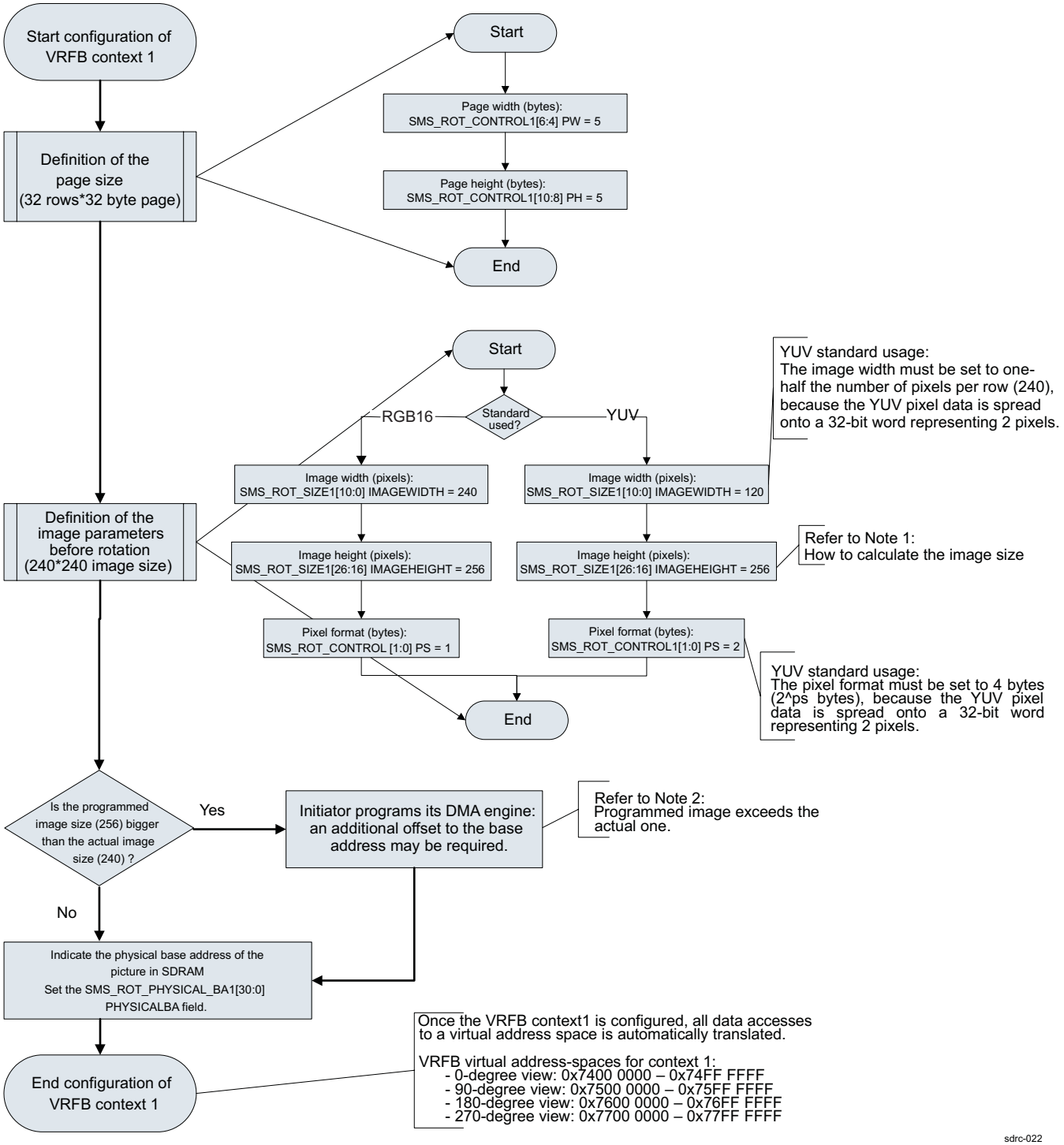


Table 9-111 lists guidelines for calculating image size.

Table 9-111. Calculating Image Size[1] [2]

Steps		RGB16 Format (16-bit Pixel Format)		YUV Format (32-bit Pixel Format)	
		IMAGEWIDTH	IMAGEHEIGHT	IMAGEWIDTH	IMAGEHEIGHT
1	Calculate the required number of pages per line and per column.	240 pixels/32 bytes × 2 bytes per pixel = 15 pages per line	240/32 rows = 7.5, rounded to 8 pages per column	120 pixels/32 bytes × 4 bytes per pixel format = 15	240/32 rows = 7.5, rounded to 8
2	Using the number of pages calculated in Step 1, calculate the image size in pixels.	15 × 32 bytes/2 bytes per pixel = 240	8 × 32 rows = 256	I5 × 32 bytes/4 bytes per pixel format = 120	256

(1) Working on a page basis, the image size must be a multiple of the page size.

(2) Depending on the page dimensions and the image size, the programmed image size (256) is larger than the actual image size (240).

To use VRFB rotation, each initiator must configure its DMA engine correctly; an additional offset to the base address may be required. For details, see the *Display Subsystem* chapter.

9.2.5.1.3 Applicative Use Case and Tips

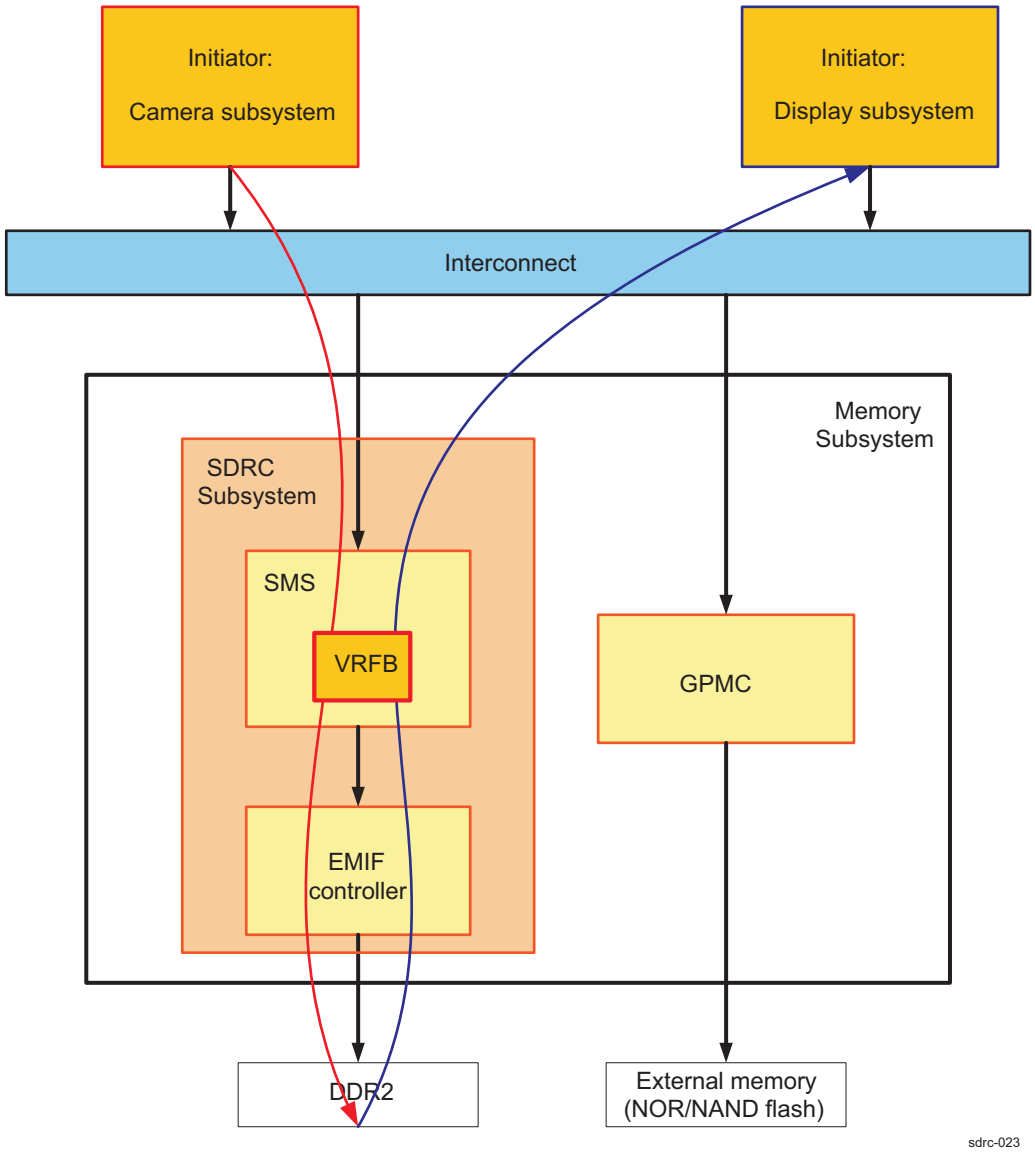
Use Case Scenario: Display a Rotated QVGA Image

Figure 9-96 this use case. The camera captures an image with a particular rotation and stores the pixel data in external memory. Displaying the image on the LCD requires different orientation.

The following components interact while displaying the rotated image:

- SDRC subsystem
- Camera subsystem
- Display subsystem
- External DDR
- VRFB

Figure 9-96. Display a Rotated QVGA Image



The following conditions exist for this application:

- QVGA image size is 320 × 240 (width × height).
- Pixel format is YUV4:2:2 (YUV2) in little endian.
- The rotation view is 90 degrees.
- The VRFB context is VRFB context 1.
- The camera module writes QVGA images to external DRAM at 15 fps.
- The display subsystem reads QVGA images from external DDR2 at 60 fps and displays them on the LCD screen.
- Read and write initiators use burst mode:
  - The camera uses 8 × 64-bit burst size.
  - The display uses 8 × 32-bit burst size.
  - Double-indexing mode is selected.
- The physical address of the buffer in external memory is 0x8030 0000.
- The memory allocation for the buffer is 320 × 240- × 16-bit (or 160 × 240 × 32-bit).

- The camera subsystem uses its dedicated DMA channel 0.
- The display subsystem uses its video channel 1.

Configuring the VRFB through the SMS Registers

The size of the page supported by the DDR2 memory is 1K byte, which is arranged as 32 × 32 bytes page (width × height).

Configuring the page size:

- SMS_ROT_CONTROLn[6:4] PW = 5 (where n = 1)
- SMS_ROT_CONTROLn[10:8] PH = 5

Configuring the image parameters:

- SMS_ROT_SIZEn[10:0] IMAGEWIDTH = 160 (where n = 1)
- SMS_ROT_SIZEn[26:16] IMAGEHEIGHT = 256
- SMS_ROT_CONTROLn[1:0] PS = 2

Physical base address and rotation angle:

- SMS_ROT_PHYSICAL_BAn[30:0] PHYSICALBA = 0x8030 0000 (where n = 1).

The image data is accessed at the following virtual address range for 90-degree rotation: 0x7500 0000 – 0x75FF FFFF.

CAUTION

Image rotation using the YUV2 image format causes the stream to become untidy. The display must be specifically configured to read and reorganize the stream to conform to the YUV2 standard.

Tips for Configuring Successful Rotation

The following guidelines ensure optimal image rotation:

- Page arrangement:

  Usually, the recommendation is to have a *square* page. If this is not possible, the longest page side should correspond to the access direction that requires the maximum bandwidth; set the longest page side to optimize the page break.

  Using a 1024-byte page size, a 32 × 32-byte page arrangement is used as an example. With a 2K-byte page organized as a 32 × 64 byte page, depending on the read or write operation, set the longest page size to optimize the page break. If 0 is written and the 270-degree view is read, page height is greater than page width (PH > PW). Set PH to 64 bytes (PH = 6).
- Virtual address memory arrangement:

  When accessing image data through virtual addresses, the maximum line size supported by the VRFB is 2,048 pixels.

  In the memory buffer, the distance between two vertically adjacent pixels is fixed at 2,048 multiplied by the pixel format in bytes. This means that when reading or writing image data through virtual addresses, there must be an offset of: (2048 – IMAGEWIDTH) × PS bytes at the end of every line.
- Base address alignment:

  For optimization, the base address is aligned on the page size. For instance, a 1K-byte page size organized as a 32 × 32-byte page is aligned on 0x400 (to be adjusted on the base address).
- To improve performance on 90° rotation consider two things:
  - Because a read access can appear more critical than a write access, a posted write may be appropriate.
  - When performing burst accesses with 90° or 270° rotation views, the burst is split on the memory side, adding latency.

    For a burst access with 90° rotation, split the data of the burst on the write side (not the read side):

- Write in 270° and read in 0°.

9.2.5.2 SMS Mode of Operation

9.2.5.2.1 The SDRAM Memory Scheduler and Arbitration Policy

The SDRAM memory scheduler improves access to external memory by:

- Optimizing SDRAM bandwidth
- Prioritizing requests to external memory

This mechanism relies on a complex arbitration policy that employs specific terminology:

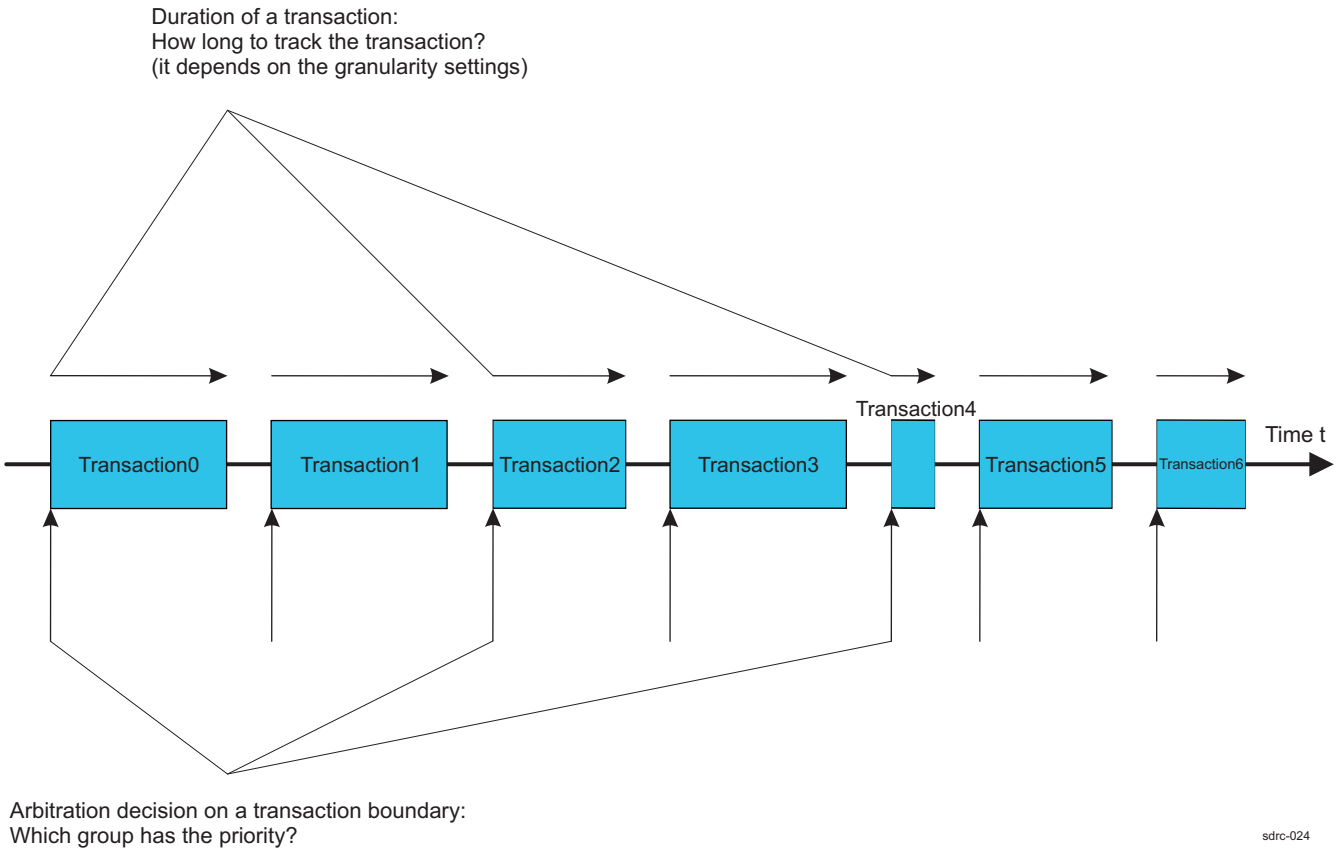
- Group: a FIFO queue of requests from initiators
- Class: A collection of groups
- Transaction: A full burst request
- Arbitration grant: Authorization of a service requested by an initiator

The arbitration policy operates on two interdependent mechanisms:

- The arbitration decision, which establishes priority for processing requests. The question: Which request is processed next? occurs on a transaction boundary.
- Arbitration granularity, which determines the length of an arbitration grant. The question: How long to keep the grant? defines the boundary of the arbitration decision point in time.

Figure 9-97, shows the link between these two concepts. On a transaction boundary, the questions: What request is serviced next, and for how long? merge the two mechanisms. The mechanisms for specifying the granularity of requests also influence the boundary of a transaction.

Figure 9-97. Arbitration Granularity Versus Arbitration Decision



9.2.5.2.2 The Arbitration Decision

The SMS module controls arbitration. Requests from initiators for access to the external SDRAM are collected into several independent FIFO queues, and each queue is assigned to a class. Priority is then assigned to groups and classes; this defines the next request to be serviced.

9.2.5.2.2.1 Burst-Complete Mechanism

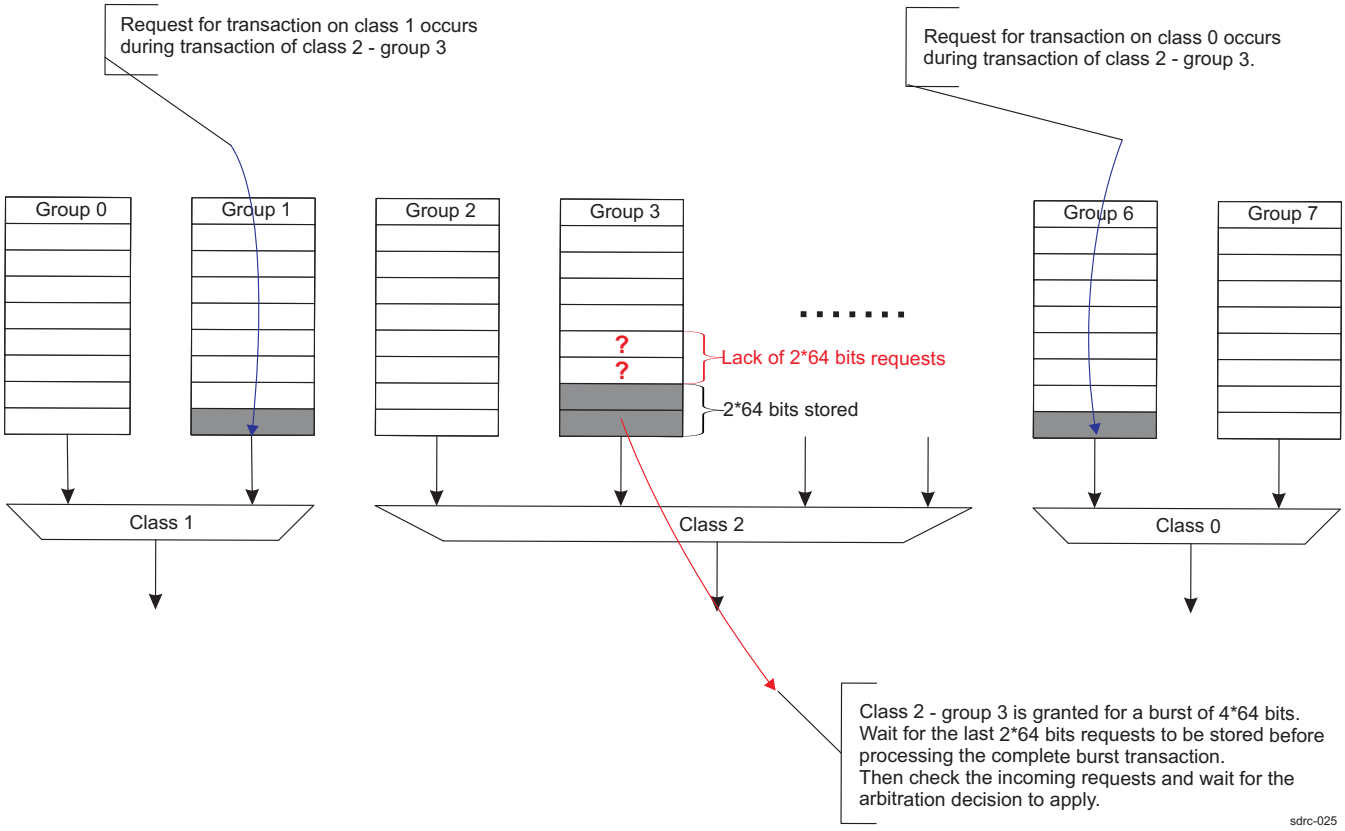
The burst-complete mechanism applies granularity to the arbitration scheme. Access cannot be granted to a group within a class until a complete burst has been stored in the FIFO.

Example:
There is no ongoing transaction on Class 0. The initiator requests a 4 × 64-bit burst on Group 3 of Class 2 (SMS_CLASS_ARBITER2[27] BURST-COMPLETE = 0x1). Only 2 × 64-bit requests are stored in the FIFO.

The mode of operation is:

- Wait for the last 2 × 64-bit request to be stored in the FIFO.
- Arbitration is requested once all requests of a burst have been received.
- After the 4 × 64-bit burst is complete, the transaction occurs.
- The arbitration choice mechanism resumes:
  - Were there any incoming requests during the last transaction?
  - What is the next request to be serviced/granted?

Figure 9-98. BURST-COMPLETE on Class 2-Group 3



9.2.5.2.2.2 Priority Between Groups

The SMS.SMS_CLASS_ARBITERi[7:6] HIGHPRIOVECTOR field defines the priority between groups in a class.

9.2.5.2.2.3 Priority Between Classes

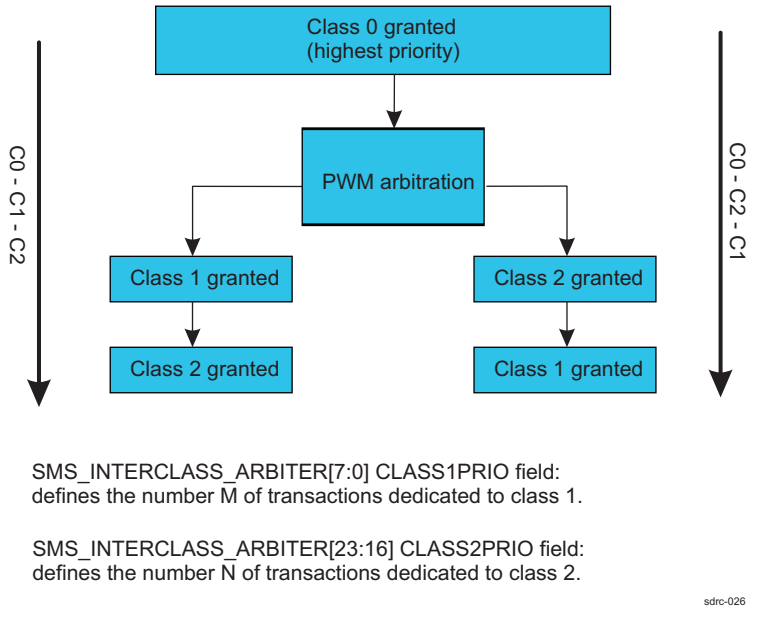
Class 0 always has the highest priority. Then you program a weight within the arbitration of Class 1 and Class 2:

- The SMS.SMS_INTERCLASS_ARBITER[23:16] CLASS1PRIO field specifies the number (M) of transactions dedicated to Class 1.
- The SMS.SMS_INTERCLASS_ARBITER[7:0] CLASS2PRIO field specifies the number (N) of transactions dedicated to Class 2.

M and N parameters are used to set a PWM arbitration. For instance, two schemes appear: at time t, Class 1 is serviced for M cycles, directly followed by service to Class 2 for N cycles. On the opposite, If Class 2 is serviced first for N cycles, then Class 1 is granted for M cycles. Arbitration is given more importance; Class 1/2 is favored over Class 2/1. As shown in Figure 9-99, two arbitration schemes can appear:

- Class 0 is serviced first, followed by Class 1, then Class 2 (C0 - C1 - C2).
- Class 0 is serviced first, followed by Class 2, then Class 1 (C0 - C2 - C1).

Figure 9-99. Priority Between Classes



9.2.5.2.3 Arbitration Granularity

Arbitration is the mechanism that give more or less importance to incoming requests depending on the initiator origin. Granularity influences the boundary of a transaction because it imposes the following questions:

- How many requests to service?
- How long to track the current transaction?
- When to make the next arbitration decision, or when does the next transaction boundary occur?

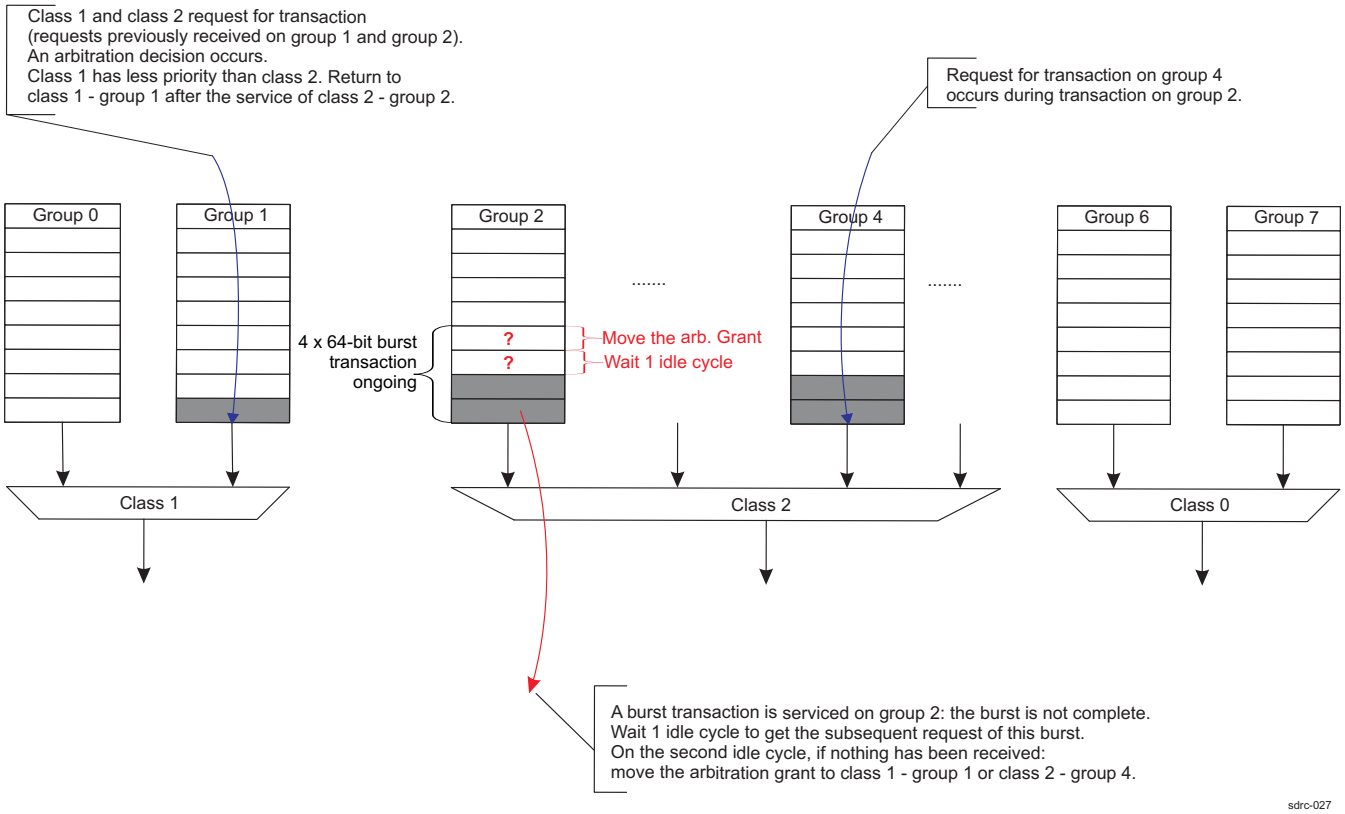
9.2.5.2.3.1 Idle Cycle

The idle gives more granularity to the arbitration; it lets the arbiter wait one idle cycle before moving the arbitration grant to another thread.

Within a burst (see Figure 9-100 for an example):

1. A thread requests a burst transaction. According to the thread ID, the last request was serviced on Class 2 – Group 2.
2. The thread cannot provide the subsequent request of the burst. The module waits for one idle cycle without moving the arbitration grant, to receive the subsequent request (from the same thread).
3. On the second idle cycle, if the subsequent request has not been received, the arbitration grant is moved so that a request from another thread is serviced.

Figure 9-100. Idle Cycle Mechanism within a Burst



Two bursts must be serviced at the burst boundary. One idle cycle after servicing these two bursts, the arbitration grant is moved.

9.2.5.2.3.2 Extended-Grant Mechanism

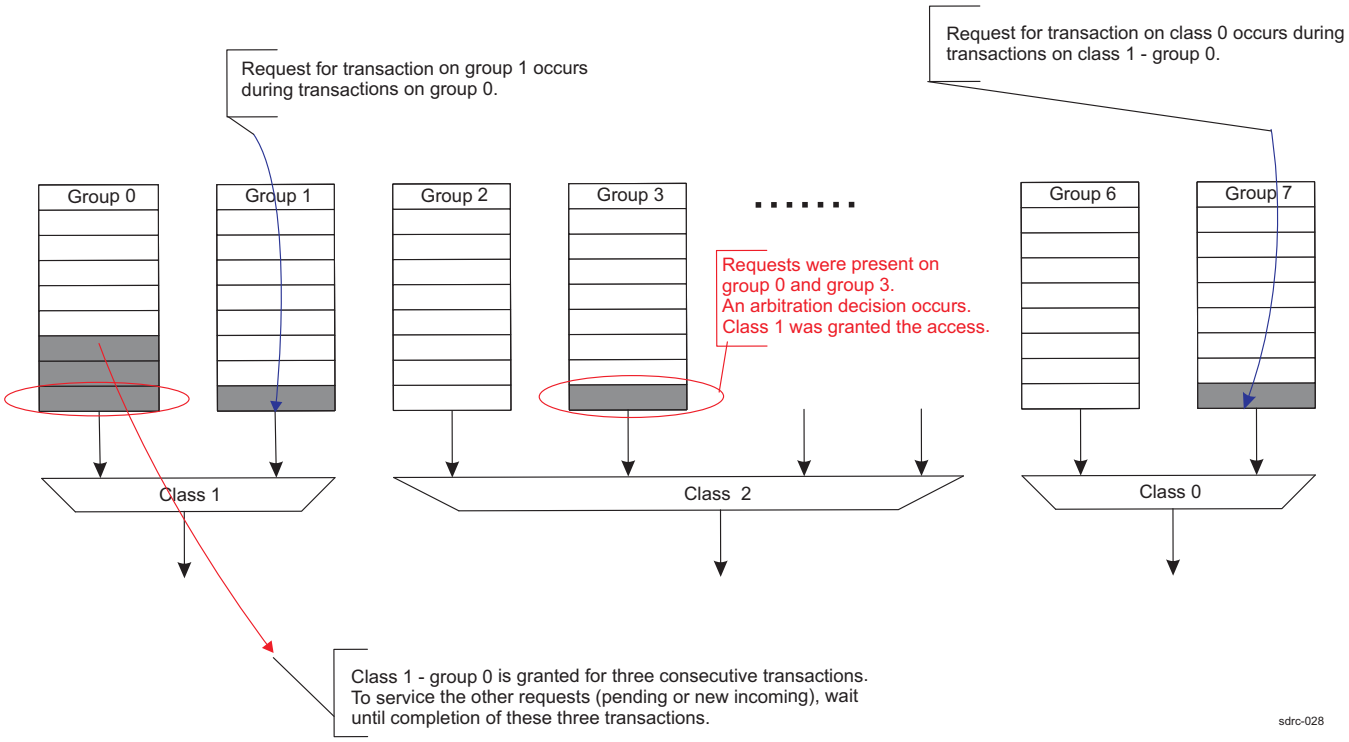
The extended-grant mechanism gives additional granularity to the arbitration. An extended grant defines the number of consecutive transactions (from 1 to 3) a group is granted.

Example:
Requests were already available on Class 1-Group 0 and Class 2-Group 3.
An arbitration decision occurs: grant access to Class 1-Group 0.
A request is now being serviced on Class 1–Group 0 with SMS_CLASS_ARBITER1[9:8] EXTENDEDGRANT = 0x3 (three consecutive transactions are granted to Group 0). The mode of operation is:

- Process the request for three cycles.
- If another request comes from another class (even Class 0) during the processing, it is ignored; the arbiter does not treat incoming requests.
- After three consecutive transactions, return to the arbitration decision:
  - Were there any incoming requests during the last transaction?
  - What is the next request to be serviced?

Figure 9-101 shows this mechanism on Class 1 - Group 0:

Figure 9-101. Example of EXTENDEDGRANT Mechanism



9.2.5.2.3.3 Number of Service Mechanism

The SMS_CLASS_ROTATIONm[4:0] NOFSERVICES field defines the number of consecutive transactions (from 1 to 31) the VRFB is granted. For more information on LRU policy and Number of Service, refer to Section 9.2.3.1.3.

9.2.5.2.4 How these Mechanisms Interact

As described in the previous sections, the arbitration within the SMS module is based on several mechanisms giving even more importance to the arbitration. Wen we need to grant the access on a transaction boundary, we recall that the question we have to answer is: what will be the next request to be serviced and for how long?
Thus, the two important concepts regarding the SMS mode of operation are:

- The arbitration decision: we choose the next request to be serviced and so we grant it the access to the external memory.
- The arbitration granularity: depending on the mechanism used, we define the boundary of the transaction: how long do we keep the grant?

Figure 9-102 describes the decision happening between classes.
Figure 9-103 explains the priority between groups within Class 1.
So far, the arbitration decision has taken place at class boundaries and within a class containing 2 groups.
Figure 9-104 recalls the generic way to service and so prioritize the requests within a class.
Finally, Figure 9-105 recalls mechanisms used to define the boundary of the transaction being serviced (how long do we keep the grant?).

Figure 9-102. Arbitration Between Classes

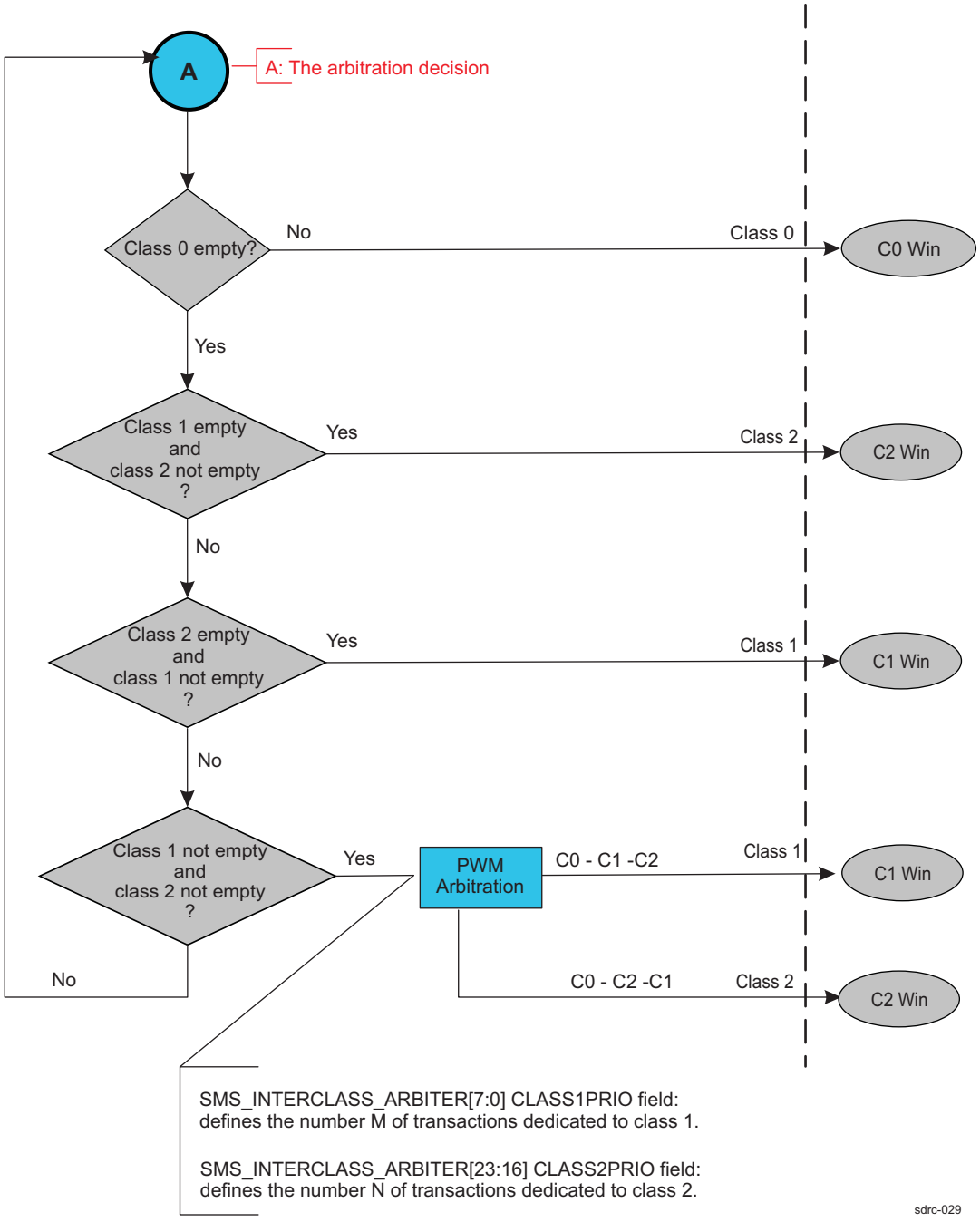


Figure 9-103. Arbitration within a Class

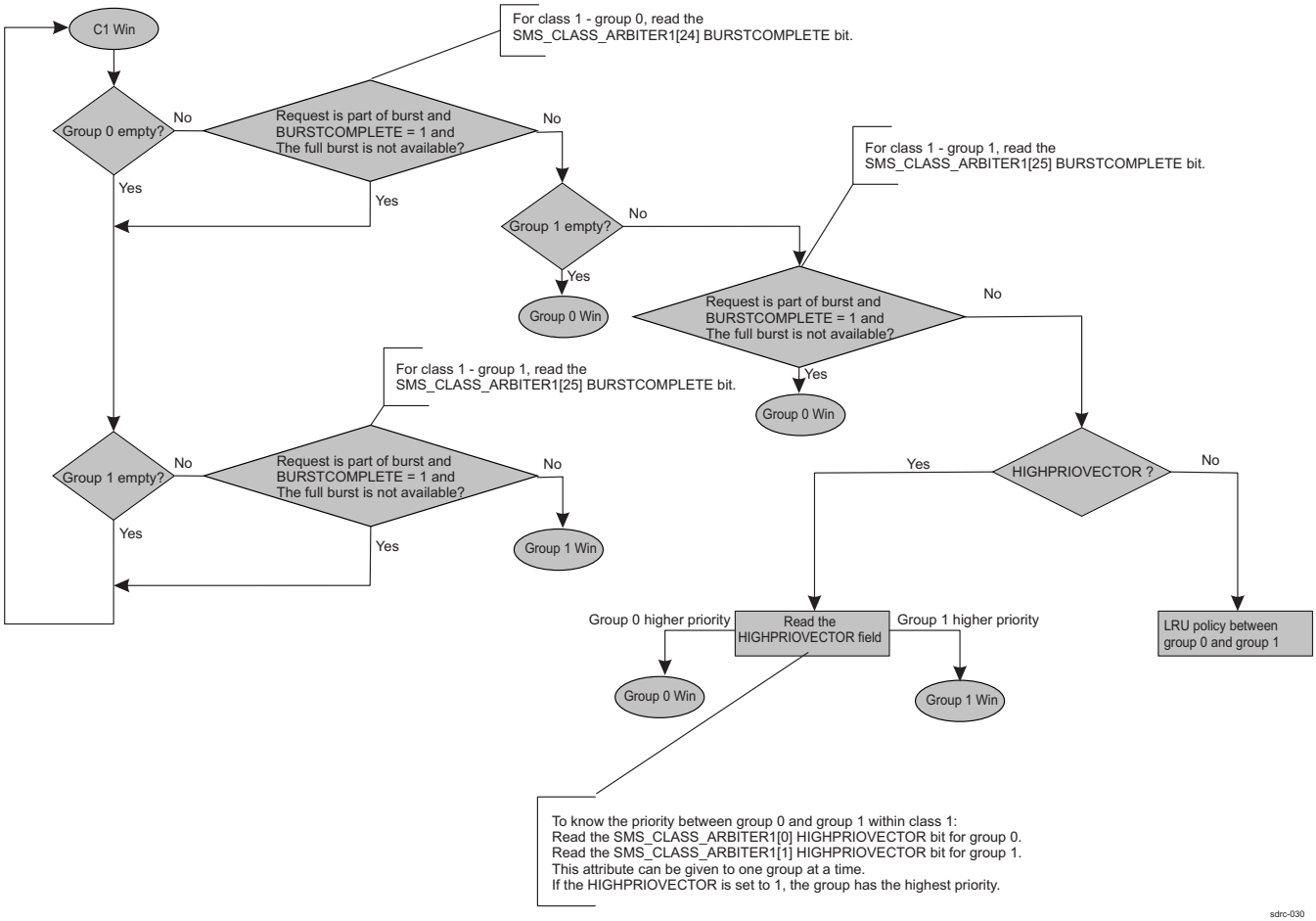


Figure 9-104. Generic Arbitration Decision

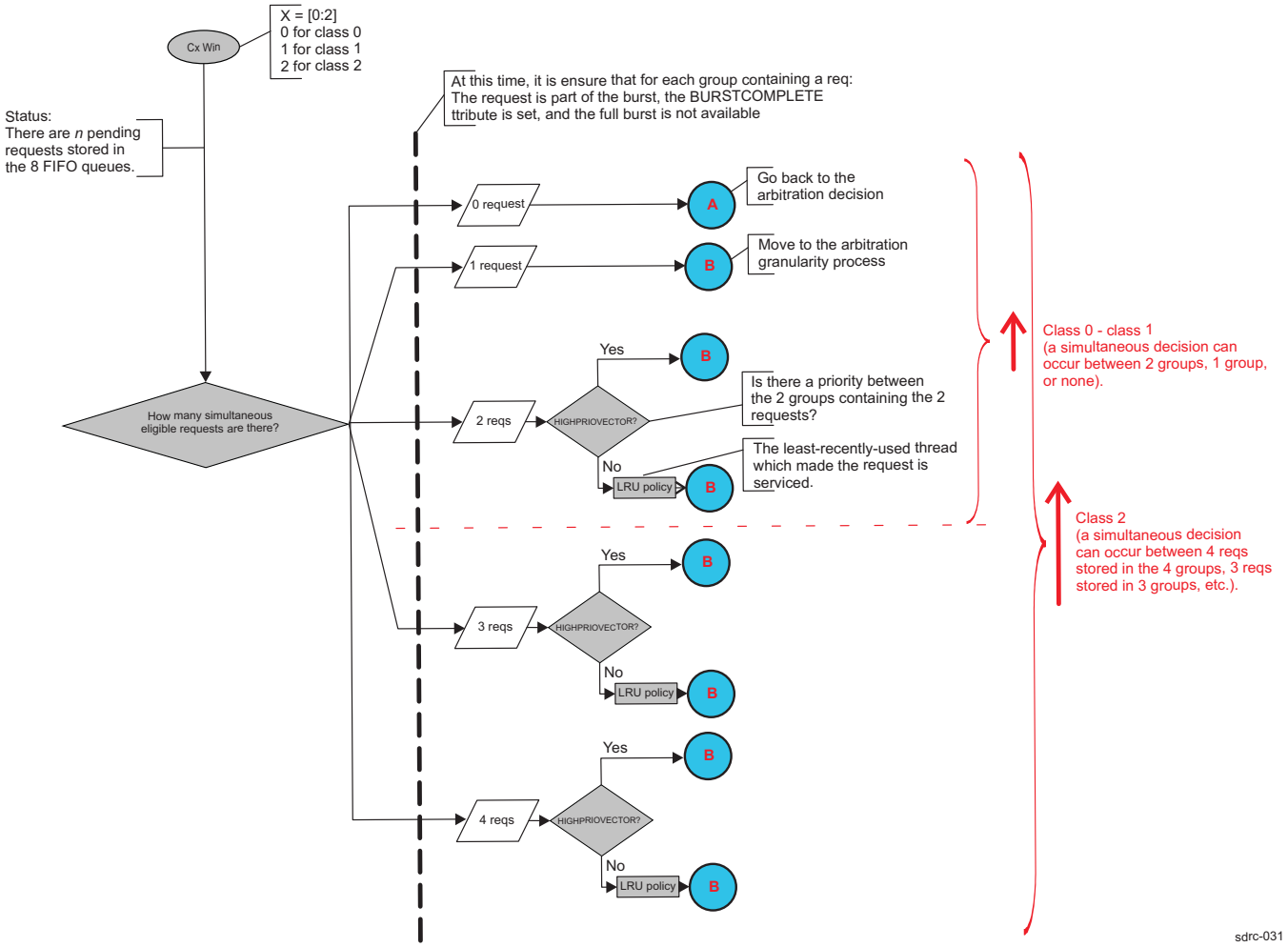
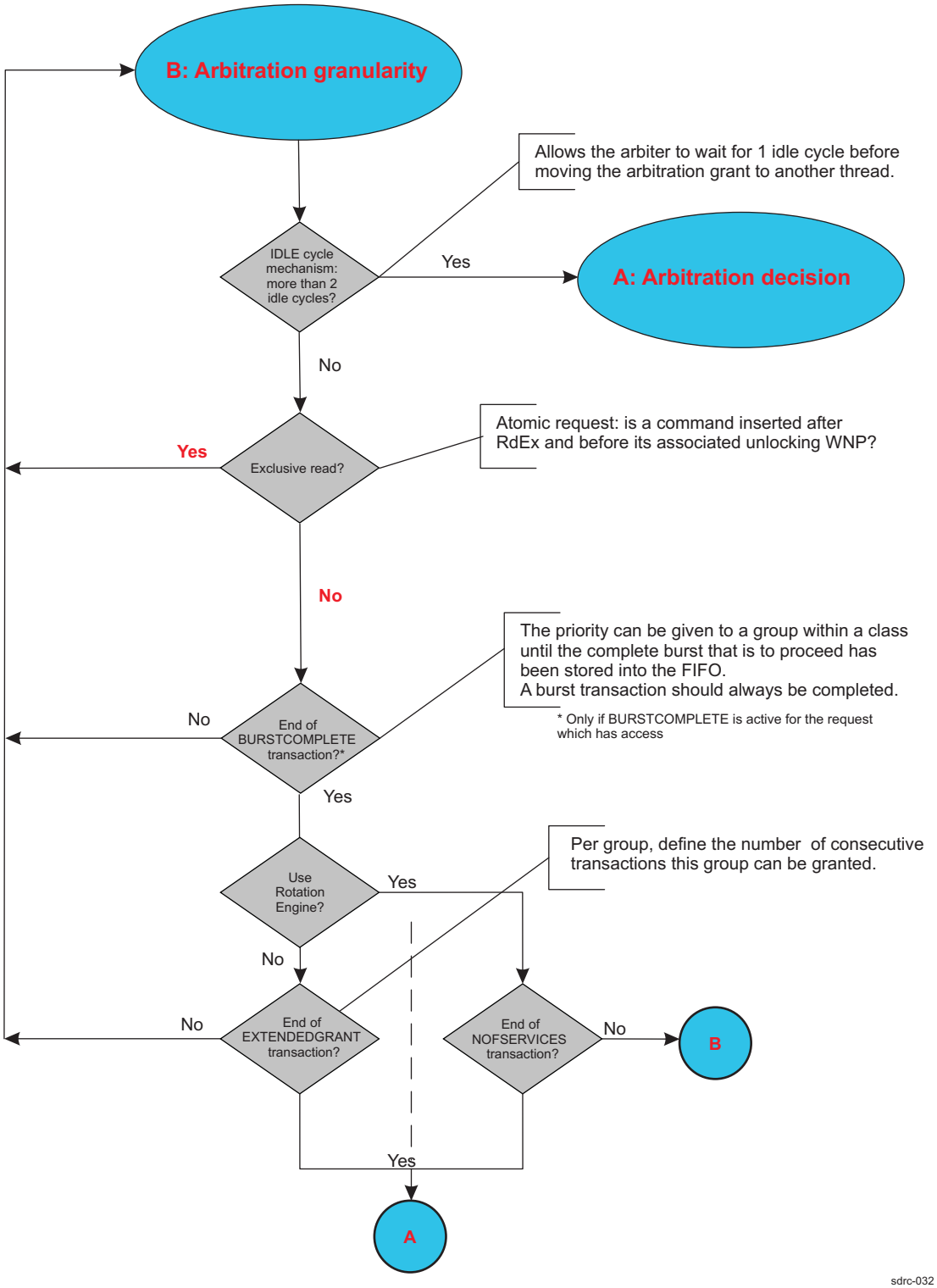


Figure 9-105. Arbitration Granularity

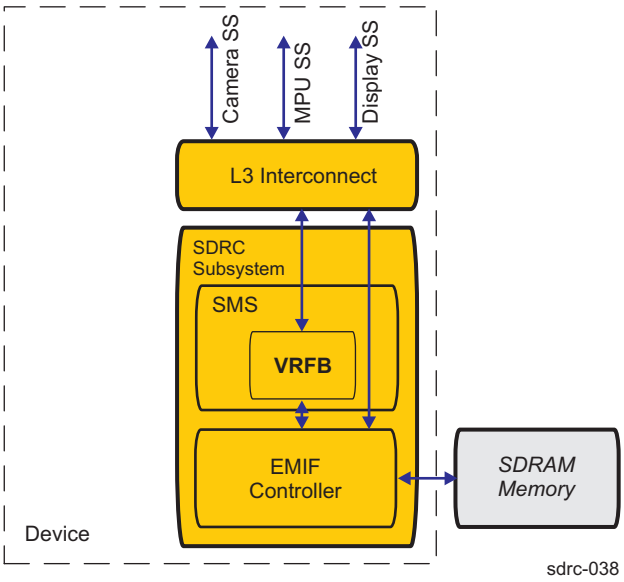


9.2.5.3 Camcorder Use Case: How to Configure the VRFB

9.2.5.3.1 Overview

This section discusses the configuration of the VRFB for rotating the video from the camera subsystem. The VRFB receives a video from the camera subsystem as a sequence of images. These images are then saved into SDRAM through the EMIF . Figure 9-106 is an overview of the VRFB in the camcorder use case.

Figure 9-106. SDRC Camcorder Use Case Overview



After the VRFB configuration, the rotation process is transparent to the MPU operations. Four buffers are required in SDRAM to ensure stabilization, a smooth encoding process, and a tearing-free preview on the LCD display. Each of the four buffers uses a concurrent context. For each context, the following parameters must be configured:

- Page and pixel size
- Picture parameter (image height and width)
- Buffer physical base address

9.2.5.3.2 Data Path

In this use case, the VRFB manages the rotation of the video stream, which is a sequence of images in YUV 4:2:2 format received from the camera subsystem.

- Write actual image received from the camera subsystem into SDRAM through the VRFB in the first view (0°).
- Display the image in SDRAM on the external display through the display subsystem in any of the four available VRFB views (0°, 90°, 180°, or 270°).

See Figure 9-96 for the VRFB data path of this use case.

The row increment in the SDRAM is programmed in the display subsystem registers and depends on the VRFB rotation angle.

9.2.5.3.3 Programming Flow

A main function automatically configures the VRFB contexts. This function is called four times to configure each of the four contexts. For each context, a set of parameters is given to the function:

- The actual image width and height (from the camera subsystem)
- The physical address (of the given context)

- The rotation of the context

The main function initializes the VFRB in three steps:

1. Virtual frame buffer configuration. This is done with a page size calculation function, as described in Section 9.2.5.3.3.1.
2. Image size configuration. This is done with a picture size calculation function, as described in Section 9.2.5.3.3.2.
3. Physical base address configuration, as described in Section 9.2.5.3.3.3.

9.2.5.3.3.1 Page Size Calculation Function

The page size is configured through three parameters: page width (PW), page height (PH), and pixel size (PS). The three page size parameters are configured into the SMS_ROT_CONTROLn registers, where n is the context number (n = 0 to 3):

- Page width: SMS_ROT_CONTROLn[6:4] PW. The PW parameter defines the page width according to the value of 2^{PW} bytes.
- Page height: SMS_ROT_CONTROLn[10:8] PH. The PH parameter defines the page height according to the value of 2^{PH} lines.
- Pixel format: SMS_ROT_CONTROLn[1:0] PS. The PS parameter defines the pixel size according to the value of 2^{PS} bytes. In this use case, the pixel size is a constant linked to the YUV 4:2:2 format. For more details about YUV 4:2:2 format, see Section 9.2.5.1.2, *Setting a VRFB Context*.

The recommendation is to have a squared page arrangement. The page can be defined as a 32-byte × 32-byte array (1024-byte page) or 16-byte × 16-byte array (256-byte page). This arrangement is pixel-based, which means the pixel size must be part of the calculation. Other settings are possible, but from a performance point of view it is recommended to have the page size consistent with the SDRAM page (that is, 2K bytes for the Mobile DDR SDRAM).

In this use case, however, the page size calculation function defines the page width and the page height. The pixel size is not part of the calculation. Hence, instead of having a 64 × 32 page (because a pixel takes 2 bytes) the function calculates a 32 × 32 page. This gives a page size of 1K-byte, and memory area usage is optimized.

First, the function checks whether the width of the image received from the camera is a multiple of 32 bytes. If the width is not a multiple of 32 bytes, the function checks whether the image is a multiple of 16 bytes. If the image from the camera is neither a multiple of 32 bytes nor 16 bytes, the function sets the page width to 32 bytes as the default. In this use case, the image received from the camera has a width of 736 pixels.

1. Check that 736 is a multiple of 32: 736 / 32 = 23. It is. As a consequence, the page width is 32 (bytes).
2. Configure a page width of 32 with the 2^{PW} bytes formula: SMS_ROT_CONTROLn[6:4] PW = 0x5.

   Then, the function checks whether the height of the image received from the camera is a multiple of 32 (rows). If the height is not a multiple of 32 (rows), the function checks whether the image is a multiple of 16 rows. If the image from the camera is neither a multiple of 32 rows nor 16 rows, the function sets the page height to 32 rows as the default. In this use case, the image received from the camera has a height of 560 lines.
3. Check that 560 is a multiple of 32: 560 / 32 = 17.5. It is not.
4. Check that 560 is a multiple of 16: 560 / 16 = 35. It is. As a consequence, the page height is 16 (lines).
5. Configure a page height of 16 with the 2^{PH} lines formula: SMS_ROT_CONTROLn[10:8] PH= 0x4.

   The pixel format of this use case is YUV 4:2:2. Two pixels are stored on 4 bytes.
6. Configure a pixel size of four bytes with the 2^{PS} bytes formula: SMS_ROT_CONTROLn[1:0] PS = 0x2.

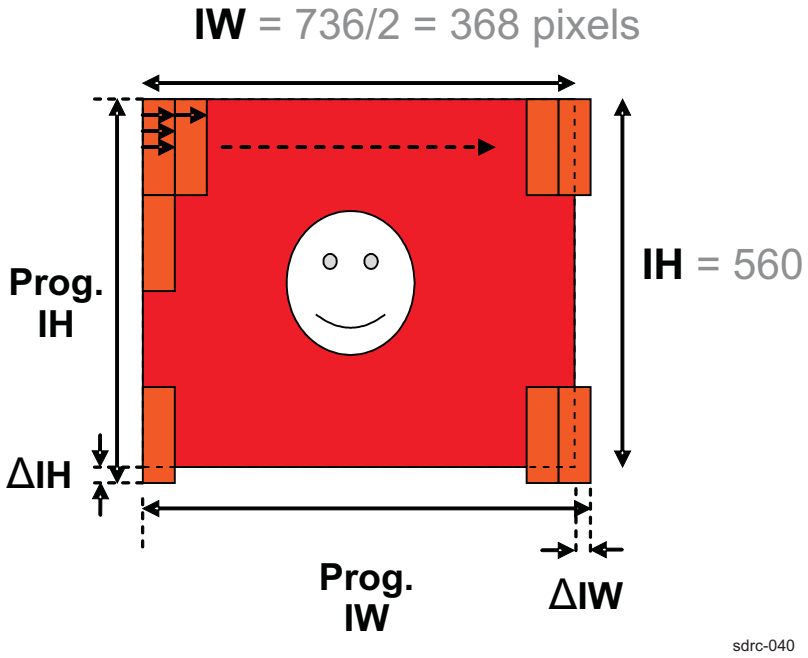
9.2.5.3.3.2 Picture Size Calculation Function

The picture size is configured through two parameters: image width (IMAGEWIDTH) and image height (IMAGEHEIGHT). Two sizes must be distinguished:

- Actual image width and actual image height . This is the size of the image received by the camera subsystem.
- Programmed image width and programmed image height. This is the size of the image programmed in the VRFB.

Figure 9-107 shows the actual image width (IW) and image height (IH) versus the programmed image width (IW + Δ_{IW}) and programmed image height (IH + Δ_{IH}).

Figure 9-107. VRFB Actual Image Size vs Programmed Image Size



The programmed image size parameters are configured into the SMS_ROT_SIZEn registers, where n is the context number (n = 0 to 3):

- Image width: SMS_ROT_SIZEn[10:0] IMAGEWIDTH. This parameter is expressed in the number of pixels.
- Image height: SMS_ROT_SIZEn[26:16] IMAGEHEIGHT. This parameter is expressed in the number of pixels.

The picture size calculation function determines the required number of pages per line and per column using the page size calculated previously. That is a page width of 8 pixels (32 bytes and 4-byte pixel) and a page height of 16 pixels.

1. Determine the required number of pages per line with the formula:
   [(actual_image_width / 2) *pixels* / page_width *bytes*] × pixel_size *bytes/pixel*
   This gives a required number of pages per line of [(736 / 2) / 32] × 4 = 46.
   If the number of pages per line is not an integer, it must be rounded up; for instance, if the result was 46.25 the number of pages required per line would be 47. Figure 9-107 shows this point.
2. Determine the required number of pages per column with the formula:
   (actual_image_height *pixels*) / page_height *lines*
   This gives a required number of pages per column of (560) / 16 = 35.
   If the number of pages per column is not an integer, it must be rounded up.

Finally, calculate the programmed image size in pixels using the number of pages calculated above:

1. Calculate the programmed image width with the formula:
   (rounded_up_number_of_pages_per_line × page_width *bytes*) / pixel_size *bytes/pixel*
   This gives a programmed image width of (46 × 32) / 4 = 368.

The function sets SMS_ROT_SIZEn[10:0] IMAGEWIDTH = 0x170.

2. Calculate the programmed image height with the formula:
   (rounded_up_number_of_pages_per_column × page_height *lines*)
   This gives a programmed image height of (35 × 16) = 560.
   The function sets SMS_ROT_SIZEn[26:16] IMAGEHEIGHT = 0x230.

Because the actual image is a multiple of the page size (both width and height, as explained in Section 9.2.5.3.3.1), the actual image size and the programmed size are the same. In other words, ΔIW and ΔIH are null in this use case.

9.2.5.3.3.3 Physical Base Address Configuration

The physical base address is the address of the picture in the external SDRAM. In this use case, the address space of CS0 is 512 Mbits, ranging from 0x8000 0000 to 0x83FF FFFF because the Mobile DDR SDRAM is a 512 Mbits memory device.

The physical address of the picture is a parameter entered by the user. It is directly written into the SMS_ROT_PHYSICAL_BAn[30:0] PHYSICALBA bit field (where n = 0 to 11, for the 12 contexts).

9.2.5.3.3.4 VRFB Use Case Summarizing Register Values

Table 9-112 summarizes all the VRFB registers to be configured with the required values.

Table 9-112. VRFB Use Case Summarizing Register Print

Register Name	Context Number n	Physical Address	Value	Value Description
SMS_ROT_CONTROLn	0	0x6C00 0180	0x0000 0452	The page size is 8-pixel wide and 16-pixel high. This is a width of 32 bytes and a height of 16 lines. The pixel format is YUV 4:2:2 (2 pixels take 4 bytes).
	1	0x6C00 0190		
	2	0x6C00 01A0		
	3	0x6C00 01B0		
SMS_ROT_SIZEn	0	0x6C00 0184	0x0230 0170	Actual image size is 736 × 560. This gives 368 × 560 with the YUV 4:2:2 pixel format
	1	0x6C00 0194		
	2	0x6C00 01A4		
	3	0x6C00 01B4		
SMS_ROT_PHYSICAL_BAn	0	0x6C00 0188	0x8090 0000	Example of physical base address
	1	0x6C00 0198	0x8100 0000	
	2	0x6C00 01A8	0x8190 0000	
	3	0x6C00 01B8	0x8200 0000	

9.2.5.4 Understanding SDRAM Subsystem Address Spaces

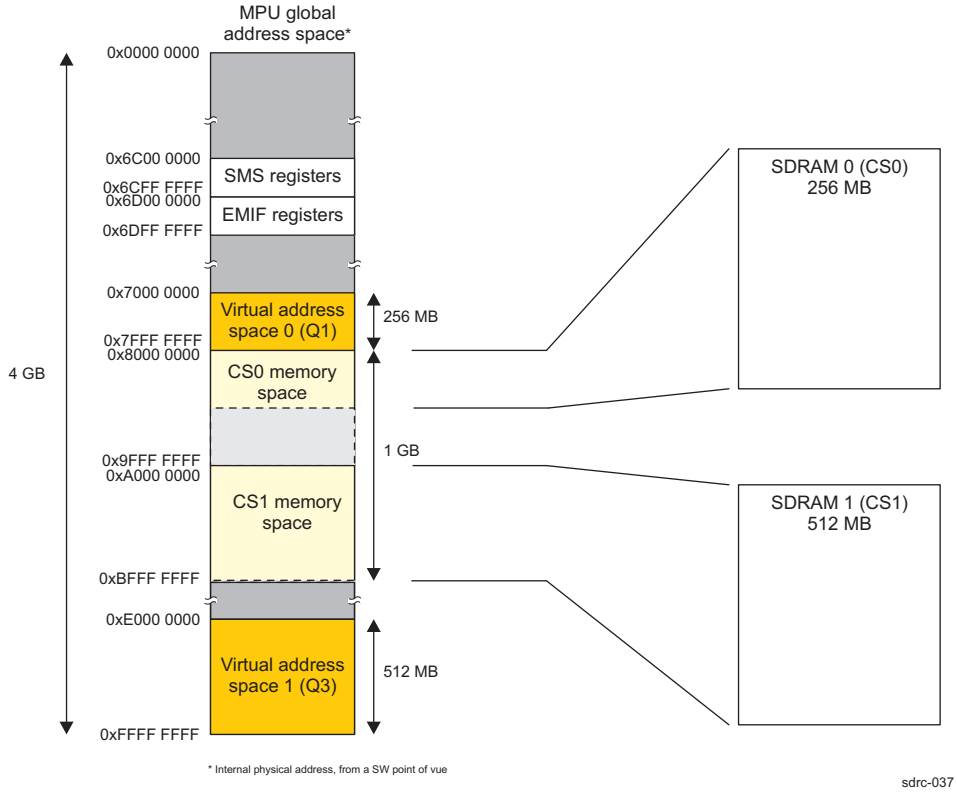
9.2.5.4.1 Physical vs Virtual Address Spaces

One thing that the SMS does is to translate virtual addresses into physical SDRAM addresses in case of rotation only, i.e. when accessing virtual address space 0 (quarter 1) and virtual address space 1 (quarter 3) as depicted in Figure 9-108 . The SMS then reinserts a request, or multiple requests depending on the SMS parameters, in the SMS request path to the EMIF controller (through the VRFB or not).

The SDRAM subsystem global memory space mapping reaches 1.768 GBytes:

- 1 GByte of CS memory space (CS0 and CS1 memory spaces): the EMIF controller automatically accesses the two external memory devices through direct accesses (addresses are simply translated).
- 768 MBytes of virtual address space (address space 0 and address space 1): the EMIF controller automatically accesses the two external memory devices trough re-organized access (requests are modified accordingly to the context number and rotation angle before address translation). See section Section 9.2.5.4.1.2 for more information on VRFB contexts and rotation angles.

Figure 9-108. SDRC Address Space in MPU Global Address Space



Configuration registers space is detailed in the following table:

Table 9-113. EMIF and SMS Configuration Registers Space

Module	Start Address	End Address	Total Space
SMS	0x6C00 0000	0x6CFF FFFF	16MB
EMIF	0x6D00 0000	0x6DFF FFFF	16MB

9.2.5.4.1.1 Physical Address Space

The physical address space of the SDRC is 1 GByte (maximum addressing capability).

The EMIF has a memory device capacity of 16 Mbits to 2 Gbits. 16 Mbits / 2 MBytes is the smallest granularity of supported memory device.

9.2.5.4.1.2 Virtual Address Space

The SMS virtual memory space is a memory space used to access a subset of the EMIF memory space through the rotation engine (Virtual Rotation Frame Buffer). The virtual address space size is 768 MBytes split into two parts: the first 256-MByte part is in the second quarter (Q1) of the memory; the second 512-MByte part is in the fourth quarter (Q3) of the memory. See chapter 2, *Memory Mapping*, for more information on global memory mapping.

The VRFB is a rotation engine that:

- supports rotations of 0, 90, 180, and 270 degrees
- can handle up to 12 concurrent rotation contexts

The VRFB has therefore 48 different contexts in the virtual address space. Table 9-114 gives the VRFB address-space memory locations at which the frame buffer object can be accessed. The table summarizes the virtual addresses of all 48 available image buffer from a global memory space (top level) point of view.

Table 9-114. VRFB Contexts vs Rotation Angle

Context Number	0°	90°	180°	270°
0	0x7000 0000	0x7100 0000	0x7200 0000	0x7300 0000
1	0x7400 0000	0x7500 0000	0x7600 0000	0x7700 0000
2	0x7800 0000	0x7900 0000	0x7A00 0000	0x7B00 0000
3	0x7C00 0000	0x7D00 0000	0x7E00 0000	0x7F00 0000
4	0xE000 0000	0xE100 0000	0xE200 0000	0xE300 0000
5	0xE400 0000	0xE500 0000	0xE600 0000	0xE700 0000
6	0xE800 0000	0xE900 0000	0xEA00 0000	0xEB00 0000
7	0xEC00 0000	0xED00 0000	0xEE00 0000	0xEF00 0000
8	0xF000 0000	0xF100 0000	0xF200 0000	0xF300 0000
9	0xF400 0000	0xF500 0000	0xF600 0000	0xF700 0000
10	0xF800 0000	0xF900 0000	0xFA00 0000	0xFB00 0000
11	0xFC00 0000	0xFD00 0000	0xFE00 0000	0xFF00 0000

The physical address of a page is calculated with the formula:

Physical address = Physical base address + Base address of page (1)

where *Physical base address* is defined through the SMS_ROT_PHYSICAL_BAn[30:0] PHYSICALBA field (buffer physical base address on which the rotation occurs), and *Base address of page* is the address obtain in function of the context number and rotation angle as given in Table 9-114.

9.2.6 SDRAM Memory Scheduler (SMS) Registers

Table 9-115 shows the base address and address space for the SMS module instances.

Table 9-115. SMS Instance Summary

Module Name	Base Address	Size
SMS	0x6C00 0000	64 Kbytes

9.2.6.1 SMS Register Mapping Summary

Table 9-116 summarizes the SMS register mapping.

Table 9-116. SMS Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
SMS_SYSCONFIG	RW	32	0x0000 0010	0x6C00 0010	Section 9.2.6.2.1
SMS_SYSSTATUS	R	32	0x0000 0014	0x6C00 0014	Section 9.2.6.2.2
SMS_RG_ATTi (1)	RW	32	0x0000 0048 + (0x0000 0020 * i)	0x6C00 0048 + (0x0000 0020 * i)	Section 9.2.6.2.3
SMS_RG_RDPERMi (1)	RW	32	0x0000 0050 + (0x0000 0020 * i)	0x6C00 0050 + (0x0000 0020 * i)	Section 9.2.6.2.4
SMS_RG_WRPERMi (1)	RW	32	0x0000 0058 + (0x0000 0020 * i)	0x6C00 0058 + (0x0000 0020 * i)	Section 9.2.6.2.5
SMS_RG_STARTj (2) where k = j - 1	RW	32	0x0000 0060 + (0x0000 0020 * (k)) (3)	0x6C00 0060 + (0x0000 0020 * (k)) (3)	Section 9.2.6.2.6
SMS_RG_ENDj (2) where k = j - 1	RW	32	0x0000 0064 + (0x0000 0020 * (k)) (3)	0x6C00 0064 + (0x0000 0020 * (k)) (3)	Section 9.2.6.2.7
SMS_SECURITY_CONTROL	RW	32	0x0000 0140	0x6C00 0140	Section 9.2.6.2.8
SMS_CLASS_ARBITER0	RW	32	0x0000 0150	0x6C00 0150	Section 9.2.6.2.9
SMS_CLASS_ARBITER1	RW	32	0x0000 0154	0x6C00 0154	Section 9.2.6.2.10
SMS_CLASS_ARBITER2	RW	32	0x0000 0158	0x6C00 0158	Section 9.2.6.2.11
SMS_INTERCLASS_ARBITER	RW	32	0x0000 0160	0x6C00 0160	Section 9.2.6.2.12
SMS_CLASS_ROTATIONm (4)	RW	32	0x0000 0164 + (0x0000 0004 * m)	0x6C00 0164 + (0x0000 0004 * m)	Section 9.2.6.2.13
SMS_ERR_ADDR	R	32	0x0000 0170	0x6C00 0170	Section 9.2.6.2.14
SMS_ERR_TYPE	RW	32	0x0000 0174	0x6C00 0174	Section 9.2.6.2.15
SMS_POW_CTRL	RW	32	0x0000 0178	0x6C00 0178	Section 9.2.6.2.16
SMS_ROT_CONTROLn (5)	RW	32	0x0000 0180 + (0x0000 0010 * n)	0x6C00 0180 + (0x0000 0010 * n)	Section 9.2.6.2.17
SMS_ROT_SIZEn (5)	RW	32	0x0000 0184 + (0x0000 0010 * n)	0x6C00 0184 + (0x0000 0010 * n)	Section 9.2.6.2.18
SMS_ROT_PHYSICAL_BAn (5)	RW	32	0x0000 0188 + (0x0000 0010 * n)	0x6C00 0188 + (0x0000 0010 * n)	Section 9.2.6.2.19

(1) i = 0 to 7.
(2) j = 1 to 7
(3) k = 0 to 6.
(4) m = 0 to 2.
(5) n = 0 to 11.

9.2.6.2 SMS Register Descriptions

9.2.6.2.1 SMS_SYSCONFIG

Table 9-117. SMS_SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x6C00 0010	**Instance**	SMS
Description	This register controls the various parameters of the Interconnect.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							RESERVED	RESERVED			SIDLEMODE		RESERVED	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000000
8	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
7:5	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
4:3	SIDLEMODE	Power management Req/Ack Control 0x0: Force Idle - An idle request is acknowledged unconditionally 0x1: No Idle - An idle request is never acknowledged. 0x2: Smart Idle - Acknowledgment to an idle request is based on the internal activity of the module 0x3: Reserved - Do not use.	RW	0x0
2	RESERVED	Write 0s for future compatibility. Reads return 0.	RW	0x0
1	SOFTRESET	Software reset 0x0: Normal mode (no reset applied) 0x1: Software reset is activated	RW	0x0
0	AUTOIDLE	Internal interface clock gating strategy 0x0: Interface clock is free-running 0x1: Automatic interface clock gating strategy is applied, based on the interconnect activity	RW	0x1

9.2.6.2.2 SMS_SYSSTATUS

Table 9-118. SMS_SYSSTATUS

Address Offset	0x0000 0014		
Physical Address	0x6C00 0014	**Instance**	SMS
Description	This register provides module status, excluding interrupt status info.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RESERVED							RESETDONE

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved for module-specific status information. Read returns 0s.	R	0x000000
7:1	RESERVED	Reserved for interconnect socket status information. Read returns 0s.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset is ongoing. 0x1: Reset complete. The module is ready to be used.	R	0x-

9.2.6.2.3 SMS_RG_ATTi

Table 9-119. SMS_RG_ATTi

Address Offset	0x0000 0048 + (0x0000 0020 * i)	**Index**	i = 0 to 7
Physical Address	0x6C00 0048 + (0x0000 0020 * i)	**Instance**	SMS
Description			
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REQINFO																															

Bits	Field Name	Description	Type	Reset
31:0	REQINFO	Request information permission The REQINFO field is a bit vector of permissions, one per MReqInfo encoding: NonHost/Host - User/Supervisor - Public/Secure - Functional/Debug - Data Transfer/Opcode Fetch (1)	RW	0x--------

(1) See Table 9-62 for more information on REQINFO values

9.2.6.2.4 SMS_RG_RDPERMi

Table 9-120. SMS_RG_RDPERMi

Address Offset	0x0000 0050 + (0x0000 0020 * i)	**Index**	i = 0 to 7
Physical Address	0x6C00 0050 + (0x0000 0020 * i)	**Instance**	SMS
Description	This register provides the list of all initiators that have permission for reading from that memory region.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CONNIDVECTOR															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0000
15:0	CONNIDVECTOR	One bit per initiator group. Bit 0 set to 1 means that initiator whose ConnID = 0 has read permission to the protected region i.	RW	0x---- (1)

(1) Reset value exported from control module for region 0 and region 1; reset value equal to 0x0000 for other

9.2.6.2.5 SMS_RG_WRPERMi

Table 9-121. SMS_RG_WRPERMi

Address Offset	0x0000 0058 + (0x0000 0020 * i)	**Index**	i = 0 to 7
Physical Address	0x6C00 0058 + (0x0000 0020 * i)	**Instance**	SMS
Description	This register provides the list of all initiators that have permission for writing to that memory region.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																CONNIDVECTOR															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0000
15:0	CONNIDVECTOR	One bit per initiator group. Bit 0 set to 1 means that initiator whose ConnID = 0 has write permission to the protected region i.	RW	0x---- (1)

(1) Reset value exported from control module for region 0 and region 1; reset value equal to 0x0000 for other

9.2.6.2.6 SMS_RG_STARTj

Table 9-122. SMS_RG_STARTj

Address Offset	0x0000 0060 + (0x0000 0020 * (k))	**Index**	k = 0 to 6
Physical Address	0x6C00 0060 + (0x0000 0020 * (k))	**Instance**	SMS
Description	This register provides the region #j start address (lowest address inside the region), with a 64-KB granularity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	STARTADDRESS															RESERVED															

Bits	Field Name	Description	Type	Reset
31	RESERVED	Write 0s for future compatibility. Reads return 0.	RW	0x0
30:16	STARTADDRESS	Region #j start address (included in the region) Aligned on 64-KB boundary. [15:0] must be written with 0s. No STARTADDRESS parameter for region 0.	RW	0x----
15:0	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0000

9.2.6.2.7 SMS_RG_ENDj

Table 9-123. SMS_RG_ENDj

Address Offset	0x0000 0064 + (0x0000 0020 * (k))	**Index**	k = 0 to 6
Physical Address	0x6C00 0064 + (0x0000 0020 * (k))	**Instance**	SMS
Description	This register provides the region #j end address (lowest address outside the region), with a 64-KB granularity.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	ENDADDRESS															RESERVED															

Bits	Field Name	Description	Type	Reset
31	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
30:16	ENDADDRESS	Region #j end address (not included in the region) Aligned on 64-KB boundary. [15:0] must be written with 0s. No ENDADDRESS parameter for region 0.	RW	0x----
15:0	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0000

9.2.6.2.8 SMS_SECURITY_CONTROL

Table 9-124. SMS_SECURITY_CONTROL

Address Offset	0x0000 0140		
Physical Address	0x6C00 0140	**Instance**	SMS
Description	This register provides the security level required to access all SMS registers.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				ROTCTXT11LOCK	ROTCTXT10LOCK	ROTCTXT9LOCK	ROTCTXT8LOCK	ROTCTXT7LOCK	ROTCTXT6LOCK	ROTCTXT5LOCK	ROTCTXT4LOCK	ROTCTXT3LOCK	ROTCTXT2LOCK	ROTCTXT1LOCK	ROTCTXT0LOCK	RESERVED										ARBITRATIONREGSLOCK	REGION1REGSLOCK	SOFTRESETLOCK	ERRORREGSLOCK	FIREWALLLOCK	SECURITYCONTROLREGLOCK

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
27	ROTCTXT11LOCK	Sets the security level to program rotation context 11 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
26	ROTCTXT10LOCK	Sets the security level to program rotation context 10 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
25	ROTCTXT9LOCK	Sets the security level to program rotation context 9 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
24	ROTCTXT8LOCK	Sets the security level to program rotation context 8 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
23	ROTCTXT7LOCK	Sets the security level to program rotation context 7 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
22	ROTCTXT6LOCK	Sets the security level to program rotation context 6 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
21	ROTCTXT5LOCK	Sets the security level to program rotation context 5 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
20	ROTCTXT4LOCK	Sets the security level to program rotation context 4 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
19	ROTCTXT3LOCK	Sets the security level to program rotation context 3 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0

Bits	Field Name	Description	Type	Reset
18	ROTCTXT2LOCK	Sets the security level to program rotation context 2 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
17	ROTCTXT1LOCK	Sets the security level to program rotation context 1 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
16	ROTCTXT0LOCK	Sets the security level to program rotation context 0 0x0: Any transaction is allowed. 0x1: Secure transaction required	RW	0x0
15:6	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000
5	ARBITRATIONREGSLOCK	Sets the security level to program arbitration control registers 0x0: Any transaction is allowed. 0x1: Secure privilege transaction required	RW	0x0
4	REGION1REGSLOCK	Region 1 security firewall registers lock bit 0x0: Region1 configuration registers can be accessed by all accesses (R/W) but depend on Full Secure Program value 0x1: Region1 configuration registers are restricted to secure supervisor accesses only (R/W) and do not depend on Full Secure Program value	RW	0x- (1)
3	SOFTRESETLOCK	Soft reset lock bit 0x0: The SMS soft reset can be triggered with any access 0x1: The SMS soft reset can be triggered only with secure supervisor accesses. When this bit is set to 1, a non secure soft reset has no effect on SMS module	RW	0x- (1)
2	ERRORREGSLOCK	Error registers lock bit 0x0: The SMS_ERR_TYPE and SMS_ERR_ADDR registers can be read and cleared with any access 0x1: The SMS_ERR_TYPE and SMS_ERR_ADDR registers can be read and cleared only with secure supervisor accesses	RW	0x- (1)
1	FIREWALLLOCK	All security firewall registers lock bit 0x0: The SMS firewall registers can be programmed and read with any access 0x1: The SMS firewall registers can be programmed and read only with secure supervisor accesses	RW	0x- (1)
0	SECURITYCONTROLREGLOCK	SMS_SECURITY_CONTROL register configuration lock bit 0x0: The SMS_SECURITY_CONTROL register is unlocked 0x1: The SMS_SECURITY_CONTROL register is locked to secure supervisor access only. Only a secure supervisor access or component reset can reset this bit to 0	RW	0x- (1)

(1) Reset value exported from control module

9.2.6.2.9 SMS_CLASS_ARBITER0

Table 9-125. SMS_CLASS_ARBITER0

Address Offset	0x0000 0150		
Physical Address	0x6C00 0150	**Instance**	SMS
Description	This register controls the arbitration parameters between the class 0 request groups.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BURST-COMPLETE		RESERVED						EXTENDEDGRANT				RESERVED												HIGHPRIOVECTOR		RESERVED					

Bits	Field Name	Description	Type	Reset
31:30	BURST-COMPLETE	Delayed service until burst request complete BurstComplete[k], k= 6 to 7 (BURST-COMPLETE[30] is for group number 6, BURST-COMPLETE[31] is for group number 7) 0x0: Group #k request to arbiter issued as soon as the first burst request is available 0x1: Group #k request to arbiter delayed until a complete burst transaction is buffered	RW	0x0
29:24	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
23:20	EXTENDEDGRANT	Extended grant service inside a class Vector specifying the number of consecutive services a group is granted. 2 bits per group ExtendedGrant[2*k+1,2*k], k = 6 to 7 (EXTENDEDGRANT[21:20] is for group number 6, EXTENDEDGRANT[23:22] is for group number 7) 0x1: 1 service for group #k when granted 0x2: 2 services for group #k when granted 0x3: 3 services for group #k when granted	RW	0x5
19:8	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000
7:6	HIGHPRIOVECTOR	High-priority attribute inside a class Vector allocating a higher priority to one of the class members. A single group may be given this attribute at a time. HighPrioVector[k], k= 6 to 7 (HIGHPRIOVECTOR[6] is for group number 6, HIGHPRIOVECTOR[7] is for group number 7) 0x0: Group #k has standard priority (LRU based). 0x1: Group #k has the highest priority over all other class members.	RW	0x0
5:0	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00

9.2.6.2.10 SMS_CLASS_ARBITER1

Table 9-126. SMS_CLASS_ARBITER1

Address Offset	0x0000 0154		
Physical Address	0x6C00 0154	**Instance**	SMS
Description	This register controls the arbitration parameters between the class 1 request groups.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						BURST-COMPLETE		RESERVED								EXTENDEDGRANT				RESERVED										HIGHPRIOVECTOR	

Bits	Field Name	Description	Type	Reset
31:26	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
25:24	BURST-COMPLETE	Delayed service until burst request complete BurstComplete[k], k= 0 to 1 (BURST-COMPLETE[24] is for group number 0, BURST-COMPLETE[25] is for group number 1) 0x0: Group #k request to arbiter issued as soon as the first burst request is available 0x1: Group #k request to arbiter delayed until a complete burst transaction is buffered	RW	0x0
23:12	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000
11:8	EXTENDEDGRANT	Extended grant service inside a class Vector specifying the number of consecutive services a group is granted. 2 bits per group ExtendedGrant[2*k+1,2*k], k = 0 to 1 (EXTENDEDGRANT[9:8] is for group number 0, EXTENDEDGRANT[11:10] is for group number 1) 0x1: 1 service for group #k when granted 0x2: 2 services for group #k when granted 0x3: 3 services for group #k when granted	RW	0x5
7:2	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
1:0	HIGHPRIOVECTOR	High-priority attribute inside a class Vector allocating a higher priority to one of the class members. A single group may be given this attribute at a time. HighPrioVector[k], k= 0 to 1 (HIGHPRIOVECTOR[1] is for group number 1, HIGHPRIOVECTOR[0] is for group number 0) 0x0: Group #k has standard priority (LRU based). 0x1: Group #k has the highest priority over all other class members.	RW	0x0

9.2.6.2.11 SMS_CLASS_ARBITER2

Table 9-127. SMS_CLASS_ARBITER2

Address Offset	0x0000 0158		
Physical Address	0x6C00 0158	**Instance**	SMS
Description	This register controls the arbitration parameters between the class 2 request groups.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		BURST-COMPLETE				RESERVED						EXTENDEDGRANT				RESERVED										HIGHPRIOVECTOR				RESERVED	

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
29:26	BURST-COMPLETE	Delayed service until burst request complete BurstComplete[k], k= 2 to 5 (BURST-COMPLETE[29] is for group number 5, BURST-COMPLETE[28] is for group number 4, BURST-COMPLETE[27] is for group number 3, BURST-COMPLETE[26] is for group number 2) 0x0: Group #k request to arbiter issued as soon as the first burst request is available 0x1: Group #k request to arbiter delayed until a complete burst transaction is buffered	RW	0x0
25:20	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
19:12	EXTENDEDGRANT	Vector specifying the number of consecutive services a group is granted. 2 bits per group ExtendedGrant[2*k+1,2*k], k = 2 to 5 (EXTENDEDGRANT[19:18] is for group number 5, EXTENDEDGRANT[17:16] is for group number 4, EXTENDEDGRANT[15:14] is for group number 3, EXTENDEDGRANT[13:12] is for group number 2) 0x1: 1 service for group #k when granted 0x2: 2 services for group #k when granted 0x3: 3 services for group #k when granted	RW	0x55
11:6	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
5:2	HIGHPRIOVECTOR	Vector allocating a higher priority to one of the class members. A single group may be given this attribute at a time. HighPrioVector[k], k= 2 to 5 (HIGHPRIOVECTOR[5] is for group number 5, HIGHPRIOVECTOR[4] is for group number 4, HIGHPRIOVECTOR[3] is for group number 3, HIGHPRIOVECTOR[2] is for group number 2) 0x0: Group #k has standard priority (LRU based). 0x1: Group #k has the highest priority over all other class members.	RW	0x0
1:0	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0

9.2.6.2.12 SMS_INTERCLASS_ARBITER

Table 9-128. SMS_INTERCLASS_ARBITER

Address Offset	0x0000 0160		
Physical Address	0x6C00 0160	**Instance**	SMS
Description	This register controls the PWM counter that defines the priority alternation between class 1 and class 2.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CLASS2PRIO								RESERVED								CLASS1PRIO							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
23:16	CLASS2PRIO	Class 2 high-priority window width (clock cycle count). Do not set to 0x00.	RW	0x40
15:8	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
7:0	CLASS1PRIO	Class 1 high-priority window width (clock cycle count). Do not set to 0x00.	RW	0x40

9.2.6.2.13 SMS_CLASS_ROTATIONm

Table 9-129. SMS_CLASS_ROTATIONm

Address Offset	0x0000 0164 + (0x0000 0004 * m)	**Index**	m = 0 to 2
Physical Address	0x6C00 0164 + (0x0000 0004 * m)	**Instance**	SMS
Description	This register controls the number of consecutive services that is allocated to a thread whose transactions have been split by the rotation engine.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											NOFSERVICES				

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0000000
4:0	NOFSERVICES	Number of RE split transactions serviced consecutively when the thread gets granted by the arbitration logic.	RW	0x01

9.2.6.2.14 SMS_ERR_ADDR

Table 9-130. SMS_ERR_ADDR

Address Offset	0x0000 0170		
Physical Address	0x6C00 0170	**Instance**	SMS
Description	This register captures the address of an access that has generated an error.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRORADDRESS																															

Bits	Field Name	Description	Type	Reset
31:0	ERRORADDRESS	Access address that has generated an error (bit 31 is always 0)	R	0x00000000

9.2.6.2.15 SMS_ERR_TYPE

Table 9-131. SMS_ERR_TYPE

Address Offset	0x0000 0174		
Physical Address	0x6C00 0174	**Instance**	SMS
Description	This register provides additional information about the access that has generated the error.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					ERRORREGIONID			RESERVED	ERRORMCMD			ERRORCONNID				RESERVED					UNEXPECTEDADD	UNEXPECTEDREQ	ILLEGALCMD	RESERVED				ERRORSECOVERLAP	ERRORSECREG	ERRORSECURITY	ERRORVALID

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
26:24	ERRORREGIONID	ID of the region that has been illegally accessed 0x0: Region 0 0x1: Region 1 ... 0x7: Region 7 0x8 to 0xF: reserved	R	0x0
23	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
22:20	ERRORMCMD	Interconnect command that caused the error	R	0x0
19:16	ERRORCONNID	Identifies the illegal access initiator interconnect ConnID of the illegal access initiator. Refer to the top level documentation of the device using the SMS module	R	0x0
15:11	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00

Bits	Field Name	Description	Type	Reset
10	UNEXPECTEDADD	Request targeting non-defined rotation contexts (such as contexts 12, 13, 14, or 15) or non-defined L3 Interconnect request signals used for context number decoding. Read 0x0: No unexpected request received Write 0x0: No effect Read 0x1: A request has been received on the interconnect with address decoding targeting rotation contexts 12, 13, 14, or 15, or a signal used for context number decoding was undefined. Write 0x1: Clear UNEXPECTEDADD bit.	RW	0x0
9	UNEXPECTEDREQ	Unexpected request received during SMS idle state Read 0x0: No unexpected request received Write 0x0: No effect Read 0x1: A request has been received on the interconnect after the SMS was put in idle mode by the system power manager. Write 0x1: Clear the UNEXPECTEDREQ bit field.	RW	0x0
8	ILLEGALCMD	Illegal command on the L3 interface Read 0x0: No illegal command received Write 0x0: No effect Read 0x1: Illegal command has been received. Write 0x1: Clear ILLEGALCMD bit field.	RW	0x0
7:4	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
3	ERRORSECOVERLAP	Protection region overlapping error Read 0x0: No overlap violation detected Write 0x0: No effect Read 0x1: A protection region overlap violation has been detected. Write 0x1: Clear ERRORSECOVERLAP bit field.	RW	0x0
2	ERRORSECREG	SMS security register accessed by nonsecure write transaction Read 0x0: No violation detected on secure registers Write 0x0: No effect Read 0x1: A nonsecure write access to the security control registers has been detected. Such writes are not allowed (reads are always allowed). 0x1: Clear ERRORSECREG bit field	RW	0x0
1	ERRORSECURITY	Security violation error Read 0x0: No illegal command received Write 0x0: No effect Read 0x1: Security violation detected Write 0x1: Clear ERRORSECURITY bit field.	RW	0x0
0	ERRORVALID	Error validity status - Must be explicitly cleared with a write transaction Read 0x0: No effect Write 0x0: All error fields no longer valid Read 0x1: Error detected and logged in the other error fields Write 0x1: Clear ERRORVALID bit field	RW	0x0

9.2.6.2.16 SMS_POW_CTRL

Table 9-132. SMS_POW_CTRL

Address Offset	0x0000 0178		
Physical Address	0x6C00 0178	**Instance**	SMS
Description	This register controls the SMS power management in conjunction with the regular interconnect socket registers.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								IDLEDELAY							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000000
7:0	IDLEDELAY	Delay (expressed in L3 clock cycle units) before autoidle, that is, before disabling the SMS functional clock when no more traffic in the SMS module.	RW	0x80

9.2.6.2.17 SMS_ROT_CONTROLn

Table 9-133. SMS_ROT_CONTROLn

Address Offset	0x0000 0180 + (0x0000 0010 * n)	**Index**	n = 0 to 11
Physical Address	0x6C00 0180 + (0x0000 0010 * n)	**Instance**	SMS
Description	This register configures the virtual rotated frame buffer module for context #n.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					PH			RESERVED	PW			RESERVED		PS	

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x000000
10:8	PH	Exponent based 2 value, 2^{ph} indicates the page height in bytes for context #n.	RW	0x0
7	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
6:4	PW	Exponent based 2 value, 2^{pw} indicates the page width in bytes for context #n.	RW	0x0
3:2	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
1:0	PS	Exponent based 2 value, 2^{ps} indicates the pixel size in bytes for context #n. A value of 3 is invalid.	RW	0x0

9.2.6.2.18 SMS_ROT_SIZEn

Table 9-134. SMS_ROT_SIZEn

Address Offset	0x0000 0184 + (0x0000 0010 * n)	**Index**	n = 0 to 11
Physical Address	0x6C00 0184 + (0x0000 0010 * n)	**Instance**	SMS
Description	This register configures the bank organization for context #n.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								IMAGEHEIGHT								RESERVED								IMAGEWIDTH							

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
26:16	IMAGEHEIGHT	Image height in pixels for context #n	RW	0x000
15:11	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x00
10:0	IMAGEWIDTH	Image width in pixels for context #n	RW	0x000

9.2.6.2.19 SMS_ROT_PHYSICAL_BAn

Table 9-135. SMS_ROT_PHYSICAL_BAn

Address Offset	0x0000 0188 + (0x0000 0010 * n)	**Index**	n = 0 to 11
Physical Address	0x6C00 0188 + (0x0000 0010 * n)	**Instance**	SMS
Description	This register allows to configure the physical base address for context #n.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	PHYSICALBA																														

Bits	Field Name	Description	Type	Reset
31	RESERVED	Write 0s for future compatibility. Read returns 0s.	RW	0x0
30:0	PHYSICALBA	Physical base address of the frame buffer for context #n in SDRAM	RW	0x00000000

9.3 On-Chip Memory (OCM) Subsystem

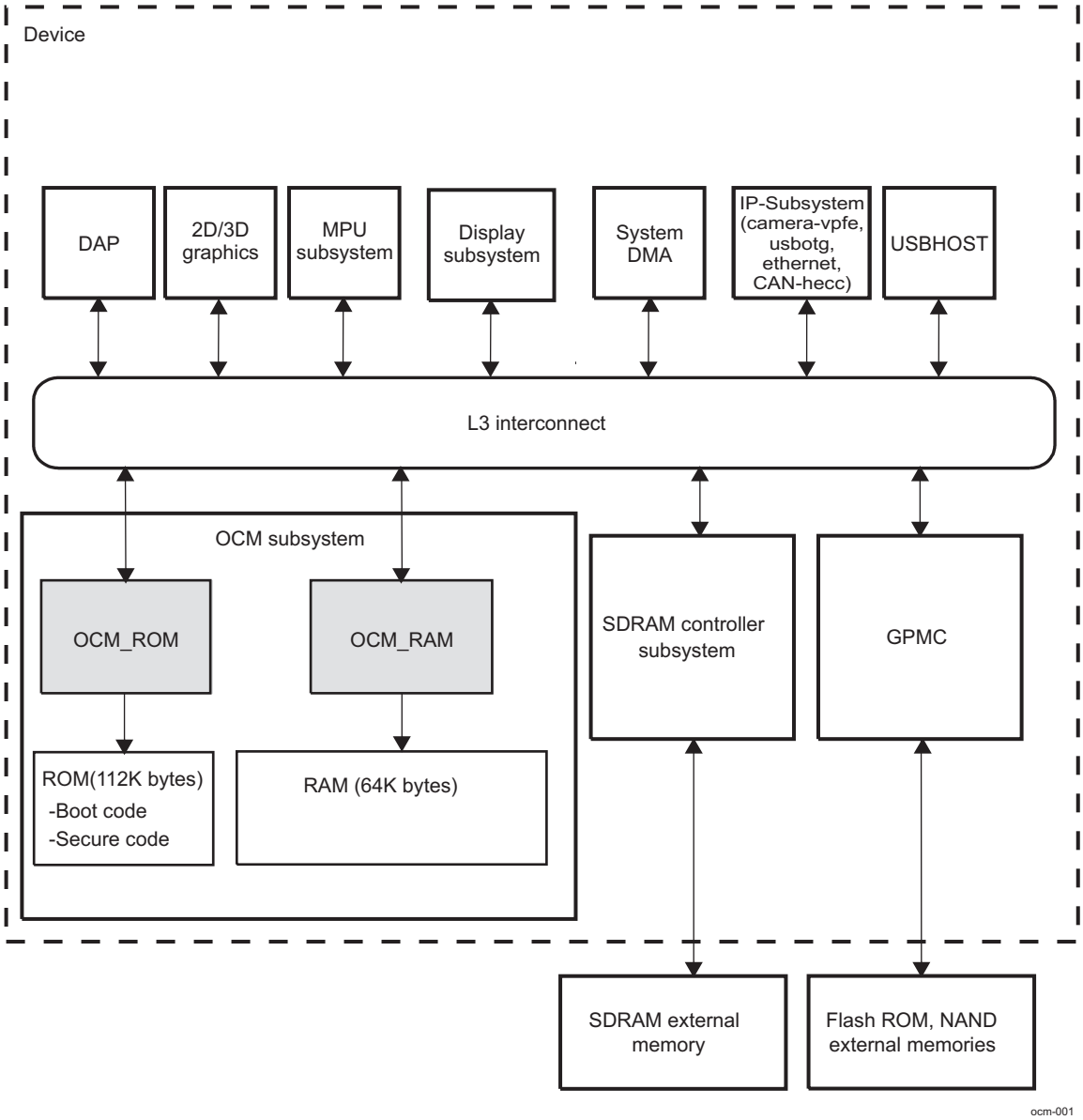
9.3.1 OCM Subsystem Overview

The on-chip memory subsystem consists of two separate on-chip memory controllers, one connected to an on-chip ROM (OCM_ROM) and the other connected to an on-chip RAM (OCM_RAM). Each memory controller has its own dedicated interface to the L3 interconnect.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

Figure 9-109 is an overview of the OCM subsystem.

Figure 9-109. OCM Subsystem Overview



Multiple L3 initiators (such as remote devices) have access to the RAM through 2D/3D graphics, the MPU subsystem, sDMA, the IP subsystem, the display subsystem, and USBHOST.

ROM is used for direct boot code, boot from external NAND flash, and secure code.

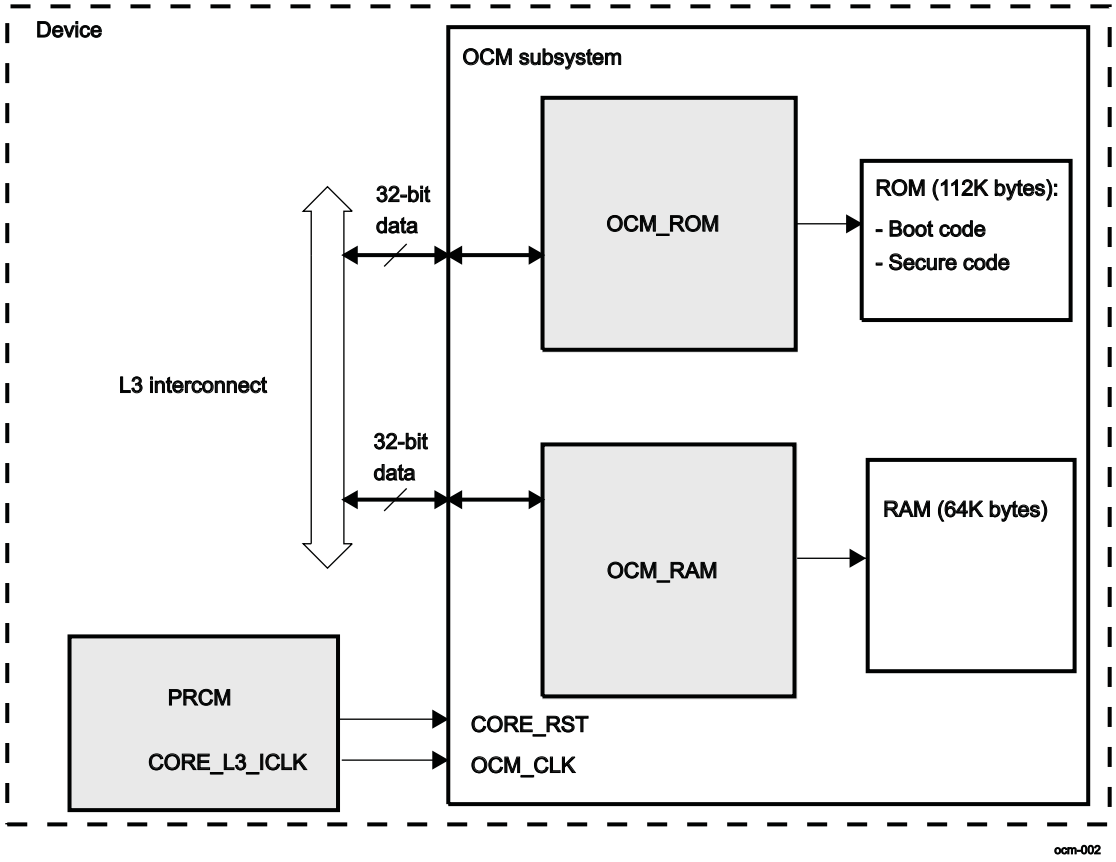
9.3.2 OCM Subsystem Integration

9.3.2.1 Description

The OCM_ROM and OCM_RAM allow transactions between the system initiators and the multiple memories, through the L3 interconnect.

Figure 9-110 shows the integration of the OCM subsystem to the processor.

Figure 9-110. OCM Subsystem Integration to the Device



9.3.2.2 Clocking, Reset, and Power-Management Scheme

9.3.2.2.1 Clocking

The on-chip boot ROM and RAM are clocked only when they are accessed.

The interface clock OCM_CLK comes from the PRCM module and runs at the L3 interconnect frequency. The OCM_CLK source is PRCM CORE_L3_ICLK output. This clock is also used as the functional clock for the OCM module.

For the OCM subsystem, no register enables the gating of OCM_CLK.

However, OCM does support the handshaking protocol with the PRCM module. It is not programmable by software. The following signals support the handshaking protocol for ROM and RAM devices:

- OCMROM_IDLEREQ/OCMROM_SIDLEACK (for ROM devices)
- OCMRAM_IDLEREQ/OCMRAM_SIDLEACK (for RAM devices)

OCM_CLK is gated if all modules (including the OCM) belonging to the L3 CLK domain send a SIdleAck back to the PRCM module after reception of the MIdleReq request.

For details, see the *Power, Reset, and Clock Management* chapter.

When the memory is not accessed by the system, the module performs automatic clock gating. Because the clock to the memory is dynamically gated, there is no extra latency when the clock must be switched on after an idle state.

9.3.2.2.2 Hardware Reset

Global reset of the module is performed by activation of the CORE_RST in the core reset domain (see the *Power, Reset, and Clock Management* chapter).

9.3.2.2.3 Power Domain

OCM power is supplied by the CORE power domain (see the *Power, Reset, and Clock Management* chapter).

9.3.3 OCM Subsystem Functional Description

9.3.3.1 OCM_ROM

The embedded ROM is used primarily for booting, flashing, and context restoring.

The device-embedded ROM (total 32K bytes) has the following characteristics:

- The ROM contains the boot area.
- The OCM_ROM supports single and burst access transactions.
- The OCM_ROM operates at full interconnect clock frequency.
- The COM_ROM needs three cycles for initial access and one cycle per subsequent access.

The memory space of the embedded ROM starts at 0x4001 4000 and ends at 0x4001 BFFF.

9.3.3.2 OCM_RAM

By default, only 2K bytes are nonsecure after reset; however, the configuration can then be changed to adapt to booting/flashing, normal boot, or to any application requirement.

The device-embedded RAM has the following characteristics:

- Operates at full L3 interconnect clock frequency
- Fully pipelined, one 32-bit access per cycle
- Restricted access support, based on:
  - A region-based partitioning (see the L3 firewall description)
  - The module owner of the access, with respect to its read and write permission to that region
  - The transaction attributes of the access, with respect to the region permission properties:
    - Secure/nonsecure
    - User/supervisor
    - Code/data access

The OCM_RAM can be partitioned using the L3 firewall (see the *Interconnect* chapter) and used as:

- Public RAM for normal RAM, or
- Secure RAM:
  - For secure data and instructions
  - For secure Level 2 data and instruction cache:
    - Secure stack
    - Secure global data
    - Secure heap
    - Secure decrypted applications
    - Crypto keys

The OCM_RAM can be partitioned using the L3 firewall and used as:

- Public RAM for a video frame buffer
- Secure RAM for a secure video frame buffer
- Secure RAM for computing heavy DRM applications such as real-time video decryption
- Secure RAM for any application

The RAM memory space starts at 0x4020 0000 and ends at 0x4020 FFFF.

9.4 Revision History

Table 9-136 lists the changes made since the previous version of this document.

Table 9-136. Document Revision History

Reference	Additions/Modifications/Deletions
Section 9.2.3.4.1.1	Added LPDDR1 information to tables and list.
Section 9.2.3.4.2.6.1	Added subsection.
Figure 9-48	Added table.
Table 9-76	Updated bit descriptions.
Section 9.2.3.4.3.2.4	Added REG_PASR bit.
Section 9.2.3.4.3.2.5	Added REG_INITREF_DIS and REG_PASR bits.
Table 9-81	Updated bit descriptions.
Section 9.2.3.4.5.1	Added last bulleted list.
Section 9.2.3.4.5.2	Added last bulleted list.
Section 9.2.3.4.5.3	Added subsection.

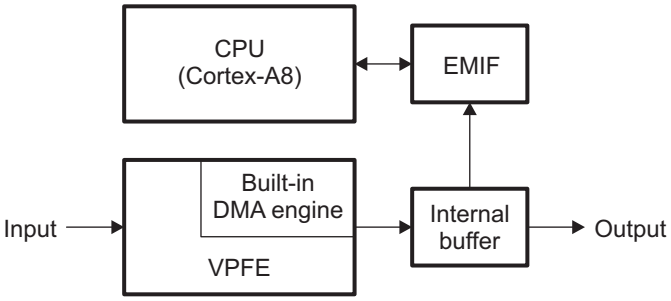
Video Processing Front End (VPFE)

10.1 Overview

The video processing subsystem (VPSS) includes a video processing front-end (VPFE) controller, which is the video input portion of the processor. The VPFE controller receives input video/image data from external capture devices and stores it to external memory. A built-in DMA engine transfers the capture data into the external memory. An internal buffer block provides a high bandwidth path between the VPSS module and the external memory. The CPU, Cortex-A8, will process the image data based on application requirements.

A simple LCD controller is available. Reference LCD chapter. Figure 10-1 illustrates the high-level block diagram of the VPSS module.

Figure 10-1. VPSS Module, Memory, and the CPU



10.1.1 Features

The VPFE controller supports the following features:

- It supports conventional Bayer pattern and Foveon sensor formats.
- It is flexible in synchronization timing generation. It can be programmed to synchronize to the external horizontal/vertical sync and field ID signals with various timings.
- It supports progressive and interlaced sensors (hardware support for up to two fields and firmware support for a higher number of fields, typically 3, 4, and 5-field sensors).
- The max pixel clock is 75 MHz.
- It supports the REC656/CCIR-656 standard.
- It supports YCbCr 422 format, either 8-bit or 16-bit with discrete horizontal and vertical sync signals.
- The input capture data can be up to 16 bits.
- It can generate optical black clamping signals.
- It has built-in digital clamping and black level compensation.
- 10-bit to 8-bit A-law compression hardware is provided.
- It has a low-pass filter that can be applied prior to writing the capture data to external memory. If this filter is enabled, two pixels in each the left and the right edges of each line are cropped from the output.
- The output data can range from 16 bits to 8 bits wide (8-bit width allows for a 50% savings in memory).
- It has a programmable culling block that can perform down-sampling of the input data.
- An external write enable signal is provided to control the timing of outputting data to the external memory.
- It supports up to 16K pixels (image size) in both the horizontal and vertical directions.

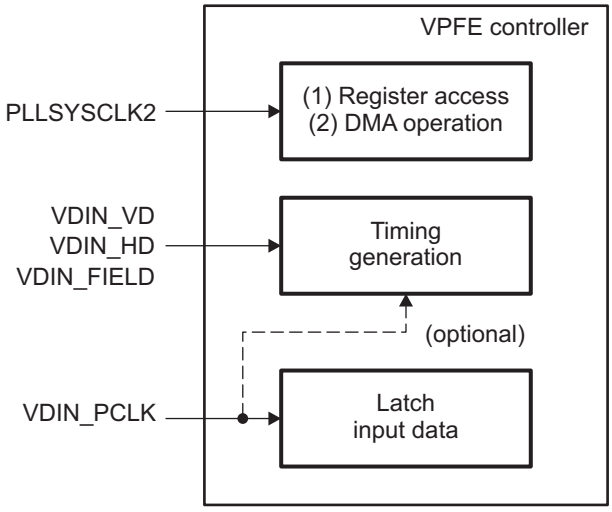
10.2 VPFE Controller - The System View

10.2.1 Clocks

The VFPE controller requires two clock sources:

- Internal clock source - PLLSYSCLK2. This clock is the reference clock used by the VPFE internal logic, including the built-in DMA operation.
- External clock source – External pixel clock signal VDIN_PCLK, which is used to generate internal synchronization signals to latch input data.

Figure 10-2. Clock Sources



10.2.2 Reset

The power, reset and clock management module can reset the VPFE controller. See the *Power, Reset and Clock Management* chapter for more details.

10.2.3 Interrupts

The VPFE controller can generate three interrupts to the CPU, as shown in Table 10-1.

Table 10-1. ARM Interrupts - VPFE

Interrupt Number	Acronym	Source
M_IRQ_88	CCDC_VD0_INT	VPFE
M_IRQ_92	CCDC_VD1_INT	VPFE
M_IRQ_93	CCDC_VD2_INT	VPFE

10.3 Functional Description

10.3.1 External IO Interface

The following subsections explains how to connect VPFE signal pins to various input devices.

10.3.1.1 Raw Data Mode

Raw data mode is a generic parallel interface that supports up to a 16-bit data path to a CMOS or CCD sensor. The signal interface is described in Table 10-2.

Table 10-2. CCD Interface Signals

Name	I/O	Function
VDIN_D[15:0]	I	Image data – A mode set by INPMOD (not R656ON). • Bit width is configurable between 8 and 16 bits (DATSIZ). • The polarity of the input image data is configurable (DATAPOL).
VDIN_VD	I	VSYNC - vertical sync signal
VDIN_HD	I	HSYNC - horizontal sync signal
VDIN_FIELD	I	Field identification signal (optional – FLDMODE) • This signal can be configured to be latched by the VD signal (FIDMD). • The polarity of the field identification signal is configurable (FLDPOL).
VDIN_WEN	I	VPFE write enable signal (optional – EXWEN) • The EXWEN signal determines when data is captured, processed, and saved to memory or sent for further processing. • If EXWEN is enabled, image data will be captured, processed, and saved to memory or sent for further processing, depending on the state of WENLOG. • Data can be saved either when VDIN_WEN is active and the pixels are within the internal frame (SPH, NPH, SLV, NLV) or when the pixels are within the internal frame (WENLOG).
VDIN_PCLK	I	Pixel clock • The PCLK signal is the signal used to latch input image data. • The input image data can be captured on either the rising or on the falling edge of the PCLK signal which is configured by field PCLK_INV. • The maximum pixel clock rate is 75 MHz.

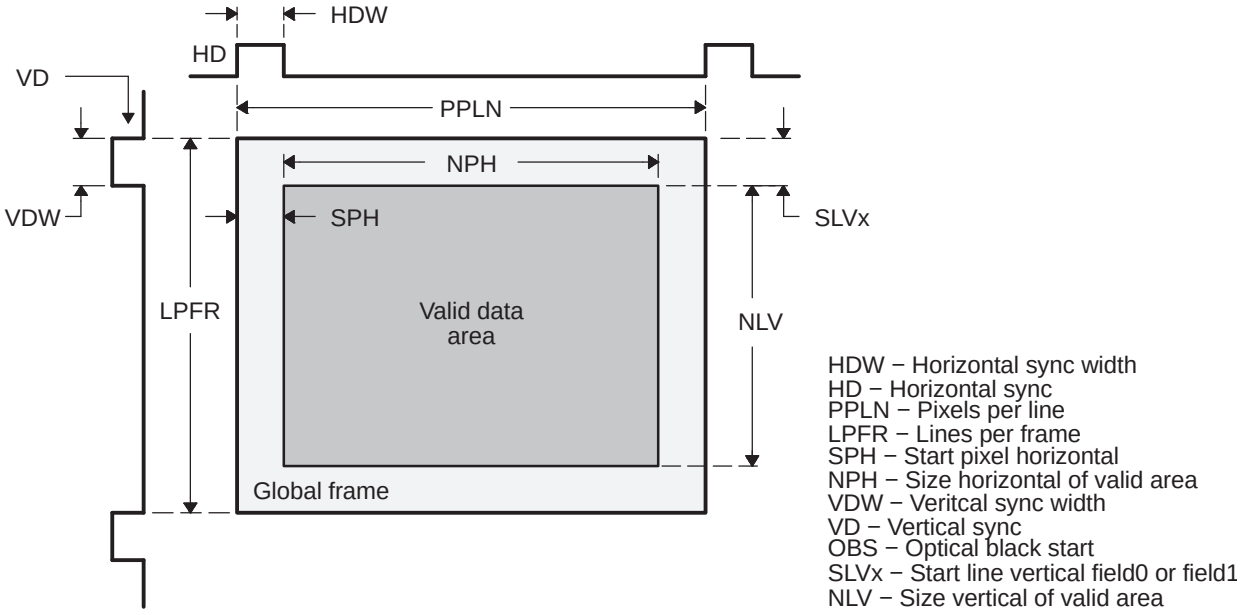
10.3.1.1.1 Mode Information – Always Required

- INPMOD – input mode
- DATSIZ – size (bit width) of input data – always stored in LSBs
- DATAPOL – polarity of input data
- VDPOL – VD polarity
- HDPOL – HD polarity
- FLDMOD – field mode

10.3.1.1.2 Timing Information – Optional, Depending on Control Signals and Sensor Mode

- If FLDMODE is enable
  - FLDPOL – VDIN_FIELD polarity
  - FIDMD – VDIN_FIELD latch information
- EXWEN – external VDIN_WEN signal
  - WENLOG – determines when data is valid along with frame settings

Figure 10-3. CCD Controller Frame and Control Signal Definitions



10.3.1.2 ITU-R BT.656 Interface

The BT.656 interface supports either 8-bit or 10-bit processing of input video YCbCr data. See Section 10.4 for instructions on how to configure this mode.

Since the data synchronization information is carried along with the data lines, no synchronization signals (i.e., VDIN_HD, VDIN_VD, and VDIN_FIELD) are necessary in this mode.

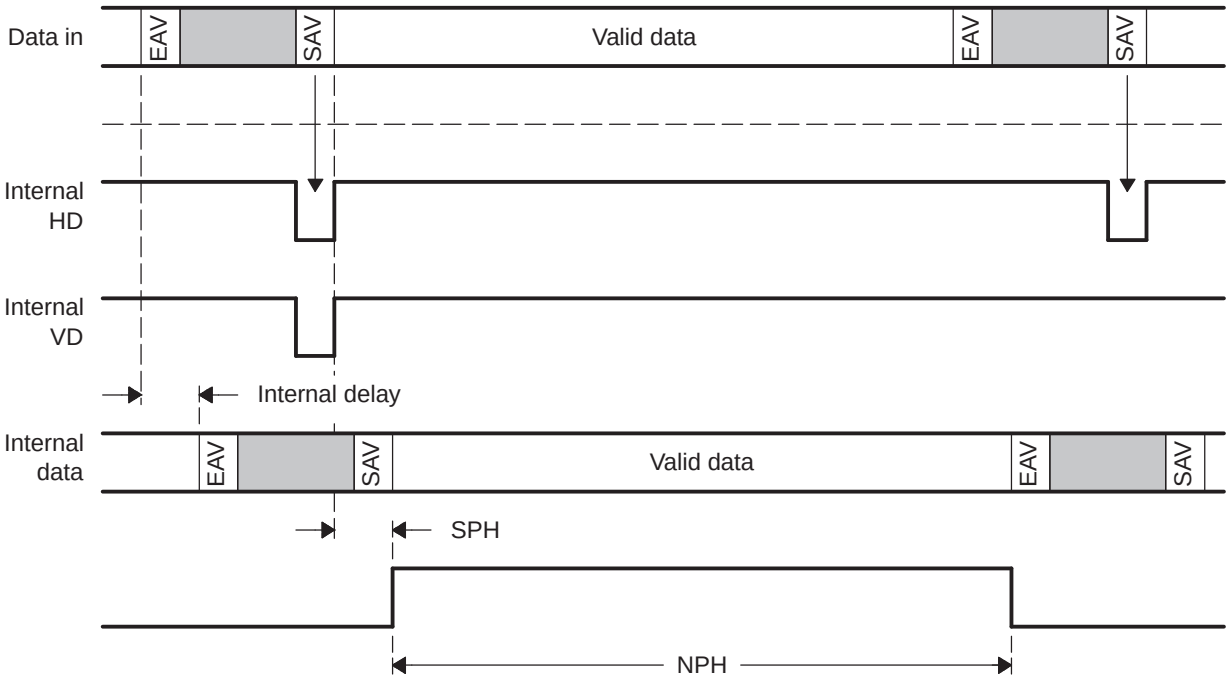
The signal interface is described in Table 10-3.

Table 10-3. ITU-R BT.656 Interface Signals

Name	I/O	Function
VDIN_D[9:0]	I	Image data – mode set by R656ON • Bit width is configurable to either eight or ten bits (BW656). • The polarity of the input image data is configurable (DATAPOL).
VDIN_PCLK	I	Pixel clock • The PCLK signal is the signal used to latch input image data. • The input image data can be captured on either the rising or on the falling edge of the PCLK signal which is configured by field PCLK_INV. (see for more details). • The maximum pixel clock rate is 75 MHz.

Two timing reference codes are transmitted as the synchronization signal. At the start and end of each video data block, two unique codes are sent, respectively. The start code is called the start of active video signal (SAV), and the end code is called the end of active video signal (EAV). The SAV and EAV codes proceed and follow valid data, as shown in Figure 10-4. The VPFE controller internally bases on SAV and EAV codes to generate the necessary synchronization signals, i.e., horizontal sync, vertical synx, and field ID.

Figure 10-4. BT.656 Signal Interface



Both timing reference signals, SAV and EAV, consist of a four-word sequence in the following format: FF 00 00 XY, where FF 00 00 are a set preamble and the fourth word defines the field identification, the state of vertical field blanking, the state of horizontal line blanking, and protection (error correction) codes. The bit format of the fourth word is shown in Table 10-4 and the definitions for bits, F, V, and H, are given in Table 10-5. F, V, and H are used in place of the usual horizontal sync, vertical sync, and blank timing control signals. Bits P3, P2, P1, and P0 are protection (error correction) bits for F, V, and H. The relationship the between F, V, and H and the protection (error correction) bits is given in Table 10-6. To enable error correction, set the ECCFVH bit in the REC656IF register. The VPFE controller automatically detects and applies error correction when the ECCFVH bit is set.

Table 10-4. Video Timing Reference Codes for SAV and EAV

Data Bit Number	First Word (FF)	Second Word (00)	Third Word (00)	Fourth Word (XY)
9 (MSB)	1	0	0	1
8	1	0	0	F
7	1	0	0	V
6	1	0	0	H
5	1	0	0	P3
4	1	0	0	P2
3	1	0	0	P1
2	1	0	0	P0
1	1	0	0	0
0	1	0	0	0

Table 10-5. F, V, H Signal Descriptions

Signal	Value	Command
F	0	Field 1
	1	Field 2
V	0	0
	1	Vertical blank
H	0	SAV
	1	EAV

Table 10-6. F, H, V Protection (error correction) Bits

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

NOTE: The VPFE controller outputs the XY code in the SAV and EAV into the external memory. In order to eliminate this, set the SPH register field to +1. Also set the NPH register field to accurately represent the number of active pixels.

10.3.1.3 Digital YCbCr Interface

The digital YCbCr interface supports either 8-bit or 16-bit devices. The signal interface is described in Table 10-7.

Unlike the BT.656 mode, discrete horizontal sync (VDIN_HD) and vertical sync (VDIN_VD) signals are required. An NTSC/PAL decoder is an example device that can be connected to the YCbCr interface.

You can use data lines YI[7:0] or CI [7:0] for input in 8-bit mode. Alternately, you can connect two separate devices; however, only one can be active at any given time. Setting the YCINSWP bit in the CCDCFG register determines which set of 8-bit inputs are active.

Use data lines YI [7:0] and CI[7:0] for input in 16-bit mode. Use the YCINSWP bit in the CCDCFG register to swap the Y and Cr/Cb data lines.

Table 10-7. CCD Interface Signals

Name	I/O	Function
VDIN_D [15:0] = YI [7:0] / CI [7:0]	I	Image data – mode set by INPMOD (not R656ON) • Bit width is only configurable to either 8 bits or 16 bits (INPMOD). • The polarity of the input image data is reversible (DATAPOL). • When the 16-bit interface is used, you can swap the Y and C inputs (YCINSWP). • When the 8-bit interface is used, you can connect either half of the bus (YCINSWP). • When the 8-bit interface is used, you can set the position of the Y data to either the even or odd pixel (Y8POS).
VDIN_VD	I	VSYNC - vertical sync signal
VDIN_HD	I	HSYNC - horizontal sync signal
VDIN_FIELD	I	Field identification signal (optional – FLDMODE) • This signal can be configured to be latched by the VD signal (FIDMD). • The polarity of the field identification signal is configurable (FLDPOL).
VDIN_PCLK	I	Pixel clock • The PCLK signal is the signal used to latch input image data. • The input image data can be captured on either the rising or on the falling edge of the PCLK signal which is configured by field PCLK_INV. • The maximum pixel clock rate is 75 MHz.

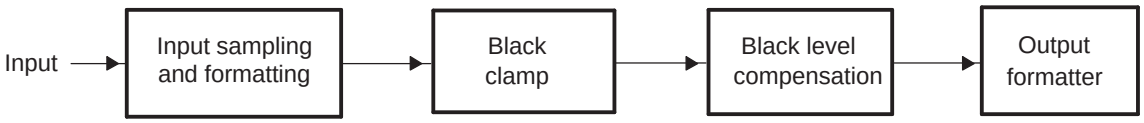
10.3.2 VPFE Data / Image Processing

This section describes the image/data processing of each module in the VPFE in more detail. Section 10.3.2.1 describes the raw data mode while Section 10.3.2.2 explains the YCbCr and BT656 modes.

10.3.2.1 Raw Data Mode

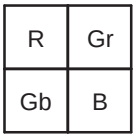
Raw data mode is enabled by setting SYN_MODE.INPMODE to 0 and setting REC656IF.REC656ON to 0. Figure 10-5 shows the corresponding data processing diagram.

Figure 10-5. Data Processing in Raw Data Mode



Typically, raw sensor data is one color per pixel in a color filter array (CFA). The color filter array applied is typically a Bayer pattern, as shown in Figure 10-6 for RGB color space. Alternatively, the special Foveon X3-family sensors capture R, G, and B lights at each pixel location. Both Bayer and Foveon sensors are supported by the VPFE controller.

Figure 10-6. Color Patterns



Bayer format with R/Gr and Gb/B in alternate lines
−Horizontal distance between same colors is 2.

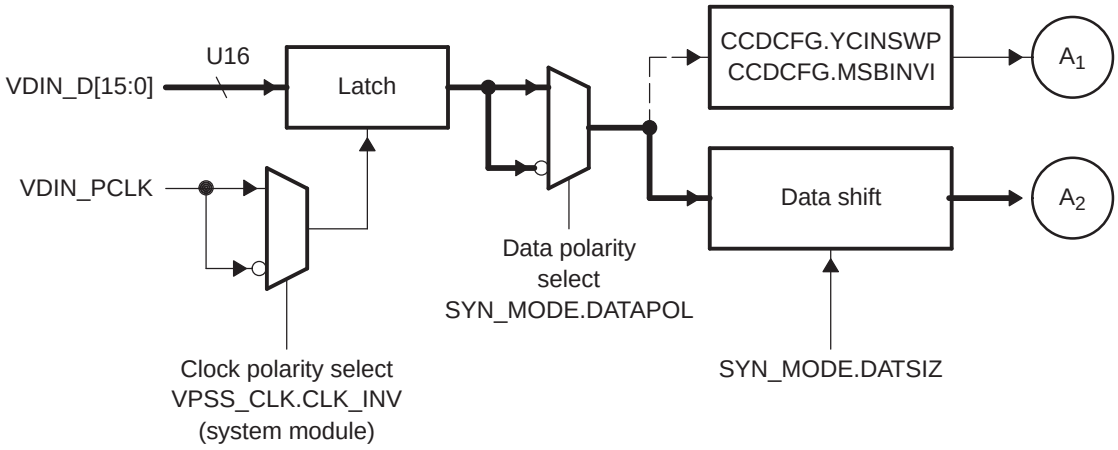
Foveon sensor with R, G, and B in same line
− Horizontal distance between same colors is 3.

10.3.2.1.1 Input Sampling and Formatting

The data path of raw data mode is shown in the thicker lines in Figure 10-7 (i.e., A2). Data path A1 is applicable to YUV input mode only.

- The pixel clock (VDIN_PCLK) latches the data.
- Pixel clock polarity can be either rising or falling edge and is set in VPSS clock control register (VPSS_CLK).PCK_INV).
- DATAPOL bit in the SYN_MODE register affects the data representation.
- Data is right-shifted to align the data in the least significant bits of the data bus and provide the maximum dynamic range for the remainder of the processing (DATSIZ bit in the SYN_MODE register). This also sets the maximum data size allowed in subsequent clipping/limiting operations and is the output data alignment when it is written to external memory.

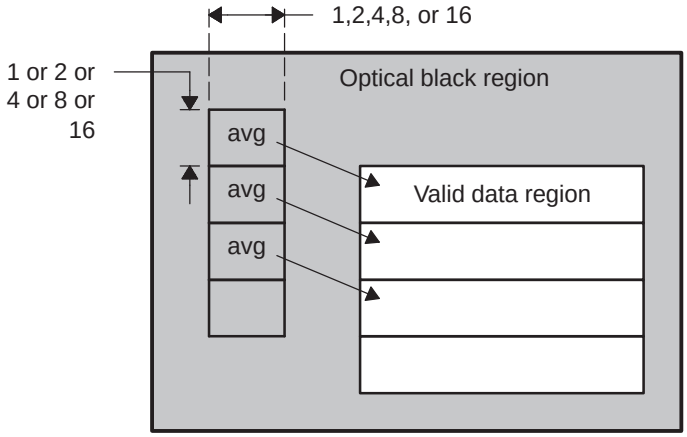
Figure 10-7. Input Formatter



10.3.2.1.2 Optical Black Clamping

Sensor manufacturers typically provide some optically masked pixels at the beginning/end of each line to allow you to determine the noise floor on any given frame of data. The optical black clamping function provides a means to average the optically black pixels and subtract that value from each input pixel as the first step in reducing the noise on the input pixels.

Figure 10-8. Optical Black Averaging & Application

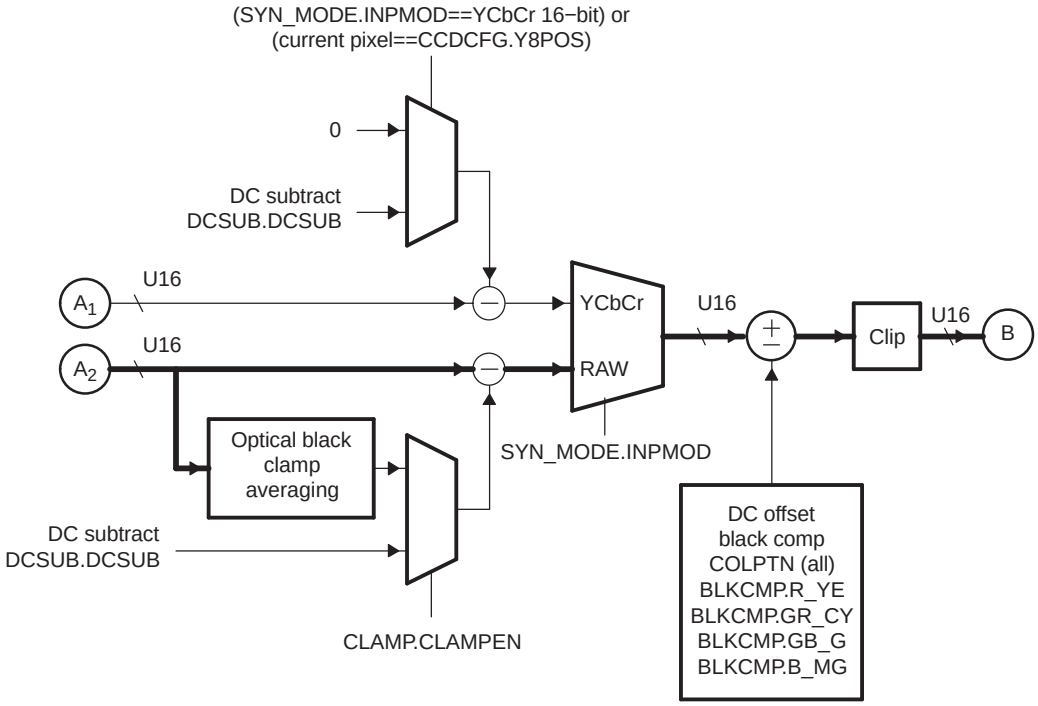


The averaging circuit takes an average of masked (black) pixel values from the image sensor, averaging pixels at the start (OBST bit in the CLAMP register) of each line (OBSLEN bit in the CLAMP register) and for the number of indicated lines (OBSLN bit in the CLAMP register), plus an optional gain adjustment (OBGAIN bit in the CLAMP register). The resultant value is subtracted from the image data at the succeeding line. You can control the position of the black pixels, the number of pixels (8 or 16) in each line that are averaged, and the number of lines (8 or 16) that are averaged.

Alternately, you can disable black clamp averaging (CLAMPEN bit in the CLAMP register) and select a constant black value for subtraction (DCSUB bit in the DCSUB register) instead of using the calculated average value.

The corresponding data path is shown in the thicker lines in Figure 10-9.

Figure 10-9. Black Clamping and Black Level Compensation



10.3.2.1.3 Black Level Compensation

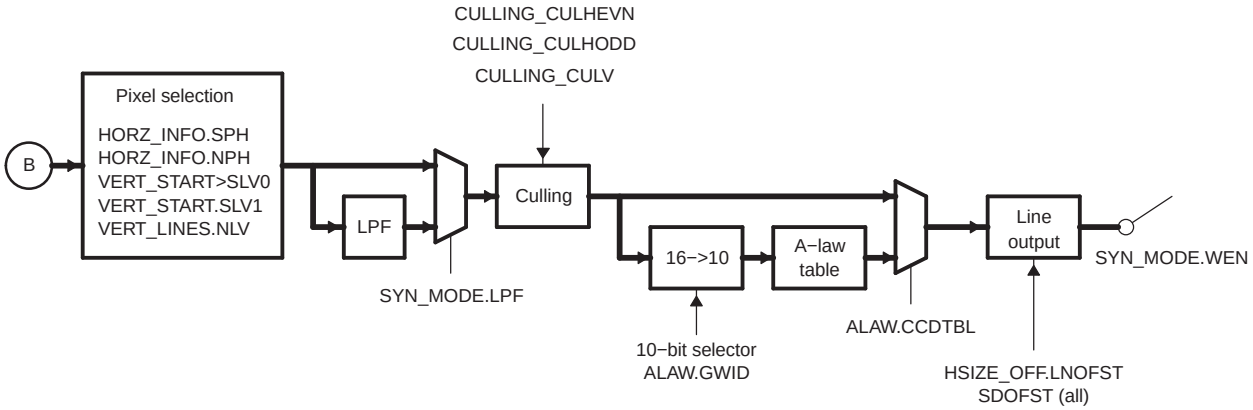
After the digital clamping is applied to the data, black level compensation is applied (Figure 10-9). In this operation, a fixed value is subtracted from the data depending on the color (i.e., R/Ye, Gr/Cy, Gb/G, and B/Mg). The offset (BLKCMP register, fields R_YE, GR_CY, GB_G, B_MG) that is applied to each data sample is selected according to the 0/1/2/3 phase and the color (0/1/2/3) specified for each phase (COLPTN). The color pattern definition is very flexible in order to accommodate many different capture devices, including normal Bayer CFA sampling, Foveon sensors, and VGA movie mode CCDs, whose VGA draft mode output does not follow the typical Bayer pattern.

10.3.2.1.4 Output Formatter

The final stage of VPFE processing is the output formatter, as shown in Figure 10-10. A framing selection is applied to limit the processing area by the settings in the HORZ_INFO, VERT_START and VERT_LINES registers.

NOTE: In addition to the framing applied at the beginning and end of the data formatter operation, you must apply a framing selection to limit the processing area. Please ensure that the settings are relative to that frame.

Figure 10-10. Output Formatter



10.3.2.1.4.1 Low Pass Filter

Use the LPF bit in the SYN_MODE register to apply an optional low-pass filter after the reframing. The low-pass filter consists of a simple 3-tap (¼ , ½ , ¼) filter. Two pixels on the left and two pixels on the right of each line are cropped if the filter is enabled.

10.3.2.1.4.2 Culling

Use the CULEVEN and CULODD bits in the CULLING register (8-bit repeating mask, one per field) to enable an optional culling operation, which culls (deletes) selected pixel data from a line. Use the CULV bit in the CULLING register to select lines from a frame.

Table 10-8 illustrates how the register values apply the decimation pattern to the data. The pixels in white will be saved to external memory and the shaded pixels are discarded.

In this case:

- CULLING = 0x59C40066, with
- CULHEVN = 0x59,
- CULHODD = 0xC4, and
- CULV = 0x66.

Table 10-8. Example for Decimation Pattern

		LSB								
	CULHEVN	0	0	1	0	0	0	1	1	
	CULHODD	1	0	0	1	1	0	1	0	0
0th line										1
1st line										1
2nd line										0
3rd line										0
4th line										1
5th line										1
6th line										0
7th line										CULV

10.3.2.1.4.3 A-Law Transformation

Use the CCDTBL bit in the ALAW register to apply an optional 10-to-8-bit A-Law transformation using a fixed A-Law table as the final processing stage. Using this causes the data width to reduce to 8 bits and allows packing to 8 bits/pixel when saving to external memory. Since the data resolution can be greater than 10 bits at this stage, you must select the 10 bits for input to the A-Law operation. Use the GWID bit in the ALAW register to select the 10 bits for input.

Figure 10-11. A-Law Table

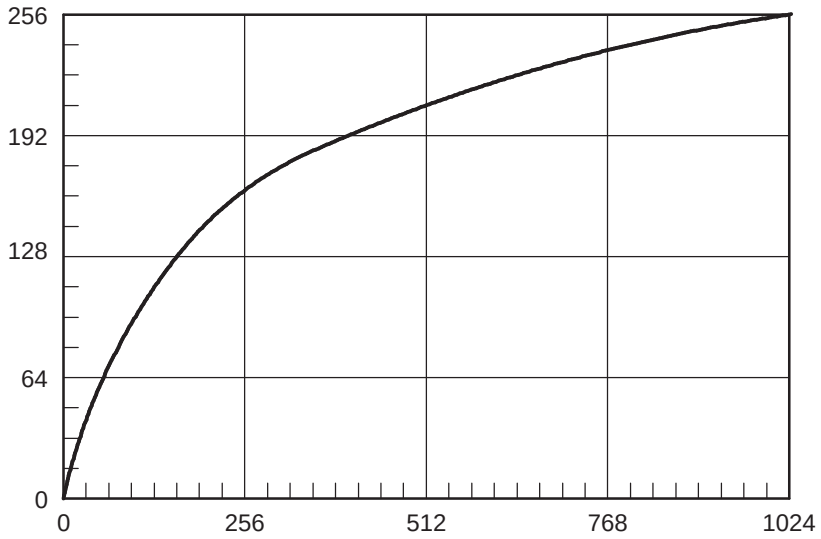


Table 10-9. A-Law Table – Part 1

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
0	0	64	64	128	112	192	140	256	161	320	176	384	189	448	200
1	1	65	65	129	113	193	141	257	161	321	176	385	189	449	200
2	2	66	66	130	113	194	141	258	161	322	177	386	189	450	200
3	3	67	67	131	114	195	142	259	161	323	177	387	189	4531	200
4	4	68	68	132	114	196	142	260	162	324	177	388	190	452	200
5	5	69	69	133	115	197	142	261	162	325	177	389	190	453	200
6	6	70	70	134	115	198	143	262	162	326	177	390	190	454	201
7	7	71	71	135	116	199	143	263	162	327	178	391	190	455	201
8	8	72	72	136	116	200	143	264	163	328	178	392	190	456	201
9	9	73	73	137	117	201	144	265	163	329	178	393	190	457	201
10	10	74	74	138	117	202	144	266	163	330	178	394	191	458	201
11	11	75	75	139	118	203	144	267	163	331	178	395	191	459	201
12	12	76	76	140	118	204	145	268	164	332	179	396	191	460	201
13	13	77	77	141	119	205	145	269	164	333	179	397	191	461	202
14	14	78	78	142	119	206	145	270	164	334	179	398	191	462	202
15	15	79	78	143	120	207	146	271	164	335	179	399	191	463	202
16	16	80	79	144	120	208	146	272	165	336	179	400	192	464	202
17	17	81	80	145	121	209	146	273	165	337	180	401	192	465	202
18	18	82	81	146	121	210	147	274	165	338	180	402	192	466	202
19	19	83	82	147	122	211	147	275	166	339	180	403	192	467	202
20	20	84	83	148	122	212	147	276	166	340	180	404	192	468	203
21	21	85	84	149	123	213	148	277	166	341	181	405	193	469	203
22	22	86	84	150	123	214	148	278	166	342	181	406	193	470	203
23	23	87	85	151	124	215	148	279	167	343	181	407	193	471	203

Table 10-9. A-Law Table – Part 1 (continued)

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
24	24	88	86	152	124	216	149	280	167	344	181	408	193	472	203
25	25	89	87	153	125	217	149	281	167	345	181	409	193	473	203
26	26	90	88	154	125	218	149	282	167	346	182	410	193	474	204
27	27	91	88	155	125	219	150	283	168	347	182	411	194	475	204
28	28	92	89	156	126	220	150	284	168	348	182	412	194	476	204
29	29	93	90	157	126	221	150	285	168	349	182	413	194	477	204
30	30	94	91	158	127	222	151	286	168	350	182	414	194	478	204
31	31	95	91	159	127	223	151	287	168	351	183	415	194	479	204
32	32	96	92	160	128	224	151	288	169	352	183	416	194	480	204
33	33	97	93	161	128	225	152	289	169	353	183	417	195	481	205
34	34	98	93	162	129	226	152	290	169	354	183	418	195	482	205
35	35	99	94	163	129	227	152	291	169	355	183	419	195	483	205
36	36	100	95	164	129	228	152	292	170	356	184	420	195	484	205
37	37	101	96	165	130	229	153	293	170	357	184	421	195	485	205
38	38	102	96	166	130	230	153	294	170	358	184	422	195	486	205
39	39	103	97	167	131	231	153	295	170	359	184	423	196	487	205
40	40	104	98	168	131	232	154	296	171	360	184	424	196	488	206
41	41	105	98	169	132	233	154	297	171	361	185	425	196	489	206
42	42	106	99	170	132	234	154	298	171	362	185	426	196	490	206
43	43	107	100	171	132	235	155	299	171	363	185	427	196	491	206
44	44	108	100	172	133	236	155	300	172	364	185	428	196	492	206
45	45	109	101	173	133	237	155	301	172	365	185	429	197	493	206
46	46	110	102	174	134	238	155	302	172	366	185	430	197	494	206
47	47	111	102	175	134	239	156	303	172	367	186	431	197	495	207
48	48	112	103	176	134	240	156	304	173	368	186	432	197	496	207
49	49	113	103	177	135	241	156	305	173	369	186	433	197	497	207
50	50	114	104	178.	135	242	157	306	173	370	186	434	197	498	207
51	51	115	105	179	136	243	157	307	173	371	186	435	198	499	207
52	52	116	105	180	136	244	157	308	173	372	187	436	198	500	207
53	53	117	106	181	136	245	157	309	174	373	187	437	198	501	207
54	54	118	106	182	137	246	158	310	174	374	187	438	198	502	208
55	55	119	107	183	137	247	158	311	174	375	187	439	198	503	208
56	56	120	108	184	137	248	158	312	174	376	187	440	198	504	208
57	57	121	108	185	138	249	159	313	175	377	188	441	198	505	208
58	58	122	109	186	138	250	159	314	175	378	188	442	199	506	208
59	59	123	109	187	139	251	159	315	175	379	188	443	199	507	208
60	60	124	110	188	139	252	159	316	175	380	188	444	199	508	208
61	61	125	110	189	139	253	160	317	175	381	188	445	199	509	208
62	62	126	111	190	140	254	160	318	176	382	188	446	199	510	209
63	63	127	112	191	140	255	160	319	176	383	189	447	199	511	209

Table 10-10. A-Law Table – Part 2

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
512	209	576	217	640	224	704	231	768	237	832	243	896	248	960	253
513	209	577	217	641	225	705	231	769	237	833	243	897	248	961	253
514	209	578	217	642	225	706	231	770	237	834	243	898	248	962	253
515	209	579	217	643	225	707	231	771	237	835	243	899	248	963	253
516	209	580	218	644	225	708	232	772	238	836	243	900	248	964	253

Table 10-10. A-Law Table – Part 2 (continued)

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
517	210	581	218	645	225	709	232	773	238	837	243	901	248	965	253
518	210	582	218	646	225	710	232	774	238	838	243	902	248	966	253
519	210	583	218	647	225	711	232	775	238	839	243	903	249	967	253
520	210	584	218	648	225	712	232	776	238	840	243	904	249	968	253
521	210	585	218	649	225	713	232	777	238	841	244	905	249	969	253
522	210	586	218	650	226	714	232	778	238	842	244	906	249	970	254
523	210	587	218	651	226	715	232	779	238	843	244	907	249	971	254
524	211	588	219	652	226	716	232	780	238	844	244	908	249	972	254
525	211	589	219	653	226	717	232	781	238	845	244	909	249	973	254
526	211	590	219	654	226	718	233	782	238	846	244	910	249	974	254
527	211	591	219	655	226	719	233	783	239	847	244	911	249	975	254
528	211	592	219	656	226	720	233	784	239	848	244	912	249	976	254
529	211	593	219	657	226	721	233	785	239	849	244	913	249	977	254
530	211	594	219	658	226	722	233	786	239	850	244	914	249	97.8	254
531	211	595	219	659	227	723	233	787	239	851	244	915	249	979	254
532	212	596	220	660	227	724	233	788	239	852	244	916	250	980	254
533	212	597	220	661	227	725.	233	789	239	853	245	917	250	981	254
534	212	598	220	662	227	726	233	790	239	854	245	918	250	982	254
535	212	599	220	663	227	727	233	791	239	855	245	919	250	983	254
536	212	600	220	664	227	728	233	792	239	856	245	920	250	984	255
537	212	601	220	665	227	729	234	793	239	857	245	921	250	985	255
538	212	602	220	666	227	730	234	794	240	858	245	922	250	986	255
539	212	603	220	667	227	731	234	795	240	859	245	923	250	987	255
540	213	604	220	668	227	732	234	796	240	860	245	924	250	988	255
541	213	605	221	669	228	733	234	797	240	861	245	925	250	989	255
542	213	606	221	670	228	734	234	798	240	862	245	926	250	990	255
543	213	607	221	671	228	735	234	799	240	863	245	927	250	991	255
544	213	608	221	672	228	736	234	800	240	864	245	928	250	992	255
545	213	609	221	673	228	737	234	801	240	865	246	929.	250	993	255
546	213	610	221	674	228	738	234	802	240	866	246	930	251	994	255
547	214	611	221	675	228	739	235	803	240	867	246	931	251	995	255
548	214	612	221	676	228	740	235	804	240	868	246	932	251	996	255
549	214	613	221	677	228	741	235	805	240	869	246	933	251	997	255
550	214	614	222	678	229	742	235	806	241	870	246	934	251	998	255
551	214	615	222	679	229	743	235	807	241	871	246	935	251	999	255
552	214	616	222	680	229	744	235	808	241	872	246	936	251	1000	255
553	214	617	222	681	229	745	235	809	241	873	246	937	251	1001	255
554	214	618	222	682	229	746	235	810	241	874	246	938	251	1002	255
555	215	619	222	683	229	747	235	811	241	875	246	939	251	1003	255
556	215	620	222	684	229	748	235	812	241	876	246	940	251	1004	255
557	215	621	222	685	229	749	235	813	241	877	246	941	251	1005	255
558	215	622	222	686	229	750	236	814	241	878	247	942	251	1006	255
559	215	623	223	687	229	751	236	815	241	879	247	943	252	1007	255
560	215	624	223	688	230	752	236	816	241	880	247	944	252	1008	255

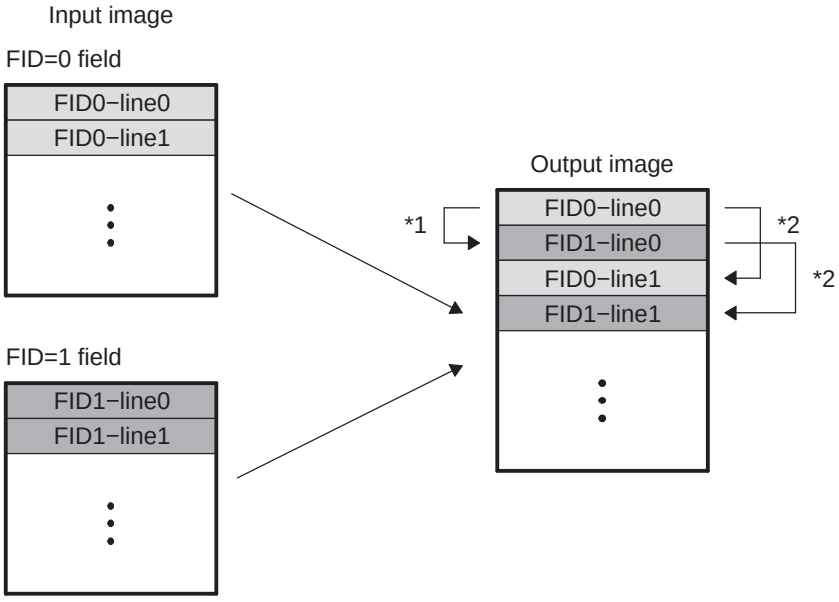
Table 10-10. A-Law Table – Part 2 (continued)

Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law	Input	A-Law
561	215	625	223	689	230	753	236	817	242	881	247	945	252	1009	255
562	215	626	223	690	230	754	236	818	242	882	247	946	252	1010	255
563	216	627	223	691	230	755	236	819	242	883	247	947	252	1011	255
564	216	628	223	692	230	756	236	820	242	884	247	948	252	1012	255
565	216	629	223	693	230	757	236	821	242	885	247	949	252	1013	255
566	216	630	223	694	230	758	236	822	242	886	247	950	252	1014	255
567	216	631	223	695	230	759	236	823	242	887	247	951	252	1015	255
568	216	632	224	696	230	760	236	824	242	888	247	952	252	1016	255
569	216	633	224	697	230	761	237	825	242	889	247	953	252	1017	255
570	216	634	224	698	231	762	237	826	242	890	247	954	252	1018	255
571	217	635	224	699	231	763	237	827	242	891	248	955	252	1019	255
572	217	636	224	700	231	764	237	828	242	892	248	956	252	1020	255
573	217	637	224	701	231	765	237	829	243	893	248	957	253	1021	255
574	217	638	224	702	231	766	237	830	243	894	248	958	253	1022	255
575	217	639	224	703	231	767	237	831	243	895	248	959	253	1023	255

10.3.2.1.4.4 Line Output Control

The final stage of the raw data mode is the line output control, which controls how the input sensor lines are written to external memory. The value of the ADR bit in the SDR_ADDR register defines the starting address where the frame should be written in external memory. The value of the LNOFST bit in the HSIZE_OFF register defines the distance between the beginning of output lines in bytes. Both the starting address and line offset must be aligned to 32-byte boundaries (i.e., either 16 or 32 pixels, depending on the PACK8 bit in the SYN_MODE register). Use the SDOFST register to define additional offsets, depending on the field ID and the even/odd line numbers. Defining additional offsets provides a means to de-interlace an interlaced, two-field input and also inverts an input image vertically. See Figure 10-12 and Figure 10-13 for example usage.

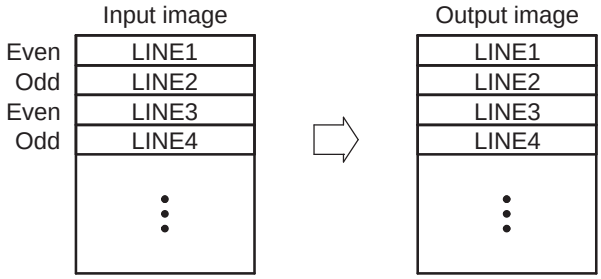
Figure 10-12. Image De-interfacing



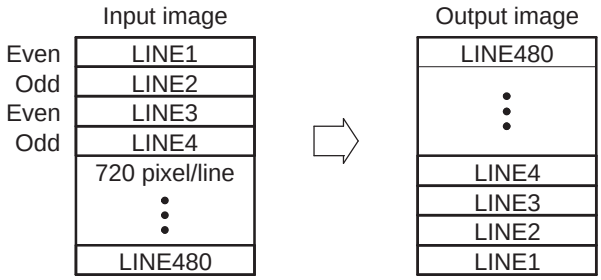
*1 − SDOFST[13:12]=00b, +1 line for FID=1 lines

*2 − SDOFST[11:9]=SDOFST[8:6]=SDOFST[5:3]=SDOFST[2:0]=001b, +2 lines

Figure 10-13. Non-inversed vs Inversed Format



SDOFST[14] = 0 ;Non inverse
SDOFST[13:12] = 00b ;+1 line when field ID is 1
SDOFST[11:9] = 001b ;+2 line, even line and field ID = 0
SDOFST[8:6] = 001b ;+2 line, odd line and field ID = 0
SDOFST[5:3] = 001b ;+2 line, even line and field ID = 1
SDOFST[2:0] = 001b ;+2 line, odd line and field ID = 1



SDOFST[14] = 1 ;Inverse
SDOFST[13:12] = 000b ;−1 line when field ID is 1
SDOFST[11:9] = 101b ;−2 line, even line and field ID = 0
SDOFST[8:6] = 101b ;−2 line, odd line and field ID = 0
SDOFST[5:3] = 101b ;−2 line, even line and field ID = 1
SDOFST[2:0] = 101b ;−2 line, odd line and field ID = 1

10.3.2.1.4.5 Output Format

The pixel data format in external memory is shown in Table 10-12.

- If pixel data format is 8-bit, every 16-bit word in external memory stores two pixel data.
- If pixel data format is greater than 8-bit, every 16-bit word in external memory stores one pixel data, and the unused bits are MSB which are filled with 0.

Table 10-11. Storage Format in external memory for Raw Data Mode

	Upper Word		Lower Word	
	MSB (31)	LSB (16)	MSB (15)	LSB (0)
16-bit	Pixel1		Pixel0	
15-bit	0	Pixel1	0	Pixel0
14-bit	0	Pixel1	0	Pixel0
13-bit	0	Pixel1	0	Pixel0
12-bit	0	Pixel1	0	Pixel0
11-bit	0	Pixel1	0	Pixel0
10-bit	0	Pixel1	0	Pixel0
9-bit	0	Pixel1	0	Pixel0
8-bit	0	Pixel1	0	Pixel0
8-bit pack	Pixel3	Pixel2	Pixel1	Pixel0

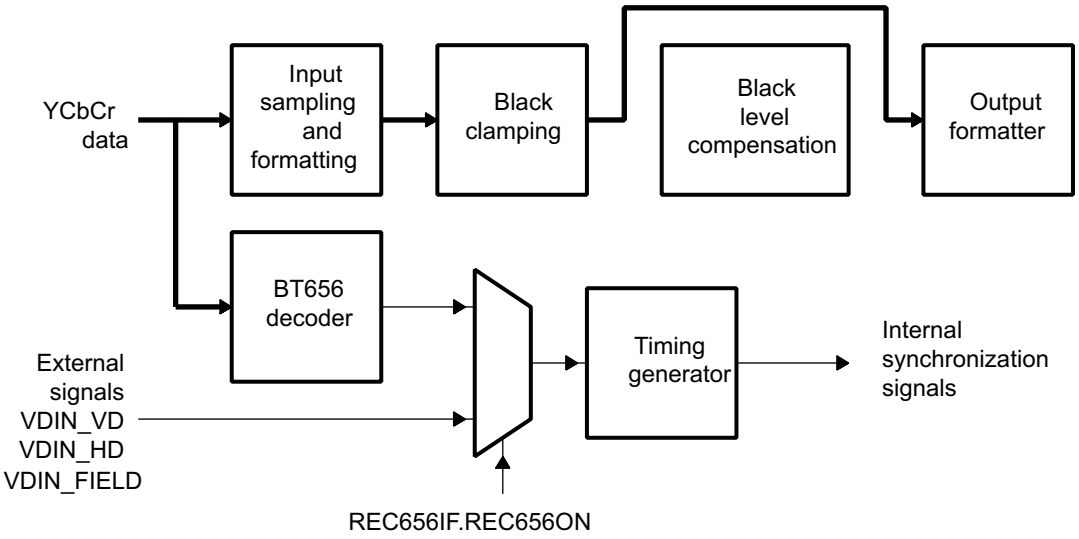
10.3.2.2 YCbCr and BT656 Modes

YCbCr mode and BT656 mode are similar in operation, as shown in Figure 10-14. The additional logic used in BT656 mode is the CCIR656 decoder, which extracts synchronization information from input YCbCr data and regenerates the corresponding timing for internal operation.

- YCbCr mode is enabled by setting field INPMODE in register SYN_MODE to 1 or 2 and clearing field REC656ON in register REC656IF.
- BT656 mode is enabled by setting REC656ON in register REC656IF to 1.

YCbCr mode typically has 8 bits per luma/choma sample while BT656 mode has 8 bits or 10 bits per luma/chroma sample.

Figure 10-14. Data Processing in YUV/BT656 Modes

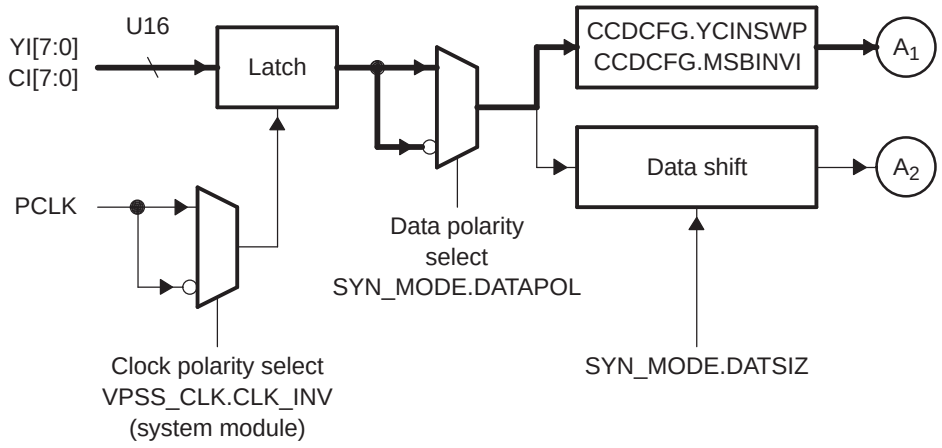


10.3.2.2.1 Input Sampling and Formatting

The data path of YCbCr/BT656 modes is shown in thicker lines in Figure 10-15 (i.e., A1). Data path A2 is applicable to raw data mode only.

- The pixel clock (VDIN_PCLK) latches data.
- Pixel clock polarity can be either rising or falling edge and is set in VPSS clock control register (VPSS_CLK).PCLK_INV).
- DATAPOL bit in the SYN_MODE register affects the data representation.
- Bit YCINSWP in register CCDCFG can be used to swap the upper and lower portions of the 16-bit YCbCr data bus.
  - In 16-bit YCbCr mode, this swap bit determines which part of the bus luma and chroma samples occupy respectively.
  - In 8-bit mode, this swap bit determines which part of the bus is the effective 8-bit input. In other words, two external 8-bit YCbCr capture devices can be tied to VPFE module directly.
- Bit MSBINVI in register CCDCFG can be used to invert the MSB of the chroma signal.

Figure 10-15. CCD Controller

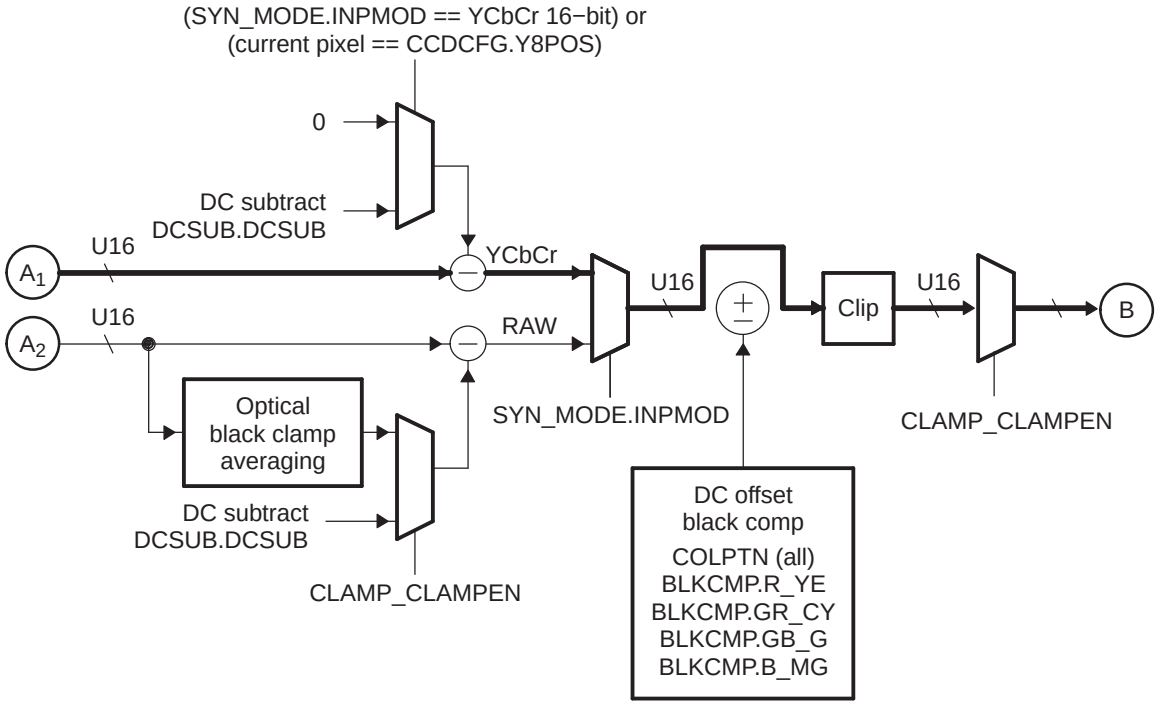


10.3.2.2.2 Black Clamping

The second step in BT.656/YCbCr processing is black clamping

Use the DCSUB bit in the DCSUB register to subtract a fixed value from the luma sample for YUV data. Set the subtraction value to zero to disable the operation. See Figure 10-16 for more details. You may notice that black compensation (BLKCMP) is not used in YCbCr and BT656 modes.

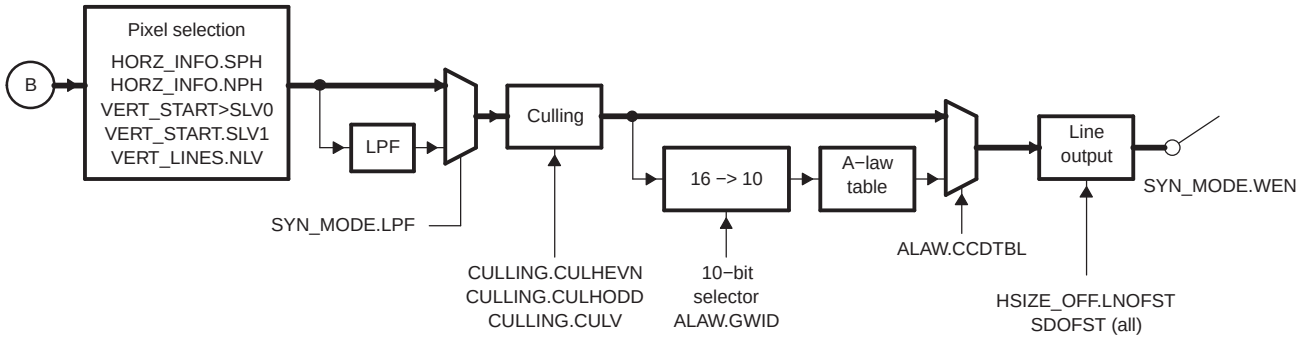
Figure 10-16. Black Clamping and Block Level Compensation



10.3.2.2.3 Output Formatter

The output formatter is the final stage of VPFE processing (as shown as thicker lines in Figure 10-17). Apply a framing selection to limit the processing area by the settings in the HORZ_INFO, VERT_START and VERT_LINES registers.

Figure 10-17. Output Formatter



- Use the LFP bit in the SYN_MODE register to disable the LPF by setting the LFP bit equal to 0.
- Culling can be used in YCbCr and BT656 modes; however, care must be taken to preserve the 422 output format.
- Do not use the A-Law transformation in YCbCr or BT.656 mode (the CCDTBL bit in the ALAW register = 0).

The pixel data in external memory is shown in Table 10-12.

Table 10-12. Storage Format in external memory for BT.656/YCbCr Modes

external memory Address	Upper word		Lower word	
	MSB (31)	LSB (16)	MSB (15)	LSB (0)
N	Y1	Cr0	Y0	Cb0
N + 1	Y3	Cr2	Y2	Cb2
N + 2	Y5	Cr4	Y4	Cb4

10.4 Programming Model

The following programming procedures should be followed to enable the VPFE controller to receive video or image capture data.

1. Power-up the VPFE via the power sleep controller (PSC) module (see *TDRx40x/41x Automotive Digital Radio and Infotainment Applications Processor ARM Subsystem Reference Guide* (SPRUF07)
2. Disable the VPFE controller
3. Configure the VPFE registers to match the system requirements, including interrupt setup
4. Enable the VPFE controller

Once the VPFE controller is enabled, it starts to process input data continuously. No user intervention is necessary to re-start the process. As a result, the VPFE interrupt must be set up correctly in order to operate properly. Users can terminate its operation by disabling the VPFE controller.

The remaining sections describe the programming procedure in more detail.

10.4.1 Enabling and Disabling the VPFE Controller

The following steps describe how to enable and disable the VPFE controller:

- Clearing the ENABLE bit in the VPFE_PCR register disables the VPFE controller.
- Setting the ENABLE bit in the VPFE_PCR register enables the VPFE controller.
  - The VPFE controller should be enabled prior to data transmission from the external device to avoid data loss.

10.4.2 Configuring VPFE Registers

10.4.2.1 General Register Setup

Table 10-13 lists the minimum register fields that must be configured.

Table 10-13. Basic Configuration of VPFE Registers

Function	Register	Fields
External signal configuration (This includes signals VDIN_VD, VDIN_HD, VDIN_FIELD, and VDIN_WEN.)	SYN_MODE	VDHDEN VDPOL HDPOL FLDMODE FLDPOL EXWEN DATAPOL
	CCDCFG	VDLC
Input data mode	REC656	R656ON
	SYN_MODE	INPMOD
Color pattern	COLPTN	All
Black compensation	BLKCMP	All
Data path configuration	SYN_MODE	WEN

Table 10-14 describes additional configuration requirements when certain conditions are met.

Table 10-14. Conditional Configuration of VPFE Registers

Category	If...	Then program...
Interlace mode	Field FLDMODE in register SYN_MODE is 1	Field FIDMD in register CCDCFG
External WEN	Field EXWEN in register SYN_MODE is 1	Field WENLOG bit in register CCDCFG
REC656 input	Field R656ON in register REC656 is 1	Field ECCFVH in register REC656 Field BW656 in register CCDCFG
RAW input	Field INPMOD in register SYN_MODE is 0 and Field R656ON in register REC656 is 1	Field DATSIZ in register SYN_MODE Field CLAMPEN in register CLAMP
16-bit YCC input	Field INPMOD in register SYN_MODE is 1 and Field R656ON in register REC656 is 1	Field YCINSWP in register CCDCFG Field MSBINVI in register CCDCFG Register DCSUB
8-bit YCC input	Field INPMOD in register SYN_MODE is 2 and Field R656ON in register REC656 is 1	Field Y8POS in register CCDCFG
Optical black clamp enabled	Field CLAMPEN in register CLAMP is 1 and Field INPMOD in register SYN_MODE is 0	Field OBGAIN in register CLAMP Field OBST in register CLAMP Field OBSLN in register CLAMP Field OBSLEN in register CLAMP
Optical black clamp disabled	Field CLAMPEN in register CLAMP is 0 and Field INPMOD in register SYN_MODE is 0	Register DCSUB
Write to external memory	Field WEN in register SYN_MODE is 1	Field LPF in register SYN_MODE Field PACK8 in register SYN_MODE Field CCDTBL in register ALAW Field BSWD in register CCDCFG Register HORZ_INFO Register VERT_START Register VERT_LINES Register CULLING Register SDR_ADDR Register HSIZE_OFF Register SDOFST
A-law	Field CCDTBL in register ALAW is 1	Field GWID in register ALAW
Interrupt	Interrupts CCDC_VD0_INT and CCDC_VD1_INT are to be used	Register VDINT

10.4.2.2 Interrupts

The VPFE controller can generate three interrupts: CCDC_VD0_INT, CCDC_VD1_INT, and CCDC_VD2_INT.

NOTE: Enable the VDHDEN field in register SYN_MODE to receive any of these VPFE controller interrupts.

The CCDC_VD0_INT and CCDC_VD1_INT interrupts occur relative to the VD pulse, as shown in Figure 10-18 and Figure 10-19. Please note that field VDPOL in register SYN_MODE changes the trigger timing.

- If field VDPOL is 1, the VDINT0 and VDINT1 counters begin counting VDIN_HD pulses from the falling edge of signal VDIN_VD.
- If field VDPOL is 0, the VDINT0 and VDINT1 counters begin counting VDIN_HD pulses from the rising edge of signal VDIN_VD.

- Interrupts CCDC_VD0_INT and CCDC_VD1_INT occur after receiving the number of horizontal lines (VDIN_HD pulse signals) set in the VDINT0 bit in the VDINT register and the VDINT1 bit in the VDINT register, respectively.

Figure 10-18. VDPOL is 0

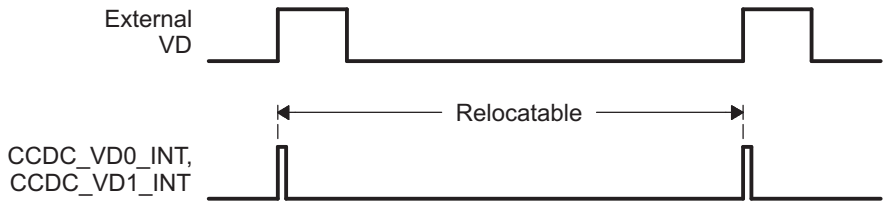
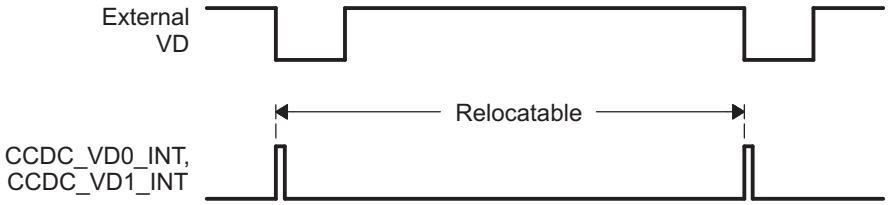
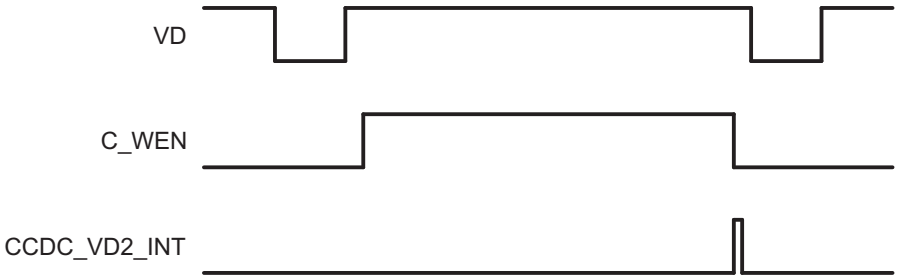


Figure 10-19. VDPOL is 1



The CCDC_VD2_INT interrupt always occurs at the falling edge of the VDIN_WEN signal (via an external pin), as shown in Figure 10-20. There are no registers in the VPFE module to configure this interrupt.

Figure 10-20. CCDC_VD2_INT Interrupt



10.4.2.3 Status

The BUSY status bit in the VPFE_PCR register is set when the start of frame occurs (if the ENABLE bit in the VPFE_PCR register is 1 at that time). It automatically resets to 0 at the end of a frame.

10.4.2.4 VDIN_VD latched Registers

The VDLC field in register CCDCFG affects the access of the following registers and register fields.

Registers
VPFE_PCR
HORZ_INFO
VERT_START
VERT_LINES
CULLING
HSIZE_OFF
SDOFST
SDR_ADDR
Register Fields
WEN in SYN_MODE
LPF in SYN_MODE

Registers
CLAMPEN in CLAMP
YCINSWP in CCDCFG

NOTE:

1. If the VDLC field is 0, changes to the above registers/register fields will take effect immediately.
2. If the VDLC field is 1, changes to the above registers/register fields will take effect at the start of a new frame (i.e., the changes are latched by the VDIN_VD signal). Reads from these registers/register fields will return the most recent write value (even though these values may not take effect).

Be careful to avoid undesired effects when using the first approach.

10.4.2.5 Inter-frame Operations

As described in the previous section, the VPFE_PCR and SDR_ADDR registers will be latched by the VDIN_VD signal if the VDLC field in the CCDCFG register is 0. This important feature provides a reliable way for programmers to enable/disable the VPFE module and/or modify the memory pointers in-between frames. For example, the VPFE interrupt service routine (ISR) can program the SDR_ADDR register to a new value before the end of the current frame. By doing so, the current frame is received without any interruption and the new external memory address (SDR_ADDR register) will be used for receiving the next frame.

10.4.3 VPFE Limitations

The major limitations of the VPFE module are summarized as follows:

- The VDIN_PCLK (pixel clock) signal cannot be higher than 75 MHz.
- The SDR_ADDR and HSIZE_OFF registers should be programmed to values with 5 LBS bits as 0; namely, the memory space they point to should be on a 32-byte boundary.
- The COLPTN register should be set to 0 in YCbCr and BT.656 modes.
- The BLKCMP register should be set to 0 in YCbCr and BT.656 modes.
- Low-pass filter (LPF field in SYN_MODE register) should be disabled in YCbCr and BT.656 modes.
- A-LAW should be disabled in YCbCr and BT.656 modes.

10.5 Video Processing Front End (VPFE) Registers

Table 10-15 lists the memory-mapped registers for the VPFE controller.

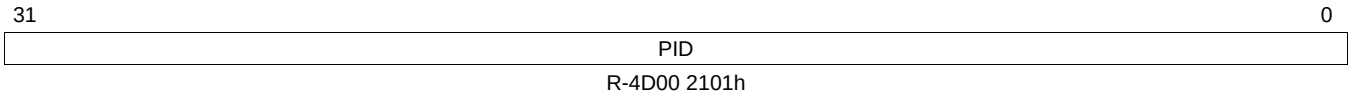
Table 10-15. CCD Controller (CCDC) Register Map

Offset	Acronym	Register Description	Section
00h	PID	Peripheral Revision and Class Information	Section 10.5.1
04h	VPFE_PCR	VPFE_Peripheral Control Register	Section 10.5.2
08h	SYN_MODE	SYNC and Mode Set Register	Section 10.5.3
Ch	Reserved	Reserved	
10h	Reserved	Reserved	
14h	HORZ_INFO	Horizontal Pixel Information	Section 10.5.4
18h	VERT_START	Vertical Line - Settings for the Starting Pixel	Section 10.5.5
1Ch	VERT_LINES	Number of Vertical Lines	Section 10.5.6
20h	CULLING	Culling Information in Horizontal and Vertical Directions	Section 10.5.7
24h	HSIZE_OFF	Horizontal Size	Section 10.5.8
28h	SDOFST	External memory/Line Offset	Section 10.5.9
2Ch	SDR_ADDR	External memory Address	Section 10.5.10
30h	CLAMP	Optical Black clamping settings	Section 10.5.11
34h	DCSUB	DC Clamp	Section 10.5.12
38h	COLPTN	CCD Color Pattern	Section 10.5.13
3Ch	BLKCMP	Black Compensation	Section 10.5.14
40h-44h	Reserved	Reserved	
48h	VDINT	VPFE Interrupt Control	Section 10.5.15
4Ch	ALAW	ALAW Configuration	Section 10.5.16
50h	REC656IF	REC656 Configuration	Section 10.5.17
54h	CCDCCFG	CCD Configuration	Section 10.5.18
58h - 94h	Reserved	Reserved	
98h	DMA_CNTL	DMA Status and Control	Section 10.5.19

10.5.1 Peripheral Revision and Class Information Register (PID)

The peripheral revision and class information register (PID) is shown in Figure 10-21 and described in Table 10-16 .

Figure 10-21. Peripheral Revision and Class Information Register (PID)



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

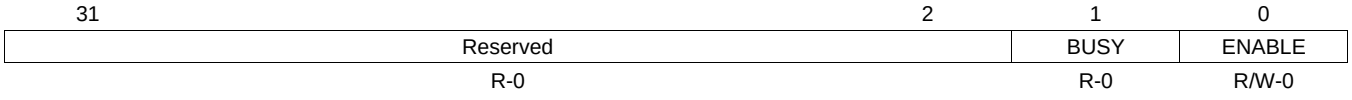
Table 10-16. Peripheral Revision and Class Information Register (PID) Field Descriptions

Bit	Field	Value	Description
31-0	PID	4D00 2101h	Peripheral identification that uniquely identifies the specific revision of the VPFE

10.5.2 VPFE_Peripheral Control Register (VPFE_PCR)

The VPFE_peripheral control register (VPFE_PCR) is shown in Figure 10-22 and described in Table 10-17.

Figure 10-22. VPFE_Peripheral Control Register (VPFE_PCR)



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

Table 10-17. VPFE_Peripheral Control Register (VPFE_PCR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Any writes to these bit(s) must always have a value of 0.
1	BUSY		VPFE busy bit
		0	Not busy
		1	Busy
0	ENABLE		VPFE enable
		0	Disable
		1	Enable

10.5.3 SYN_MODE Format and Description Register (SYN_MODE)

The sync and mode set register (SYN_MODE) is shown in Figure 10-23 and described in Table 10-18.

Figure 10-23. Sync and Mode Set Register (SYN_MODE)

31																19	18	17	16
Reserved																		WEN	VDHDEN
R-0																		R/W-0	R/W-0

15	14	13	12	11	10	8	7	6	5	4	3	2	1	0
FLDSTAT	LPF	INPMOD		PACK8	DATSIZ		FLDMODE	DATAPOL	EXWEN	FLDPOL	HDPOL	VDPOL	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 10-18. Sync and Mode Set Register (SYN_MODE) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Any writes to these bit(s) must always have a value of 0.
17	WEN		Data write enable. Controls whether or not input raw data is written to external memory. This bit is latched by VD.
		0	Disable
		1	Enable
16	VDHDEN		VD/HD enable. Activates internal timing generator to synchronize with external VD/HD signals. This bit should be set to 1 when HD and VD signals are used at any time.
		0	Disable
		1	Enable
15	FLDSTAT		Field status. Indicates the status of the current field when in interlaced mode.
		0	Odd field
		1	Even field
14	LPF		3-tap low-pass (anti-aliasing) filter. This bit is latched by VD.
		0	Off
		1	On
13-12	INPMOD	0-3h	Setting data input mode
		00	Raw data
		01	YCbCr 16-bit
		10	YCbCr 8-bit
		11	Reserved
11	PACK8		Pack to 8-bit/pixel (into external memory)
		0	Normal (16 bits/pixel)
		1	Pack to 8 bits/pixel
10-8	DATSIZ	0-7h	CCD data width is only valid when INPMOD is set to 0.
		000	16 bits
		001	15 bits
		010	14 bits
		011	13 bits
		100	12 bits
		101	11 bits
		110	10 bits
		111	8 bits
7	FLDMODE		Sensor field mode
		0	Non-interlaced (progressive)
		1	Interlaced

Table 10-18. Sync and Mode Set Register (SYN_MODE) Field Descriptions (continued)

Bit	Field	Value	Description
6	DATAPOL		Input data polarity
		0	Normal (no change)
		1	One's complement
5	EXWEN		External WEN selection. When set to 1 and when VDHDEN is set to 1, the WEN signal is used as the external memory write enable (to external memory). The data is stored to memory only when the external sync (HD and VD) signals are active.
		0	Do not use external WEN (write enable)
		1	Use external WEN (write enable)
4	FLDPOL		Field indicator polarity
		0	Positive
		1	Negative
3	HDPOL		HD sync polarity
		0	Positive
		1	Negative
2	VDPOL		VD sync polarity
		0	Positive
		1	Negative
1-0	Reserved	0	Any writes to these bit(s) must always have a value of 0.

10.5.4 Horizontal Pixel Information Register (HORZ_INFO)

The horizontal pixel information register (HORZ_INFO) is shown in Figure 10-24 and described in Table 10-19.

Figure 10-24. Horizontal Pixel Information Register (HORZ_INFO)

31	30	16
Reserved	SPH	
R-0	R/W-0	

15	14	0
Reserved	NPH	
R-0	R/W-0	

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

Table 10-19. Horizontal Pixel Information Register (HORZ_INFO) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-16	SPH	0-7FFFh	Start pixel, horizontal. The SPH sets the pixel clock position at which data output to external memory begins, measured from the start of HD. This bit field is latched by VD.
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-0	NPH	0-7FFFh	Number of pixels, horizontal. NPH sets the number of horizontal pixels that is output to external memory = (NPH + 1) and 0xFFF0 (i.e., the number of horizontal output pixels truncates to multiples of 16). This bit field is latched by VD.

10.5.5 Vertical Line - Settings for the Starting Pixel Register (VERT_START)

The vertical line - settings for the starting pixel register (VERT_START) is shown in Figure 10-25 and described in Table 10-20.

Figure 10-25. Vertical Line - Settings for the Starting Pixel Register (VERT_START)

31	30	16
Reserved	SLV0	
R-0	R/W-0	

15	14	0
Reserved	SLV1	
R-0	R/W-0	

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

Table 10-20. Vertical Line - Settings for the Starting Pixel Register (VERT_START) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-16	SLV0	0-7FFFh	Start line, vertical (field 0). SLV0 sets line at which data output to external memory will begin, measured from the start of VD. This bit field is latched by VD
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-0	SLV1	0-7FFFh	Start line, vertical (field 1). SLV1 sets line at which data output to external memory will begin, measured from the start of VD. For a progressive sensor this field is ignored. This bit field is latched by VD

10.5.6 Number of Vertical Lines Register (VERT_LINES)

The number of vertical lines register (VERT_LINES) is shown in Figure 10-26 and described in Table 10-21.

Figure 10-26. Number of Vertical Lines Register (VERT_LINES)

31	16
Reserved	
R-0	

15	14	0
Reserved	NLV	
R-0	R/W-0	

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

Table 10-21. Number of Vertical Lines Register (VERT_LINES) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-0	NLV	0-7FFFh	Number of lines, vertical. NLV sets the number of vertical lines that will be output to external memory. The number of lines output to external memory = (NLV + 1). This bit field is latched by VD.

10.5.7 Culling Information in Horizontal and Vertical Directions Register (CULLING)

The culling information in horizontal and vertical directions register (CULLING) is shown in Figure 10-27 and described in Table 10-22.

Figure 10-27. Culling Information in Horizontal and Vertical Directions Register (CULLING)

31	24	23	16
CULHEVN		CULHODD	
R/W-255		R/W-255	

15	8	7	0
Reserved		CULV	
R-0		R/W-255	

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

Table 10-22. Culling Information in Horizontal and Vertical Directions Register (CULLING) Field Descriptions

Bit	Field	Value	Description
31-24	CULHEVN	0-FFh	Horizontal Culling Pattern for Even Line, 8-bit mask. LSB is first pixel, MSB is 8th pixel, then pattern repeats. This bit field is latched by VD.
		0	CULLING (deletion)
		1	Retain (to be saved to external memory).
23-16	CULHODD	0-FFh	Horizontal Culling Pattern for Odd Line, 8-bit mask. LSB is first pixel, MSB is 8th pixel, then pattern repeats. This bit field is latched by VD.
		0	CULLING (deletion)
		1	Retain (to be saved to external memory).
15-8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7-0	CULV	0-FFh	Vertical Culling Pattern, 8-bit mask. LSB is first line, MSB is 8th line, then pattern repeats. This bit field is latched by VD.
		0	CULLING (deletion)
		1	Retain (to be saved to external memory).

10.5.8 Horizontal Size (HSIZE_OFF)

The horizontal size register (HSIZE_OFF) is shown in Figure 10-28 and described in Table 10-23.

Figure 10-28. Horizontal Size Register (HSIZE_OFF)

31	16
Reserved	
R-0	

15	0
LNOFST	
R/W-0	

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

Table 10-23. Horizontal Size Register (HSIZE_OFF) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15-0	LNOFST	0 - FFFh	Address offset for each line. LNOFST Sets offset for each output line in external memory Either 16 or 32 pixels depending on setting of PACK8. The 5 LSB are ignored, and a zero is returned when read; the offset will be on a 32-byte boundary. For optimal performance in the system, the address offset should be on a 256-byte boundary. This bit field is latched by VD.

10.5.9 External Memory Line Offset Register (SDOFST)

The external memory line offset register (SDOFST) is shown in Figure 10-29 and described in Table 10-24.

Figure 10-29. External Memory Line Offset Register (SDOFST)

31	15
Reserved	
R-0	

14	13 12	11 9	8 6	5 3	2 0
FIINV	FOFST	LOFTS0	LOFTS1	LOFTS2	LOFTS3
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 10-24. External Memory Line Offset Register (SDOFST) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14	FIINV		Field identification signal inverse. This field is latched by VD.
		0	Non inverse
		1	Inverse
13-12	FOFST		Line offset value of field ID = 1. This field is latched by VD.
		00	+1 line
		01	+2 line
		10	+3 line
		11	+4 line
11-9	LOFTS0		Line offset values of even line and even field ID = 0. This field is latched by VD.
		000	+1 line
		001	+2 lines
		010	+3 lines
		011	+4 lines
		100	-1 line
		101	-2 lines
		110	-3 lines
		111	-4 lines
8-6	LOFTS1		Line offset values of odd line and even field ID = 0. This field is latched by VD.
		000	+1 line
		001	+2 lines
		010	+3 lines
		011	+4 lines
		100	-1 line
		101	-2 lines
		110	-3 lines
		111	-4 lines

Table 10-24. External Memory Line Offset Register (SDOFST) Field Descriptions (continued)

Bit	Field	Value	Description
5-3	LOFTS2		Line offset values of even line and odd field ID = 1. This field is latched by VD.
		000	+1 line
		001	+2 lines
		010	+3 lines
		011	+4 lines
		100	-1 line
		101	-2 lines
		110	-3 lines
		111	-4 lines
2-0	LOFTS3		Line offset values of odd line and odd field ID = 1. This field is latched by VD.
		000	+1 line
		001	+2 lines
		010	+3 lines
		011	+4 lines
		100	-1 line
		101	-2 lines
		110	-3 lines
		111	-4 lines

10.5.10 external memory Address Register (SDR_ADDR)

The external memory address register (SDR_ADDR) is shown in Figure 10-30 and described in Table 10-25.

Figure 10-30. external memory Address Register (SDR_ADDR)

31	0
SDR_ADDR	
R/W-0	

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

Table 10-25. External Memory Address Register (SDR_ADDR) Field Descriptions

Bit	Field	Value	Description
31-0	SDR_ADDR	0-FFFF FFFFh	32-bit external memory starting address for VPFE output. This bit field is latched by VD. The address should be aligned on a 32-byte boundary. Therefore, the 5 LSB's are ignored. Furthermore, reading this register will always show the 5 LSB's as 0. For optimal performance in the system, the address should be on a 256-byte boundary.

10.5.11 Optical Black Clamping Settings Register (CLAMP)

The optical black clamping settings register (CLAMP) is shown in Figure 10-31 and described in Table 10-26.

Figure 10-31. Optical Black Clamping Settings Register (CLAMP)

31	30 28	27 25	24 16
CLAMPEN	OBSLEN	OBSLN	OBST
R/W-0	R/W-0	R/W-0	R/W-0

15 10	9 5	4 0
OBST	Reserved	OBGAIN
R/W-0	R-0	R/W-0

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

Table 10-26. Optical Black Clamping Settings Register (CLAMP) Field Descriptions

Bit	Field	Value	Description
31	CLAMPEN		Clamp enable. Enable or disable clamping of CCD data based on the calculated average of optical black samples. This bit is latched by VD.
		0	Disable
		1	Enable
30-28	OBSLEN		Optical black sample length. Number of Optical Black Sample pixels per line to include in the average calculation
		000	1 pixels
		001	2 pixels
		010	4 pixels
		011	8 pixels
		100	16 pixels
		5-7	Reserved.
27-25	OBSLN		Optical black sample lines. Number of Optical Black Sample lines to include in the average calculation
		000	1 lines
		001	2 lines
		010	4 lines
		011	8 lines
		100	16 lines
		5-7	Reserved
24-10	OBST	0-7FFFh	Start pixel of optical black samples. The start pixel position of optical black samples, specified from the start of HD in pixel clocks.
9-5	Reserved	0	Any writes to these bit(s) must always have a value of 0.

Table 10-26. Optical Black Clamping Settings Register (CLAMP) Field Descriptions (continued)

Bit	Field	Value	Description
4-0	OBGAIN	0-1Fh	Gain to apply to the optical black average. Multiply the optical black average with the specified gain.
		1Fh	1 + 15/16
		1Eh	1+ 14/16
		...	...
		10h	1 + 0/16
		0Fh	0 + 15/16
		0Eh	0 + 14/16
		0Dh	0 + 13/16
		...	...
		02h	0 + 2/16
		01h	0 + 1/16
		00h	0 + 0/16

10.5.12 DC Clamp Register (DCSUB)

The DC clamp register (DCSUB) is shown in Figure 10-32 and described in Table 10-27.

Figure 10-32. DC Clamp Register (DCSUB)

31	16
Reserved	
R-0	

15	14	13	0
Reserved		DCSUB	
R-0		R/W-0	

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

Table 10-27. DC Clamp Register (DCSUB) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Any writes to these bit(s) must always have a value of 0.
13-0	DCSUB	0-3FFFh	DC level to subtract from CCD data. The DC value set here is subtracted from the CCD data when OBS clamping is disabled - CLAMP.CLAMPEN.

10.5.13 CCD Color Pattern Register (COLPTN)

The CCD color pattern register (COLPTN) is shown in Figure 10-33 and described in Table 10-28.

Figure 10-33. CCD Color Pattern Register (COLPTN)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
CP3LPC3	CP3LPC2	CP3LPC1	CP3LPC0	CP2LPC3	CP2LPC2	CP2LPC1	CP2LPC0
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
CP1LPC3	CP1LPC2	CP1LPC1	CP1LPC0	CP0LPC3	CP0LPC2	CP0LPC1	CP0LPC0
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 10-28. CCD Color Pattern Register (COLPTN) Field Descriptions

Bit	Field	Value	Description
31-30	CP3LPC3	0-3h	Color Pattern for 3rd Line, Pixel counter = 3
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
29-28	CP3LPC2	0-3h	Color Pattern for 3rd Line, Pixel counter = 2
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
27-26	CP3LPC1	0-3h	Color Pattern for 3rd Line, Pixel counter = 1
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
25-24	CP3LPC0	0-3h	Color Pattern for 3rd Line, Pixel counter = 0
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
23-22	CP2LPC3	0-3h	Color Pattern for 2nd Line, Pixel counter = 3
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
21-20	CP2LPC2	0-3h	Color Pattern for 2nd Line, Pixel counter = 2
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
19-18	CP2LPC1	0-3h	Color Pattern for 2nd Line, Pixel counter = 1
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg

Table 10-28. CCD Color Pattern Register (COLPTN) Field Descriptions (continued)

Bit	Field	Value	Description
17-16	CP2LPC0	0-3h	Color Pattern for 2nd Line, Pixel counter = 0
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
15-14	CP1LPC3	0-3h	Color Pattern for 1st Line, Pixel counter = 3
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
13-12	CP1LPC2	0-3h	Color Pattern for 1st Line, Pixel counter = 2
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
11-10	CP1LPC1	0-3h	Color Pattern for 1st Line, Pixel counter = 1
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
9-8	CP1LPC0	0-3h	Color Pattern for 1st Line, Pixel counter = 0
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
7-6	CP0LPC3	0-3h	Color Pattern for 0th Line, Pixel counter = 3
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
5-4	CP0LPC2	0-3h	Color Pattern for 0th Line, Pixel counter = 2
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
3-2	CP0LPC1	0-3h	Color Pattern for 0th Line, Pixel counter = 1
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg
1-0	CP0LPC0	0-3h	Color Pattern for 0th Line, Pixel counter = 0
		00	R/Ye
		01	Gr/Cy
		10	Gb/G
		11	B/Mg

10.5.14 Black Compensation Register (BLKCMP)

The black compensation register (BLKCMP) is shown in Figure 10-34 and described in Table 10-29.

Figure 10-34. Black Compensation Register (BLKCMP)

31	24	23	16
R_YE		GR_CY	
R/W-0		R/W-0	

15	8	7	0
GB_G		B_MG	
R/W-0		R/W-0	

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

Table 10-29. Black Compensation Register (BLKCMP) Field Descriptions

Bit	Field	Value	Description
31-24	R_YE	0-FFh	Black level compensation for R/Ye pixels (-128:+127). 2's complement, MSB is sign bit.
23-16	GR_CY	0-FFh	Black level compensation for Gr/Cy pixels (-128:+127). 2's complement, MSB is sign bit.
15-8	GB_G	0-FFh	Black level compensation for Gb/G pixels (-128:+127). 2's complement, MSB is sign bit.
7-0	B_MG	0-FFh	Black level compensation for B/Mg pixels (-128:+127). 2's complement, MSB is sign bit.

10.5.15 VPFE Interrupt Control Register (VDINT)

The VPFE interrupt control (VDINT) register is shown in Figure 10-35 and described in Table 10-30.

Figure 10-35. VPFE Interrupt Control (VDINT) Register

31	30	16
Reserved	VDINT0	
R-0	R/W-0	

15	14	0
Reserved	VDINT1	
R-0	R/W-0	

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

Table 10-30. VPFE Interrupt Control (VDINT) Register Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-16	VDINT0	0-7FFFh	CCDC_VD0_INT interrupt timing. Specify VDINT0 in units of horizontal lines from the start of VD pulse. Resulting value is VDINT0+1. Note that if the rising edge (or falling edge if programmed) of the HD lines up with the rising edge (or falling edge if programmed) of VD, the 1st HD is not counted.
15	Reserved	0	Any writes to these bit(s) must always have a value of 0.
14-0	VDINT1	0-7FFFh	CCDC_VD1_INT interrupt timing. Specify VDINT1 in units of horizontal lines from the start of VD pulse. Resulting value is VDINT1+1. Note that if the rising edge (or falling edge if programmed) of the HD lines up with the rising edge (or falling edge if programmed) of VD, the 1st HD is not counted.

10.5.16 ALAW Configuration Register (ALAW)

The ALAW configuration register (ALAW) is shown in Figure 10-36 and described in Table 10-31.

Figure 10-36. ALAW Configuration (ALAW) Register

31	8
Reserved	
R-0	

7	4	3	2	0
Reserved		CCDTBL	GWDI	
R-0		R/W-0	R/W-4	

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

Table 10-31. ALAW Configuration (ALAW) Register Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.
3	CCDTBL		Apply Gamma (A-LAW) to VPFE data saved to external memory
		0	Disable
		1	Enable
2-0	GWDI		A-law Width Input (A-LAW table)
		000	Bits 15-6
		001	Bits 14-5
		010	Bits 13-4
		011	Bits 12-3
		100	Bits 11-2
		101	Bits 10-1
		110	Bits 9-0
		Others	Reserved

10.5.17 REC656IF Configuration Register (REC656IF)

The REC656IF configuration register (REC656IF) is shown in Figure 10-37 and described in Table 10-32.

Figure 10-37. REC656IF Configuration Register (REC656IF)

31	8
Reserved	
R-0	

7	2	1	0
Reserved		ECCFVH	R656ON
R-0		R/W-0	R/W-0

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

Table 10-32. REC656IF Configuration Register (REC656IF) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Any writes to these bit(s) must always have a value of 0.
1	ECCFVH		FVH error correction enable
		0	Disable
		1	Enable
0	R656ON		REC656 interface enable
		0	Disable
		1	Enable

10.5.18 CCD Configuration Register (CCDCFG)

The CCD configuration register (CCDCFG) is shown in Figure 10-38 and described in Table 10-33.

Figure 10-38. CCD Configuration Register (CCDCFG)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
VDLC	Reserved	MSBINVI	BSWD	Y8POS	Reserved		WENLOG
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0		R/W-0
7	6	5	4	3			0
FIDMD		BW656	YCINSWP	Reserved			
R/W-0		R/W-0	R/W-0	R-0			

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

Table 10-33. CCD Configuration Register (CCDCFG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Any writes to these bit(s) must always have a value of 0.
15	VDLC		Enable latching function registers on internal VSYNC. If this bit is set, all the register fields that are VSYNC latched will take on new value immediately. Care should be taken not to alter fields that can cause undesired behavior to the output data.
		0	Latched on VSYNC
		1	Not latched on VSYNC
14	Reserved	0	Any writes to these bit(s) must always have a value of 0.
13	MSBINVI		MSB of Chroma input signal stored to external memory inverted.
		0	Normal
		1	MSB inverted
12	BSWD		Byte Swap Data stored to external memory. Number of pixels must be even if byte packing is selected if this bit is set.
		0	Normal
		1	Swap Bytes
11	Y8POS		Location of Y signal when YCbCr 8bit data is input.
		0	Even pixel
		1	Odd pixel
10-9	Reserved		Any writes to these bit(s) must always have a value of 0.
8	WENLOG		Specifies CCD valid area
		0	Internal valid signal & WEN signal is ANDed logically
		1	Internal valid signal & WEN signal is Ored logically
7-6	FIDMD		Setting of FID detection function
		00	FID signal is latched at the VSYNC timing. The external Field signal is latched when the VD is active and the active edge of the HD signal
		01	FID signal is not latched. The field signal is not latched at all.
		10	FID signal is latched at edge of VD. The field signal is latched on the active edge of the VD signal.
		11	FID signal is latched based on phase of VD and HD. The field signal is latched when the VD signal is active and the HD signal is inactive (opposite phase).
5	BW656		The data width in CCIR656 input mode. BW656 takes precedence over the SYN_MODE fields INPMOD and DATSIZ if 656 mode is enabled.
		0	8-bits
		1	10-bits

Table 10-33. CCD Configuration Register (CCDCFG) Field Descriptions (continued)

Bit	Field	Value	Description
4	YCINSWP		Y input (YIN[7:0]) and C input (CIN[7:0]) are swapped. This bit is latched on the VSYNC/VD signal
		0	YIN[7:0] = Y signal/CIN[7:0] = C signal
		1	YIN[7:0] = C signal/CIN[7:0] = Y signal
3-0	Reserved	0	Any writes to these bit(s) must always have a value of 0.

10.5.19 DMA Control Register (DMA_CNTL)

The DMA control (DMA_CNTL) register is shown in Figure 10-39 and described in Table 10-34.

Figure 10-39. DMA Control (DMA_CNTL) Register

31	30	16
OVERFLOW	Reserved	
R/W-0	R-0	

15	3	2	0
Reserved		CPRIORITY	
R-0		R/W-0	

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

Table 10-34. DMA Control (DMA_CNTL) Field Descriptions

Bit	Field	Value	Description
31	OVERFLOW		DMA Overflow Flag. Flag bit that is set when data is dropped due to a delay in writing data out of the DMA interface. This bit remains set until a "1" is written by the software.
		0	No overflow has occurred
		1	Overflow has occurred
30-3	Reserved		Any writes to these bit(s) must always have a value of 0.
2-0	CPRIORITY		Command Priority. Sets the priority that all commands should be sent with on the DMA bus. This register should only be modified when the module is inactive or it could cause violations of the CBA specifications.
		0	Highest priority
		0x7	Lowest priority

2D/3D Graphics Accelerator

This chapter describes the 2D/3D graphics accelerator (SGX).

NOTE: The SGX subsystem is an instantiation by Texas Instruments of the POWERVR™ SGX530 core from Imagination Technologies Ltd.

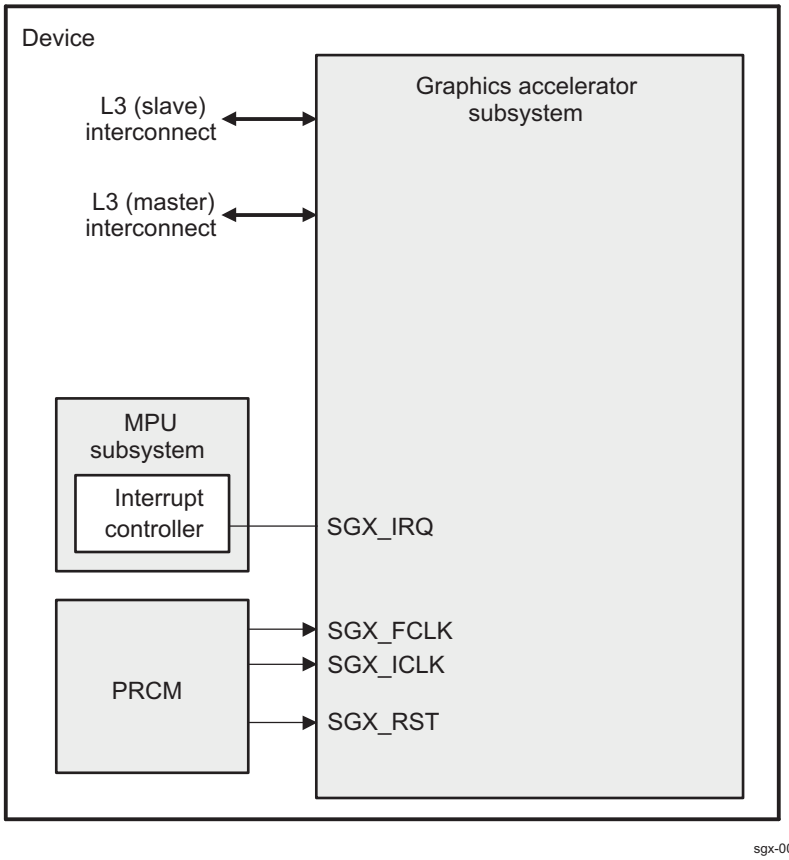
Topic **Page**

11.1 SGX Overview

The 2D/3D graphics accelerator (SGX) subsystem accelerates 2-dimensional (2D) and 3-dimensional (3D) graphics applications. The SGX subsystem is based on the POWERVR™ SGX core from Imagination Technologies. SGX is a new generation of programmable POWERVR graphic cores. The POWERVR™ SGX530 v1.2.1 architecture is scalable and can target all market segments from mainstream mobile devices to high-end desktop graphics. Targeted applications include feature phone, PDA, and hand-held games.

Figure 11-1 shows the SGX subsystem in the device.

Figure 11-1. Graphics Accelerator Highlight



The SGX graphics accelerator efficiently processes a number of various multimedia data types concurrently:

- Pixel data
- Vertex data
- Video data
- General-purpose processing

This is achieved using a multithreaded architecture using two levels of scheduling and data partitioning enabling zero overhead task switching.

The SGX subsystem is connected by a 64-bit master and a 32-bit slave interface to the L3 interconnect.

11.1.1 POWERVR SGX Main Features

- 2D graphics, 3D graphics, vector graphics, and programming support for GP-GPU functions
- Tile-based architecture
- USSE™ – multithreaded engine incorporating pixel and vertex shader functionality

- Advanced shader feature set – in excess of Microsoft VS3.0, PS3.0, and OpenGL2.0
- Industry standard API support – Direct3D Mobile, OpenGL ES 1.1 and 2.0, OpenVG v1.0.1
- Fine-grained task switching, load balancing, and power management
- Advanced geometry direct memory access (DMA) driven operation for minimum CPU interaction
- Programmable high-quality image anti-aliasing
- POWERVR SGX core MMU for address translation from the core virtual address to the external physical address (up to 4GB address range)
- Fully virtualised memory addressing for OS operation in a unified memory architecture
- Advanced and standard 2D operations [e.g., vector graphics, BLTs (block level transfers), ROPs (raster operations)]

11.1.2 SGX 3D Features

- Deferred pixel shading
- On-chip tile floating point depth buffer
- 8-bit stencil with on-chip tile stencil buffer
- 8 parallel depth/stencil tests per clock
- Scissor test
- Texture support:
  - Cube map
  - Projected textures
  - 2D textures
  - Nonsquare textures
- Texture formats:
  - RGBA 8888, 565, 1555
  - Monochromatic 8, 16, 16f, 32f, 32int
  - Dual channel, 8:8, 16:16, 16f:16f
  - Compressed textures PVR-TC1, PVR-TC2, ETC1
  - Programmable support for all YUV formats
- Resolution support:
  - Frame buffer maximum size = 2048 x 2048
  - Texture maximum size = 2048 x 2048
- Texture filtering:
  - Bilinear, trilinear
  - Independent minimum and maximum control
- Antialiasing:
  - 4x multisampling
  - Up to 16x full scene anti-aliasing
  - Programmable sample positions
- Indexed primitive list support
  - Bus mastered
- Programmable vertex DMA
- Render to texture:
  - Including twiddled formats
  - Auto MipMap generation
- Multiple on-chip render targets (MRT).
  Note: Performance is limited when the on-chip memory is not available.

11.1.3 Universal Scalable Shader Engine (USSE) – Key Features

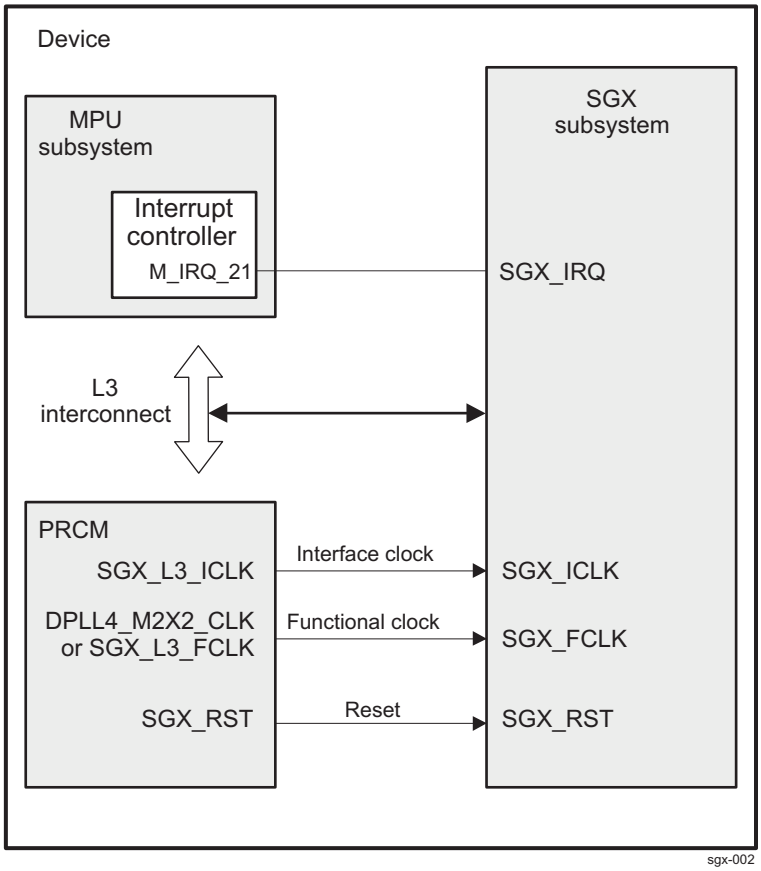
The USSE is the engine core of the POWERVR SGX architecture and supports a broad range of instructions.

- Single programming model:
  - Multithreaded with 16 simultaneous execution threads and up to 64 simultaneous data instances
  - Zero-cost swapping in, and out, of threads
  - Cached program execution model
  - Dedicated pixel processing instructions
  - Dedicated video encode/decode instructions
- SIMD execution unit supporting operations in:
  - 32-bit IEEE float
  - 2-way 16-bit fixed point
  - 4-way 8-bit integer
  - 32-bit bit-wise (logical only)
- Static and dynamic flow control:
  - Subroutine calls
  - Loops
  - Conditional branches
  - Zero-cost instruction predication
- Procedural geometry:
  - Allows generation of primitives
  - Effective geometry compression
  - High-order surface support
- External data access:
  - Permits reads from main memory using cache
  - Permits writes to main memory
  - Data fence facility
  - Dependent texture reads

11.2 SGX Integration

Figure 11-2 highlights the SGX subsystem integration in the device.

Figure 11-2. SGX Subsystem Integration



11.2.1 Clocking, Reset, and Power-Management Scheme

11.2.1.1 Clocks

The SGX subsystem operates from two clocks: an interface clock (SGX_ICLK) and a functional clock (SGX_FCLK). The power, reset, and clock management (PRCM) module generates and distributes both clocks inside the device. The SGX clock tree is depicted in Figure 4-31, *SGX Power Domain Clocking Scheme*, in , *Power, Reset, and Clock Management*.

Table 11-1. Clock Descriptions

Signal Name	I/O[1]	Description
SGX_FCLK	I	Functional clock (two possible clock sources) -> Functional clock domain
SGX_ICLK	I	Interface clock (L3 interconnect clock domain) -> Interface clock domain

(1) I =Input, O=Output,

- The SGX_ICLK interface clock manages the data transfer on the L3 master and slave ports.

  The source of SGX_ICLK is the PRCM clock (SGX_L3_ICLK), which belongs to the SGX clock domain and runs at the L3 interconnect clock speed. The SGX_ICLK frequency is selected based on the whole device L3 interconnect clock frequency. For more information on the interface clock, see , *Power, Reset, and Clock Management*.

  When no longer required by the SGX subsystem, SGX_ICLK can be disabled by software at the PRCM level by setting the PRCM.CM_ICLKEN_SGX[0] EN_SGX bit to 0. For more information, see

Figure 4-47, *SGX Power Domain Clock Controls*, in , *Power, Reset, and Clock Management*.

NOTE: SGX_ICLK is cut only if the SGX is ready to go into idle state. For more information, see , *Power, Reset, and Clock Management*.

- SGX_FCLK is the functional clock and is used inside the SGX subsystem to generate SGX 2D and 3D domain clock signals.

  The source of SGX_FCLK is either the PRCM clock (SGX_L3_FCLK, which is derived from SGX_ICLK) or the peripheral DPLL clock DPLL4_M2X2_CLK as depicted in Figure 4-47, *SGX Power Domain Clock Controls*. Selection is made at the PRCM level by setting the PRCM.CM_CLKSEL_SGX[2:0] CLKSEL_SGX bit field. A divider of 3, 4, or 6 is applied to the SGX_FCLK frequency with regard to the PRCM SGX_L3_FCLK frequency. By default the SGX_FCLK clock is SGX_L3_FCLK / 3.

  Whether SGX_FCLK is provided by SGX_L3_FCLK or DPLL4_M2X2_CLK, its gating depends on the PRCM.CM_FCLKEN_SGX[1] EN_SGX bit. Clearing this bit to 0 indicates that both SGX clocks are no longer needed and can be cut at the PRCM level, if the module is ready to enter the idle state.For more information, see Section 4.5.5.6, *SGX Power Domain*, in , *Power, Reset, and Clock Management*

11.2.1.2 Resets

The SGX subsystem has its own reset domain. Global reset of the SGX is performed by activating the SGX_RST signal in the SGX_RST domain. Software controls the release of SGX_RST using the PRCM.RM_RSTCTRL_SGX[0] SGX_RST bit.

11.2.1.3 Power Management

The SGX subsystem has its own power domain (SGX power domain). See , *Power, Reset, and Clock Management*, for additional information about the SGX power domain.

As described in Section 11.2, *SGX Integration*, the SGX subsystem receives two clock signals from the PRCM module. The functional clock is either a division of the interface clock with a ratio of 1:3, 1:4, or 1:6, or a 96–MHz clock issued from a DPLL4. The functional clock is used inside the SGX to generate clock signals to the multiple internal module SGX clock domains. The division ratio depends on the PRCM registers setting. For more information, see , *Power, Reset, and Clock Management*.

Three power-management modes are defined:

- Deep power sleep (All clocks are gated.)
- Idle (2D and 3D clocks are gated.)
- 3D (No clock is gated.)

The SGX handles the automatic clock gating performed on the multiple internal module clock domains.

11.2.2 Hardware Requests

11.2.2.1 Interrupt Request

The SGX subsystem can generate one interrupt (SGX_IRQ) to the MPU subsystem interrupt controller mapped on M_IRQ_21.

11.3 SGX Functional Description

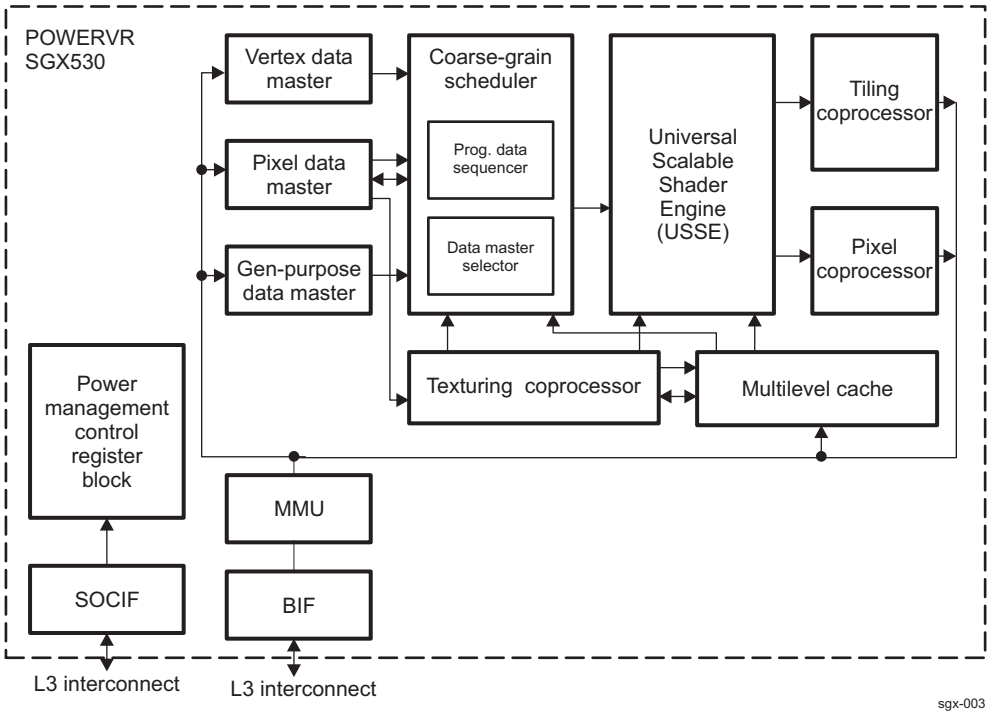
11.3.1 SGX Block Diagram

The SGX subsystem is based on the POWERVR SGX530 core from Imagination Technologies. The architecture uses both programmable and hard coded pipelines to perform various processing tasks required in 2D, 3D, and video processing. The SGX architecture comprises the following elements:

- Coarse grain scheduler
  - Programmable Data Sequencer (PDS)
  - Data Master Selector (DMS)
- Vertex data master (VDM)
- Pixel data master (PDM)
- General-purpose data master
- USSE
- Tiling coprocessor
- Pixel coprocessor
- Texturing coprocessor
- Multilevel cache

Figure 11-3 shows a block diagram of the SGX cores.

Figure 11-3. SGX Block Diagram



11.3.2 SGX Elements Description

The coarse grain scheduler (CGS) is the main system controller for the POWERVR SGX Architecture. It consists of two stages, the DMS and the PDS. The DMS processes requests from the data masters and determines which tasks can be executed given the resource requirements. The PDS then controls the loading and processing of data on the USSE.

There are three data masters in the SGX core:

- The VDM is the initiator of transform and lighting processing within the system. The VDM reads an

input control stream, which contains triangle index data, and state data. The state data indicates the PDS program, size of the vertices and the amount of USSE output buffer resource available to the VDM. The triangle data is parsed to determine unique indices that must be processed by the USSE. These are grouped together according to the configuration provided by the driver and presented to the DMS.

- The PDM is the initiator of rasterization processing within the system. Each pixel pipeline processes pixels for a different half of a given tile which allows for optimum efficiency within each pipe due to locality of data. For each task it determines the amount of resource required within the USSE. It merges this with the state address and issues a request for execution on the USSE to the DMS.
- The general-purpose data master responds to events within the system (such as end of a pass of triangles from the ISP, end of a tile from the ISP, end of render, or parameter stream breakpoint event). Each event causes either an interrupt to the host or synchronized execution of a program on the PDS. The program may, or may not cause a subsequent task to be executed on the USSE.

The USSE is a user-programmable processing unit. Although general in nature, its instructions and features are optimized for three types of task: processing vertices (vertex shading), processing pixels (pixel shading), and video/imaging processing.

The multilevel cache is a 2-level cache consisting of two modules: the main cache and the mux/arbiter/demux/decompression unit (MADD). The MADD is a wrapper around the main cache module designed to manage and format requests to and from the cache, as well as providing Level 0 caching for texture and USSE requests. The MADD can accept requests from the PDS, USSE, and texture address generator modules. Arbitration is performed between the three data streams; any required texture decompression is also performed.

The texturing coprocessor performs texture address generation and formatting of texture data. It receives requests from either the iterators or USSE modules and translates these into requests into the multilevel cache. Data returned from the cache are then formatted according to the texture format selected and sent to the USSE for pixel-shading operations.

To process pixels in a tiled manner, the screen is divided into tiles and arranged as groups of tiles by the tiling coprocessor. An inherent advantage of tiling architecture is that a large amount of vertex data can be rejected at this stage, thus reducing both the memory storage requirements and the amount of pixel processing to be performed.

The pixel coprocessor is the final stage of the pixel-processing pipeline and controls the format of the final pixel data sent to the memory. It supplies the USSE with an address into the output buffer, and the USSE returns the relevant pixel data. The address order is determined by the frame buffer mode. The pixel coprocessor contains a dithering and packing function.

11.4 SGX Register Mapping

Table 11-2 describes the SGX memory mapping in the device.

Table 11-2. Instance Summary

Module Name	Start Address	End Address	Size
SGX	0x5000 0000	0x5000 FFFF	64K bytes

CAUTION

The SGX registers are limited to 32-bit data accesses; 8- and 16-bit accesses are not allowed because they can corrupt register content.

Display Subsystem

This chapter describes the display subsystem.

Topic **Page**

12.1 Display Subsystem Overview

The display subsystem provides the logic to display a video frame from the memory frame buffer (either SDRAM or SRAM) on a liquid-crystal display (LCD) panel or a TV set. The display subsystem integrates the following elements:

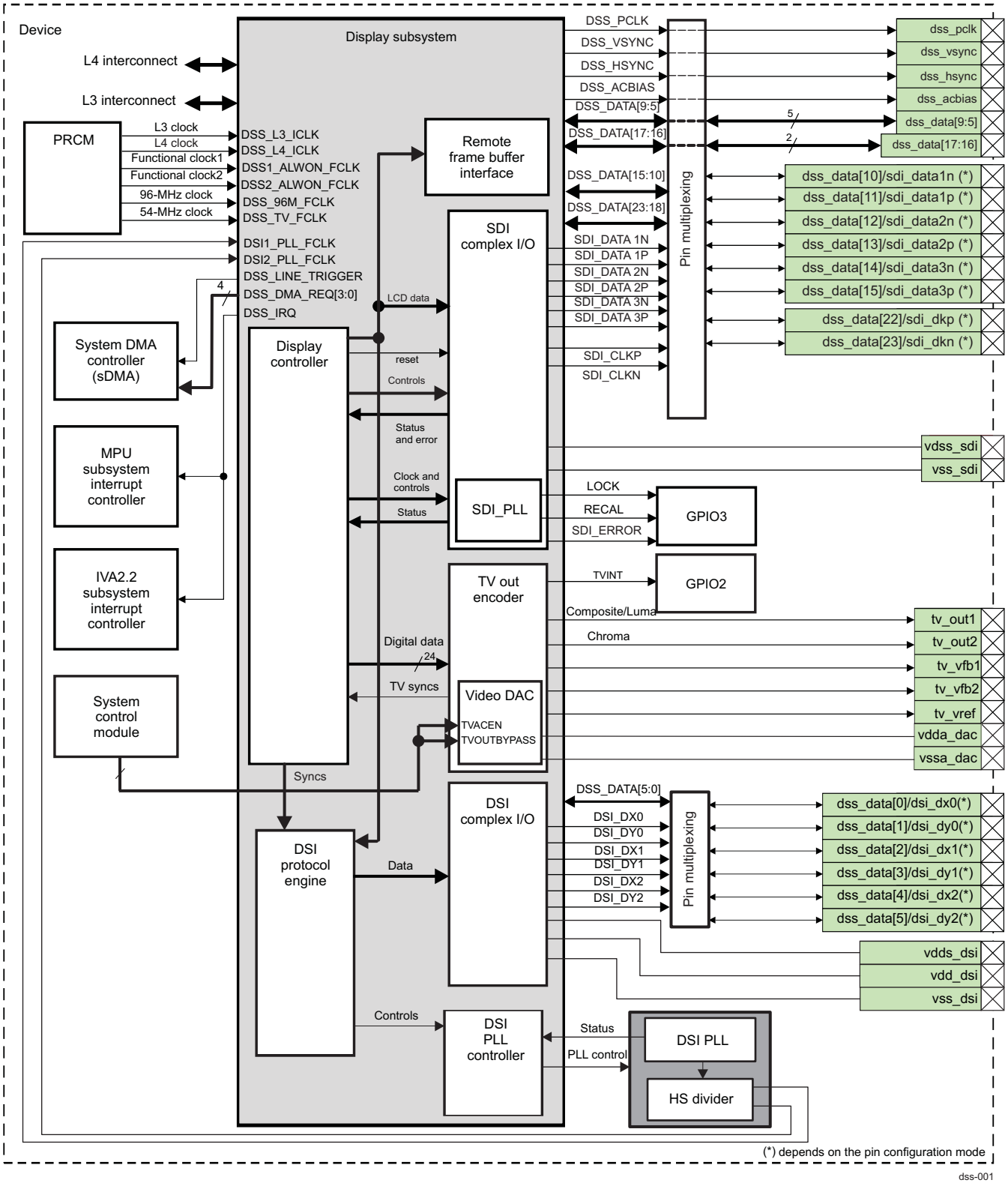
- Display controller (DISPC) module
- Remote frame buffer interface (RFBI) module
- Serial display interface (SDI) complex input/output (I/O) module with the associated phased-locked loop (PLL)
- Display serial interface (DSI) complex I/O module and a DSI protocol engine
- DSI PLL controller that drives a DSI PLL and high-speed (HS) divider.
- NTSC/PAL video encoder

The display controller and the DSI protocol engine are connected to the L3 and L4 interconnect; the RFBI and the TV out encoder modules are connected to the L4 interconnect.

NOTE: The SDI module, the DSI complex I/O module, and the DSI PLL controller are not connected to an L3 or L4 interconnect. Specific display subsystem registers manage their programmable features.

Figure 12-1 shows a block diagram of the display subsystem.

Figure 12-1. Display Subsystem Highlight



NOTE: For more information about connecting the LOCK, RECAL, SDI_ERROR, and TVINT signals through the GPIO2 and GPIO3 modules, see Chapter 21, *GPIO*.

The display subsystem includes the following main features:

- Display controller
  - Display modes
    - Programmable pixel display modes (1, 2, 4, 8, 12, 16, and 24 bits-per-pixel [BPP] modes)
    - Programmable display size supported:
      - XGA - 1024 × 768 VESA timings at 60 fps (pixel clock = 63.5 MHz)
      - WXGA - 1280 × 800 VESA timings at 59.91 fps (pixel clock = 71 MHz)
      - SXGA+ - 1400 × 1050 direct drive of LCD with minimal blanking at 50 fps (pixel clock = 75 MHz)
      - HD 720p - 1280 × 720 at 60 fps (pixel clock = 74.25 MHz)
    - 256 × 24-bit entries palette in red, green, and blue (RGB)
    - Programmable pixel rate up to 75 MHz

    NOTE: The panel size is programmable and can be any width that is a multiple of 8 pixels (line length) in the range [1:2048] pixels (in the case of the RFBI mode, the minimum transfer size is a byte). The maximum resolution is 2048 (lines) x 2048 (pixels).

  - Display support
    - Four types of displays are supported: Passive (super-twist nematic [STN]) and active (thin-film transistor [TFT]) colors, passive (STN), and active (TFT) monochromes.
    - 4-/8-bit monochrome passive matrix panel interface support (15 grayscale levels supported using dithering block)
    - 8-bit color passive matrix panel interface support (3375 colors supported for a color panel using dithering block)
    - 12-/16-/18-/24-bit active matrix panel interface support (replicated or dithered encoded pixel values)
    - Remote frame buffer support through the RFBI module
    - Partial display through the RFBI module
    - Second 24-bit digital output
    - Multiple-cycle output format on 8-/9-/12-/16-bit interface time division multiplexing (TDM)
    - HDMI through external bridge
  - Signal processing
    - Overlay support for graphics (ARGB, RGBA, RGB, or Color Look-Up Table (CLUT)) and video1 (YCbCr 4:2:2, or ARGB, RGBA, RGB), video2 (YCbCr 4:2:2, or ARGB, RGBA, RGB)
    - Programmable video resizer independent horizontal and vertical resampling: Upsampling (up to x8) and downsampling (down to 1/4), maximum input width of 1024 pixels in 5-tap mode, and 2048 pixels in 3-tap configurations; no limitation on input height
    - Rotation 90-, 180-, and 270-degrees
    - Transparency color key (source and destination)
    - Synchronized buffer update
    - Programmable video color space conversion YCbCr 4:2:2 into RGB
    - Hardware cursor
    - Gamma curve support on LCD output
    - Multiple-buffer support
    - Mirroring support
    - Programmable color phase rotation (CPR)
    - Alpha blending support (no rescaling in ARGB or RGBA formats)
  - Advanced

- Self-refresh using the DMA FIFO
- Arbitration between high/low priority (graphics video1 and video2)
- FIFO handcheck in STALL mode

– Power modes: Low-power saving modes

- RFBI (MIPI® DBI protocol)
  – Access to remote frame buffer (RFB) direct microprocessor unit (MPU) interface
    - Sends commands to the RFB panel through the L4 interconnect
    - Sends data, received from the display controller or from the MPU through the DISPC pixel data bus, to the RFB panel
    - Reads data/status from the RFB to the L4 interconnect
  – RFB interface
    - 8-/9-/12-/16-bit 8086-series parallel interface
    - Two programmable configurations for two devices connected to the RFBI module
  – Data formats
    - Programmable pixel modes (12-/16-/18-/24-BPP modes in RGB format)
    - Programmable output formats on one/multiple cycles per pixel (data from the display controller and from the L4 interconnect)
  – Interconnect/FIFO
    - One slave port with DMA request and interconnect FIFO of 24x32-bit depth (for write access to DSS.RFBI_DATA register only)
    - One video port FIFO of 8 × 24-bit depth receiving data from the display controller
- SDI
  – TI Flatlink™ 3G display interface support
  – Pin multiplexing allows simultaneous operation with a single 9-bit RFBI-driven display (no support for simultaneous dual RFBI panel and SDI)

NOTE: The SDI pins are multiplexed with LCD parallel outputs.

- MIPI DSI
  – Transfer pixels and data received on the video port or L4 interconnect to the display through the DSI DSI_PHY
  – The maximum resolution supported on the video port is XGA at 60 fps with 24-bit pixels (maximum pixel clock of 67 MHz) for low voltage.
  – Supports video mode and command mode
  – Bidirectional data link support (only one data lane is used in reverse direction in command mode)
  – Supports up to two data-configurable lanes, in addition to the clock signaling (minimum of one data link and maximum of two, depending on speed, signal integrity requirements, and number of displays)
  – Maximum data rate of 800 Mbps per data pair
  – Data splitter for 2-data lane configuration
  – Error-correction code (ECC) and check-sum generation
  – Burst support for the video mode
  – RGB16, RGB18 packed and nonpacked, and RGB24 formats supported for video mode
  – Serial configuration port (SCP) for the DSI_PHY complex I/O and DSI PLL
  – Connection to the DSI_PHY complex I/O through PPI
  – Data interleaving support for one synchronous stream (video mode) from the display controller and up to three interleaved asynchronous streams (command mode) from the interconnect concurrently
  – Data interleaving supports up to four interleaved asynchronous streams (command mode) from the

interconnect or video port when there is no video mode
  - MIPI DCS support (transparent to the protocol engine, no decoding and interpretation of the information from and to the peripheral)
  - Supports selection between low-power state and HS mode between HS packet transfers
  - Generic data type (DT) support

NOTE: The DSI pins are multiplexed with LCD parallel outputs.

- Video encoder
  - NTSC/PAL encoder outputs with the following standards:
    - NTSC-J, M
    - PAL-B, D, G, H, I
    - PAL-M
  - CGMS-A as described in the CEA-608-x Standard.
  - Input data interface compatible with the following protocols:
    - 24-bit input bus compatible with external sync
    - RGB 4:4:4
  - Dual output data 10-bit interface for internal digital-to-analog converter (DAC) that supports:
    - Composite video (CVBS)
    - Separate video (S-video)
  - TV output data supports ITU-R BT 470-7 recommendation standard for consumer market
  - Master clock input 13.5 MHz, 27 MHz (supports ITU-R 601 sampling for NTSC/PAL), and 54 MHz
  - Programmable horizontal sync, vertical timing, and waveforms
  - Programmable subcarrier frequency and SCH
  - Internal test pattern generation (color bar, flat field, color burst)
  - 2x/4x oversampling
  - Supports square pixel sampling (NTSC: 12.27 MHz, 24.54 MHz, 49.09 MHz PAL: 14.75 MHz, 29.5 MHz, 59 MHz)

CAUTION

In square pixel mode, an external clock generator is required to provide sampling frequencies.

  - TV detection gating pulse generation

12.2 Display Subsystem Environment

This section describes the two main functions handled by the display subsystem:

- LCD support
- TV display support

12.2.1 LCD Support

LCD panels can be connected to the display subsystem of the device using parallel and/or serial interfaces.

12.2.1.1 Parallel Interface

In parallel interface, the paths of the display subsystem modules are the display controller and the RFBI.

The display controller provides the required control signals to interface the memory frame buffer (SDRAM or SRAM) directly to the external displays. The display controller is connected to the memory through the L3 interconnect and has its own DMA (with embedded FIFOs) to read data from the system memory. The L3 interconnect is the master port, while the L4 interconnect is the slave port of the display subsystem.

The display controller has two I/O pad modes at the module level:

- RFBI mode (RFBI enabled), which implements the MIPI DBI 2.0 protocol
- Bypass mode (RFBI disabled), which implements the MIPI DPI 1.0 protocol

The DSS.DISPC_CONTROL[16:15] GPOUT[1:0] bits control selection of the display subsystem modules (see Table 12-1).

Table 12-1. I/O Pad Mode Selection

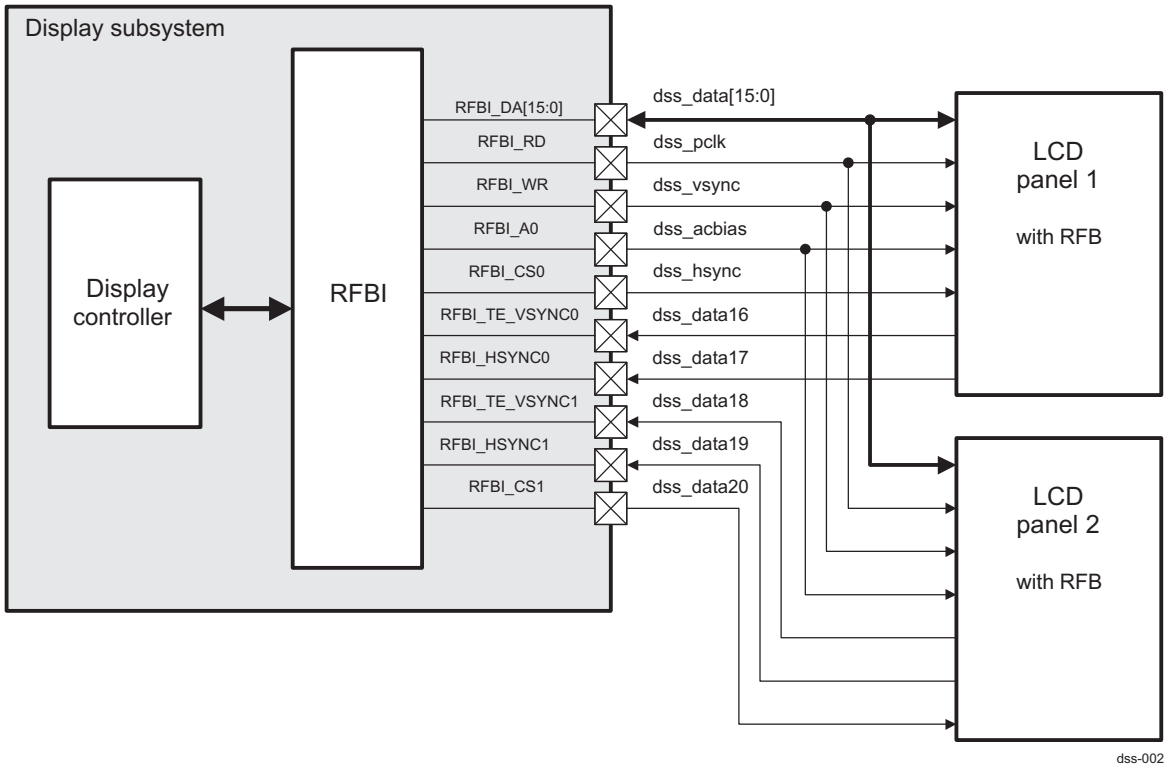
DSS.DISPC_CONTROL[16] GPOUT1	DSS.DISPC_CONTROL[15] GPOUT0	Mode
0	0	Reset
0	1	RFBI mode
1	0	Invalid
1	1	Bypass mode

The RFB of the LCD panel is directly connected to the RFBI module of the device. The RFBI controls the reads/writes from/to the RFB. The RFBI receives the output from the DISPC (which takes data from the memory) and generates the signals to control the LCD panel. Through the RFBI, the MPU can send commands or parameter/display data to the LCD panel and directly set the DISPC registers to read/write the data from/to the memory in the LCD panel. The RFBI can manage up to two LCD panels when the serial interface is not used.

12.2.1.1.1 Parallel Interface in RFBI Mode (MIPI DBI Protocol)

Figure 12-2 shows the LCD support parallel interface in RFBI mode (example for 16-bit data interface).

Figure 12-2. LCD Support Parallel Interface (RFBI Mode)



NOTE: If the SDI is used, the second display device connected to RFB#1 is limited to 9-bit data.

Configure the DSS.RFBI_CONTROL[3:2] CONFIGSELECT bit field to drive signals for LCD 1 only, LCD 2 only, or both LCD 1 and LCD 2.

Table 12-2 describes the interface signals to/from the LCD panel in RFBI mode.

Table 12-2. LCD Interface Signals (RFBI Mode)

Signal Name	Type[1]	Description
RFBI_DA[15:0]	I/O	RFBI I/O data
RFBI_RD	O	Read access signal
RFBI_WR	O	Write access signal
RFBI_A0	O	Command/data selection signal
RFBI_CS0	O	Chip-select (CS) signal for LCD 1
RFBI_CS1	O	CS signal for LCD 2
RFBI_TE_VSYNC0	I	Tearing effect (TE) synchronization signal (TE or VSYNC for LCD panel 1)
RFBI_HSYNC0	I	HSYNC from LCD panel 1
RFBI_TE_VSYNC1	I	TE synchronization signal (TE or VSYNC for LCD panel 2)
RFBI_HSYNC1	I	HSYNC from LCD panel 2

(1) I = Input, O = Output

- RFBI_DA[15:0]: The pixel data comprises the RFBI pixel data (bits 15:0). A write/read command must be sent to the LCD panel to send/read the data.

  Before any data access, the application must send commands and parameters when it is necessary to configure an LCD panel. The data is used as input in read operations during production test and also to read the status of the registers in the LCD panel and pixels from the embedded frame buffer in the LCD panel module. RFBI_DA is multiplexed at the chip-level boundary with dss_data [15:0].

- RFBI_RD: This is the read-enable signal used to indicate when a read from the embedded memory in the LCD panel is ongoing. The RFBI registers describe the behavior of the read signal (off/on/cycle time). The polarity of the read-enable signal is programmable. This signal is multiplexed at the chip-level boundary with dss_pclk. The read is used to get status/data information from the LCD panel.
- RFBI_WR: The write-enable signal is used to indicate when a write is ongoing. The RFBI registers describe the behavior of the write signal (off/on/cycle time). The polarity of the write-enable signal is programmable. This signal is multiplexed at the chip-level boundary with dss_vsync.
- RFBI_A0: The signal is asserted to indicate its status: Command or data. The polarity is programmable and the status of the signal depends on the RFBI registers written by the application (CMD/READ/STATUS/PARAM/PIXEL). The register in use by the hardware defines the status of RFBI_A0. The order of the writes/reads to the RFBI registers CMD/READ/STATUS/PARAM/PIXEL defines the transitions of A0. This signal is multiplexed at the chip-level boundary with dss_acbias.
- RFBI_CS*x*: The signal is the chip-select (CS*x*) asserted to indicate which LCD panel is selected and must be ready to receive/transmit commands and data. When RE or WE is on, CS*x* must not be changed (*x* = 0 for LCD panel 1; *x* = 1 for LCD panel 2). CS0 is multiplexed at the chip-level boundary with dss_hsync, and CS1 is multiplexed at the chip-level boundary with dss_data[20].
- RFBI_TE_VSYNC*x*: Based on the trigger mode selected, the signal is the TE pulse signal or the LCD panel VSYNC (vertical synchronization) pulse signal. RFBI_TE_VSYNC*x* is used by the TE logic as the synchronization signal to send the pixel to the LCD panel.

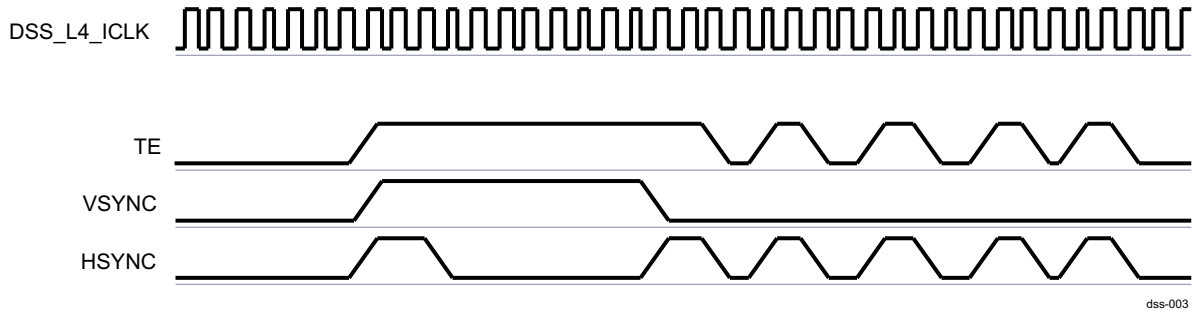
  To select the trigger mode, configure the DSS.RFBI_CONFIGi[3:2] TRIGGERMODE bit field (0x0: Internal trigger mode with the DSS.RFBI_CONTROL[4] ITE bit, 0x1: External trigger mode with the TE signal RFBI_TE_VSYNC*x*, 0x2: External trigger mode with the RFBI_TE_VSYNC*x*, and RFBI_HSYNC*x* signals with the programmable line counter).

  These signals are multiplexed at the chip-level boundary with dss_data[16] (RFBI_TE_VSYNC0) and dss_data[18] (RFBI_TE_VSYNC1) (LCD panel 1: *x* = 0; LCD panel 2: *x* = 1).
- RFBI_HSYNC*x*: The HSYNC pulse signals indicate to the RFBI module when horizontal synchronization occurs. The polarity of the HSYNC signals is programmable. The minimum pulse width of the signal is two L4 cycles. RFBI_HSYNC is used by the TE logic as a synchronization signal to send the pixel to the LCD panel. These signals are multiplexed at the chip-level boundary with dss_data[17] (RFBI_HSYNC0) and dss_data[19] (RFBI_HSYNC1) (LCD panel 1: *x* = 0; LCD panel 2: *x* = 1).

12.2.1.1.1.1 Description of the TE Pulse Signal

The externally-generated TE synchronization signal is a logical OR or AND operation between the HSYNC and VSYNC signals (see Figure 12-3). The logical operation (OR or AND) depends on the HSYNC and VSYNC signals polarity. The VSYNC signal indicates to the RFBI module when vertical synchronization occurs; the HSYNC signal indicates to the RFBI module when horizontal synchronization occurs.

Figure 12-3. External Generation of TE Signal Based on Logical OR Operation Between HSYNC and VSYNC (Active-High)



The RFBI module detects the VSYNC and HSYNC pulses embedded in the received signal. VSYNC is detected based on the minimum pulse width defined by the DSS.RFBI_VSYNC_WIDTH register.

HSYNC is detected based on the minimum pulse width defined by the DSS.RFBI_HSYNC_WIDTH register.

The signal is generated from external logic based on the VSYNC/HSYNC of the LCD panel. The automatic trigger can be programmed based on the RFBI_TE signal or use a bit field in the RFBI registers to start data capture.

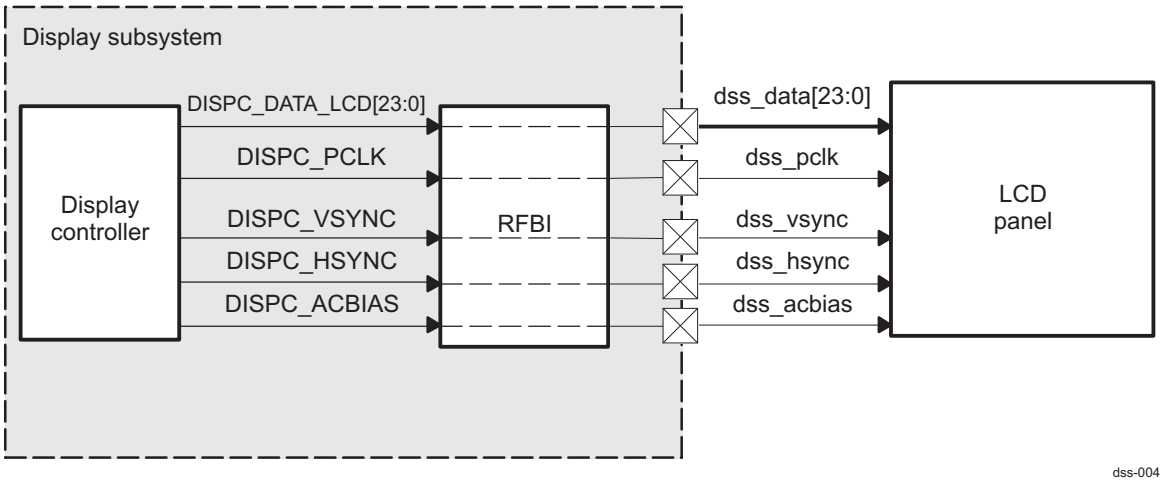
The polarity of the TE signal is programmable. The HSYNC and VSYNC pulses embedded in the TE signal have the same polarity, which is active high for an ORed signal and active low for an ANDed signal. The minimum pulse width of the signal is two L4 cycles. Hardware resets the line counter when the VSYNC occurs and increments it at every HSYNC. Transfer to the LCD panel begins when the line counter reaches the programmable line number.

12.2.1.1.2 Parallel Interface in Bypass Mode (MIPI DPI Protocol)

When bypass mode is enabled, the display controller must be set to use it.

Figure 12-4 shows the LCD support parallel interface in bypass mode.

Figure 12-4. LCD Support Parallel Interface (Bypass Mode)



NOTE: In bypass mode, the SDI and the parallel interface cannot be used at the same time.

Table 12-3 describes the interface signals to/from the LCD panel in bypass mode.

Table 12-3. LCD Interface Signals (Bypass Mode)

Signal Name	Type[1]	Description
DISPC_DATA_LCD[23:0]	O	LCD data from the display controller module
DISPC_PCLK	O	Pixel CLK from the display controller module
DISPC_VSYNC	O	VSYNC from the display controller module
DISPC_HSYNC	O	HSYNC from the display controller module
DISPC_ACBIAS	O	ACBIAS from the display controller module

(1) I = Input, O = Output, I/O = Input/Output

- DISPC_DATA_LCD[23:0]: The panel pixel data comes directly from the display controller module. DISPC_DATA_LCD is connected at the chip-level boundary with dss_data[23:0].
- DISPC_PCLK: This signal is the pixel clock that comes directly from the display controller. This signal is multiplexed at the chip-level boundary with dss_pclk.
- DISPC_VSYNC: Uses the vertical synchronization signal from the display controller. The LCD frame clock (VSYNC) toggles after all the lines in a frame are transmitted to the LCD panel and a programmable number of line clock cycles has elapsed both at the beginning and at the end of each frame. This signal is multiplexed with dss_vsync at the chip-level boundary.
- DISPC_HSYNC: Uses the horizontal synchronization signal from the display controller. The LCD line clock (HSYNC) toggles after all pixels in a line are transmitted to the LCD panel and a programmable

number of pixel clock wait-states elapse, both at the beginning and at the end of each line. This signal is multiplexed on the chip-level boundary with dss_hsync.

- DISPC_ACBIAS: Uses the ac-bias signal from the display controller.
  - In passive matrix technology, the ac-bias signal is configured to transition each time a programmable number of line clocks occurs. To prevent a dc charge within the screen pixels, the power and ground supplies of the panel are periodically switched. The DISPC signals the panel to switch the polarity by toggling the ac-bias pin.
  - In active matrix technology, the ac-bias signal acts as an output-enable signal to indicate when data must be latched using the pixel clock. This signal is multiplexed on the chip-level boundary with dss_acbias.

12.2.1.1.3 LCD Output and Data Format for the Parallel Interface

This section describes the pixel data bus and shows timing diagrams of transactions and synchronizations in both RFBI and bypass modes.

Figure 12-5 through Figure 12-11 show the pixel data bus for bypass mode, depending on the use of 4-, 8-, 12-, 16-, 18-, or 24-pixel data output pins. In RFBI mode, the pixel data bus is reformatted in accordance with the input and output data bus width.

Table 12-4 lists the number of displayed pixels per pixel clock cycle based on the type of display panel.

Table 12-4. Number of Displayed Pixels per Pixel Clock Cycle Based on Display Type

Display Panel	Number of Displayed Pixels per Pixel Clock Cycle
Monochrome 4-bit	4
Monochrome 8-bit	8
Passive matrix color	8/3
Active matrix	1

- Passive matrix technology, Monochrome mode

  Monochrome displays use either a 4-bit or 8-bit interface. Each bit represents one pixel (on or off), which means that either 4 or 8 pixels are sent to the LCD at each pixel clock.

  Figure 12-5 and Figure 12-6 show 4- and 8-bit monochrome displays, respectively.

Figure 12-5. LCD Pixel Data Monochrome4 Passive Matrix

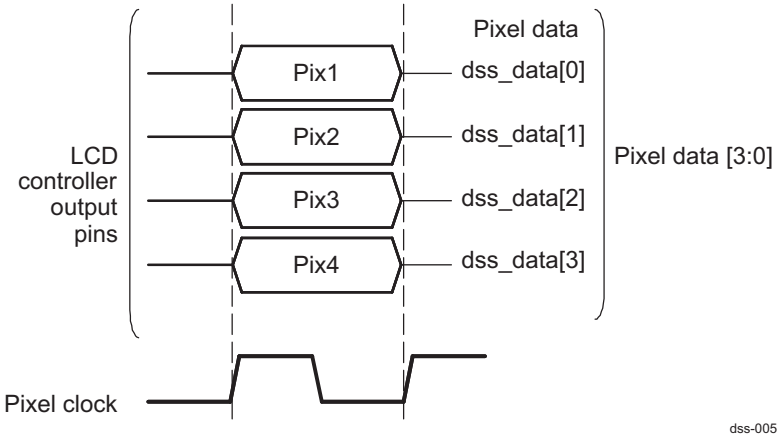
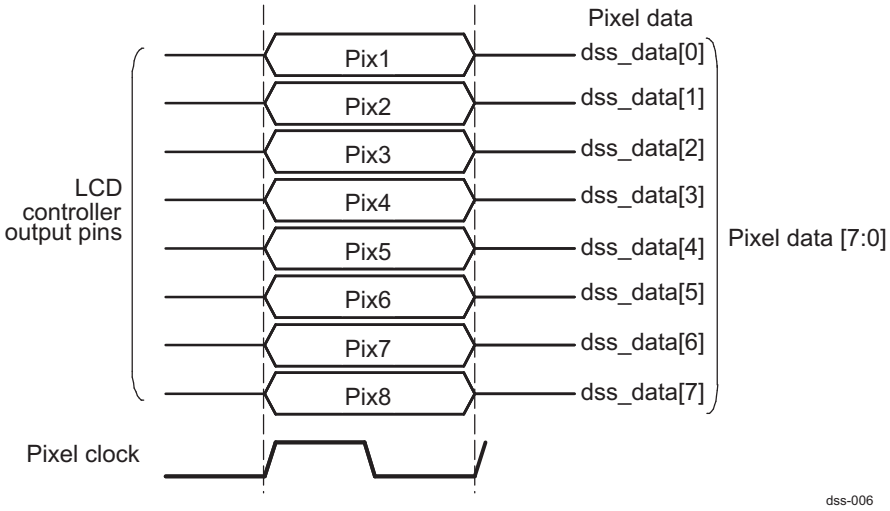


Figure 12-6. LCD Pixel Data Monochrome8 Passive Matrix

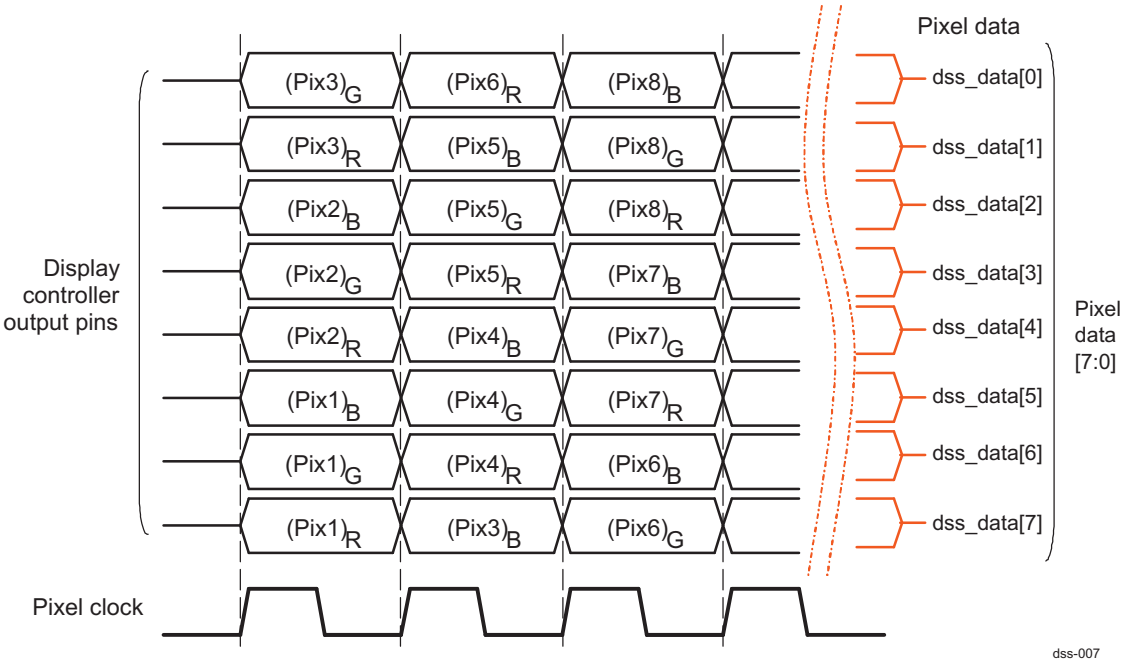


- Passive matrix technology, color mode

  Color passive displays use 8-bit data input lines. In a given pixel clock cycle, each line represents one color component (red, green, or blue).

  Figure 12-7 shows an 8-bit color passive matrix display.

Figure 12-7. LCD Pixel Data Color Passive Matrix



- Active matrix technology

  Active matrix displays bypass the STN dithering logic block and the output FIFO. Each line represents one pixel.

  Figure 12-8 through Figure 12-11 show 12-, 16-, 18-, and 24-active matrix displays, respectively.

Figure 12-8. LCD Pixel Data Color12 Active Matrix

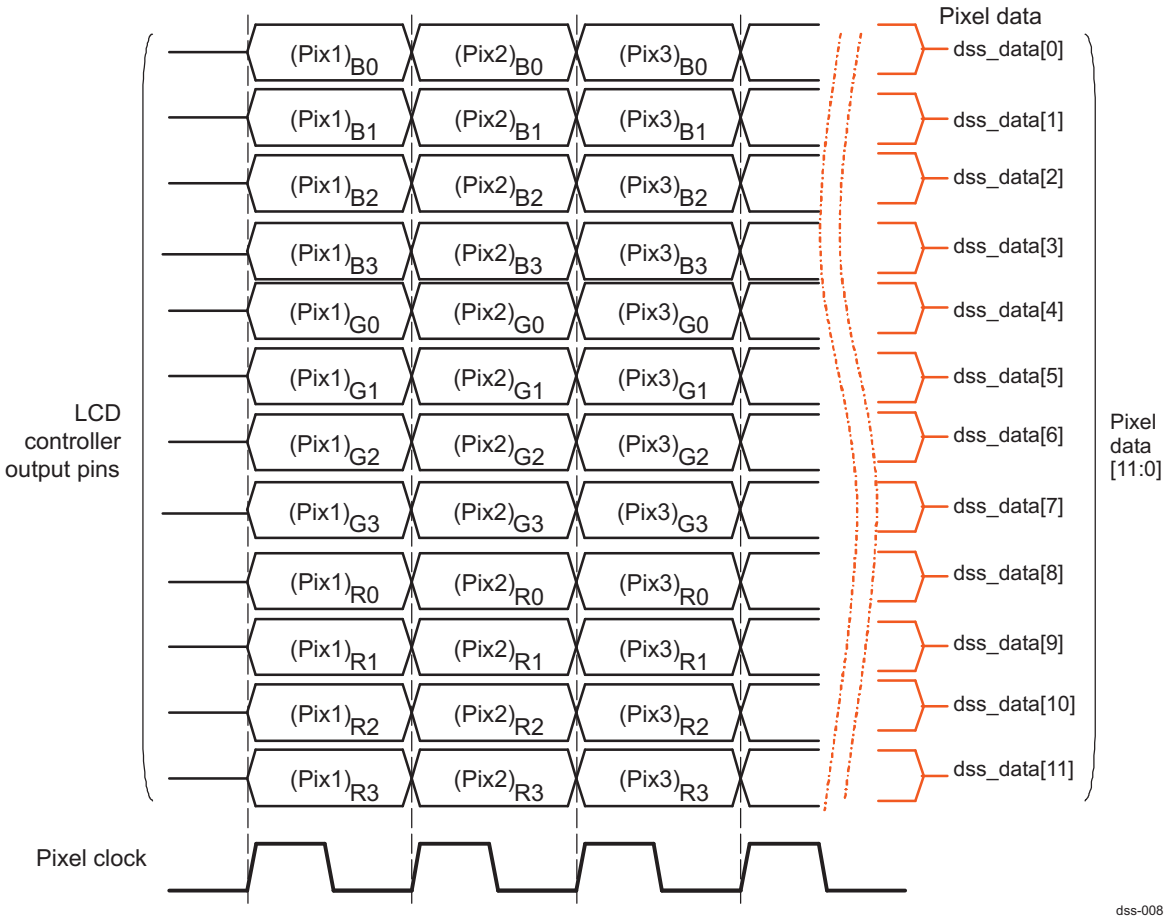


Figure 12-9. LCD Pixel Data Color16 Active Matrix

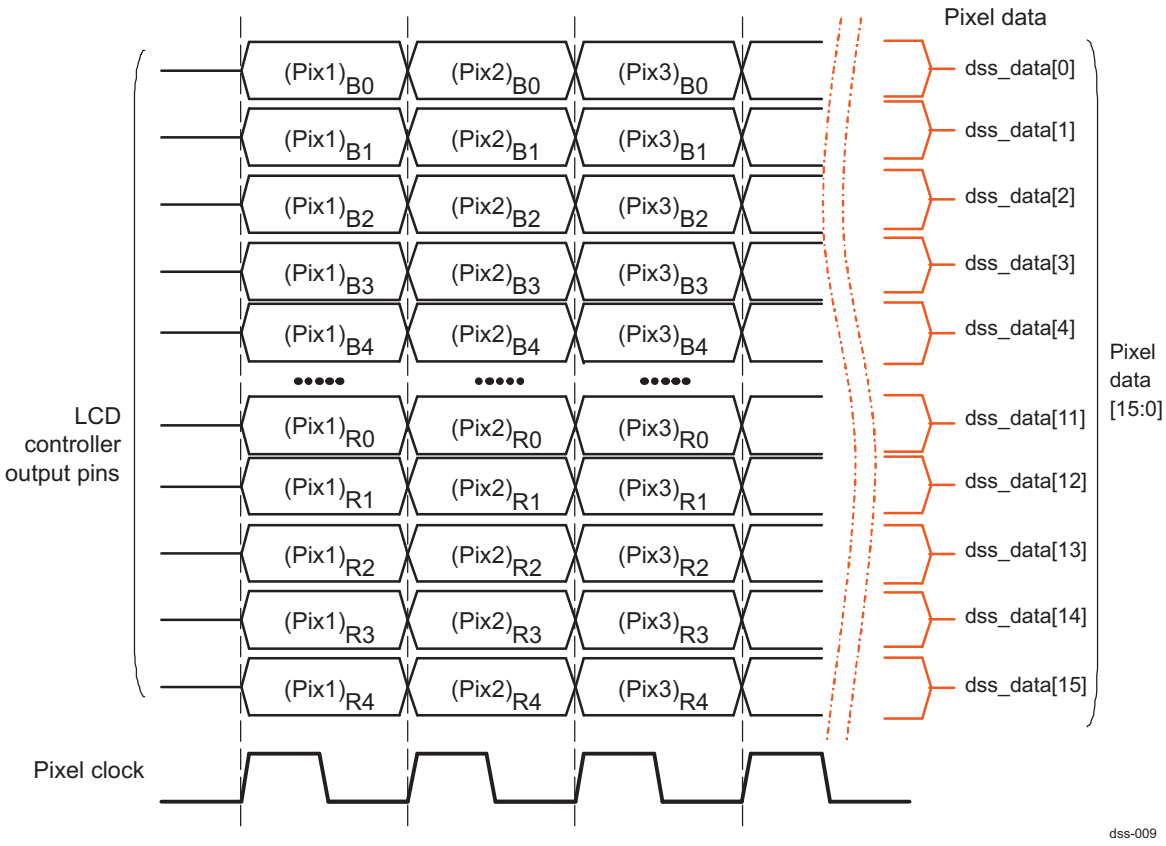


Figure 12-10. LCD Pixel Data Color18 Active Matrix

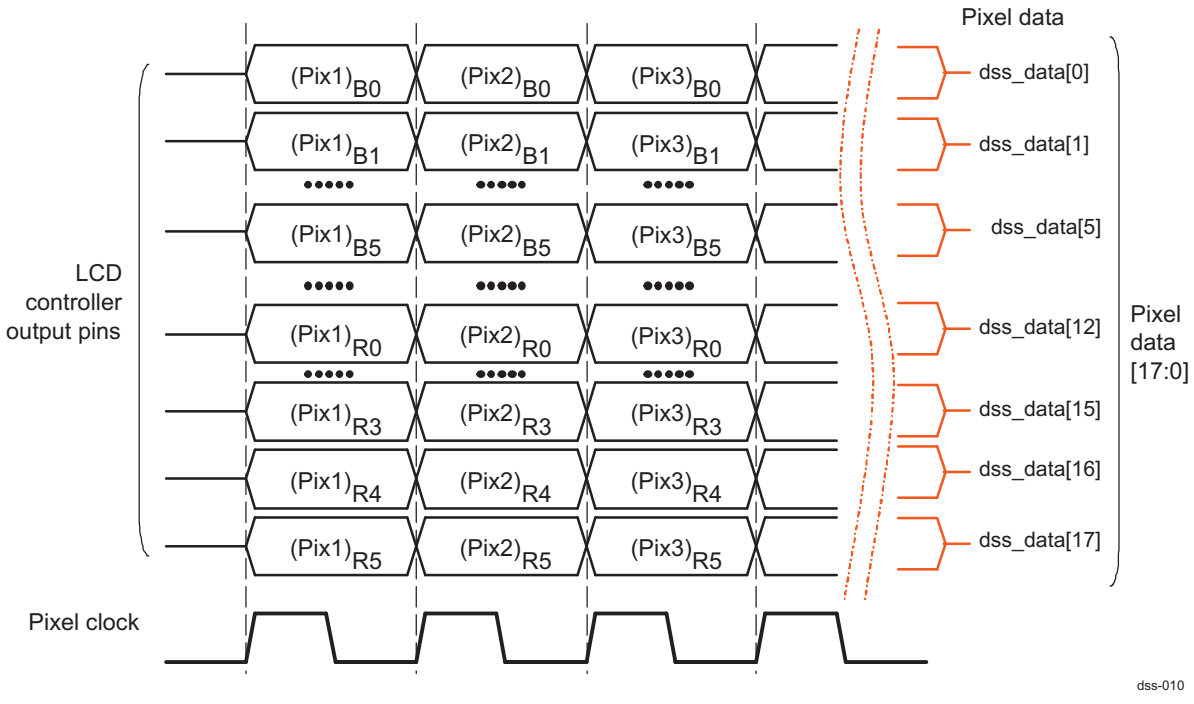
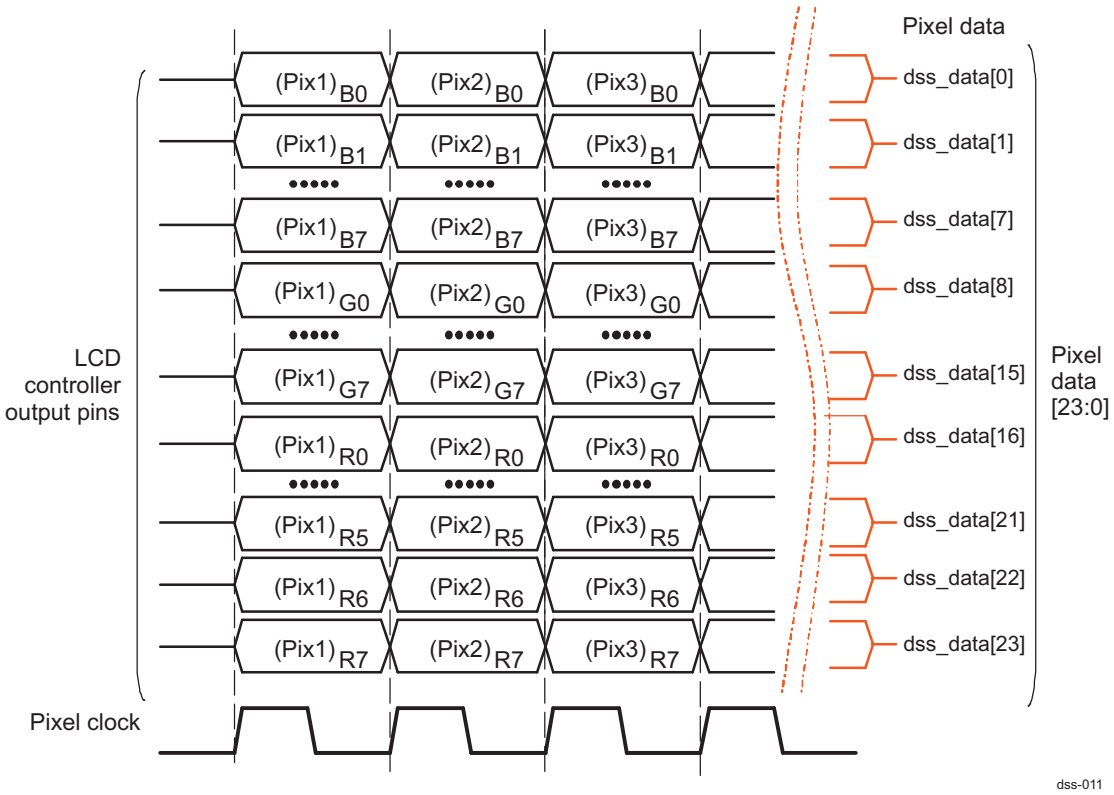


Figure 12-11. LCD Pixel Data Color24 Active Matrix

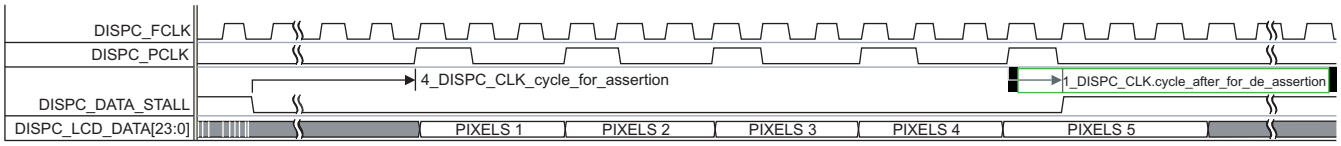


12.2.1.1.4 Transaction Timing Diagrams

- Timing diagrams in flow control mode
  - Stall signal

    The stall signal is used in RFBI and DSI modes. In the case of RFBI mode, it is used to indicate when the display controller must stop sending data over the LCD output interface. The RFBI module asserts the stall signal to stop data output by the display controller. It is deasserted to indicate when new data must be outputted by the display controller.

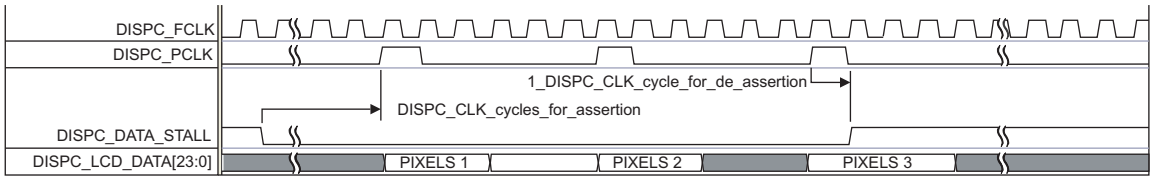
Figure 12-12. RFBI Data Stall Signal Diagram



To avoid underflow of the DMA FIFO, the FIFO handcheck feature can be enabled by setting the DSS.DISPC_CONFIG[16] FIFOHANDCHECK bit to 1. The fullness of the FIFOs associated with the pipelines used for the LCD output is checked when the STALL signal is inactive before providing data to the pipeline. This prevents emptying the FIFO when the RFBI module requests data and there is not enough data in the display controller DMA FIFO. This feature must be enabled only when the STALL mode is used (DSS.DISPC_CONTROL[11] STALLMODE bit set to 1).

When the FIFO handcheck feature is activated, the pixel transfer to the RFBI module during STALL inactivity period can be stopped (no DISPC_PCLK pulse) and restarted when there is enough data in the FIFO. The FIFO handcheck ensures that underflow can not occur for the pipelines associated with the LCD output in RFBI mode. Figure 12-13 details the RFBI data stall with FIFO handcheck mode activated.

Figure 12-13. RFBI Data Stall Signal Diagram With Handcheck



– RFBI timing diagrams

Table 12-5 lists the programmable timing fields. Figure 12-14 through Figure 12-16 show timing diagrams of read/write transactions to the LCD panel for the RFBI mode.

Table 12-5. Programmable Timing Fields in RFBI Mode

Timing Name	Register Field	Description
CSOnTime	DSS.RFBI_ONOFF_TIMEi[3:0] CSONTIME (with I = 0 or 1)	CS assertion time from start access time
CSOffTime	DSS.RFBI_ONOFF_TIMEi[9:4] CSOFFTIME (with I = 0 or 1)	CS deassertion time from start access time
WeCycleTime	DSS.RFBI_CYCLE_TIMEi[5:0] WECYCLETIME (with I = 0 or 1)	The time when A0 becomes valid until write cycle completion
WEOnTime	DSS_RFBI_ONOFF_TIMEi[13:10] WEONTIME (with I = 0 or 1)	WE assertion delay time from start access time
WEOffTime	DSS_RFBI_ONOFF_TIMEi[19:14] WEOFFTIME (with I = 0 or 1)	WE deassertion delay time from start access time
RECycleTime	DSS.RFBI_CYCLE_TIMEi[11:6] RECYCLETIME (with I = 0 or 1)	The time when A0 becomes valid until read cycle completion
REOnTime	DSS_RFBI_ONOFF_TIMEi[23:20] REONTIME (with I = 0 or 1)	RE assertion delay time from start access time
REOffTime	DSS.RFBI_ONOFF_TIMEi[29:24] REOFFTIME (with I = 0 or 1)	RE assertion delay time from start access time
CSPulseWidth	DSS.RFBI_CYCLE_TIMEi[17:12] CSPULSEWIDTH (with I = 0 or 1)	The time when write cycle time or read cycle time completes

Figure 12-14. Command Data Write

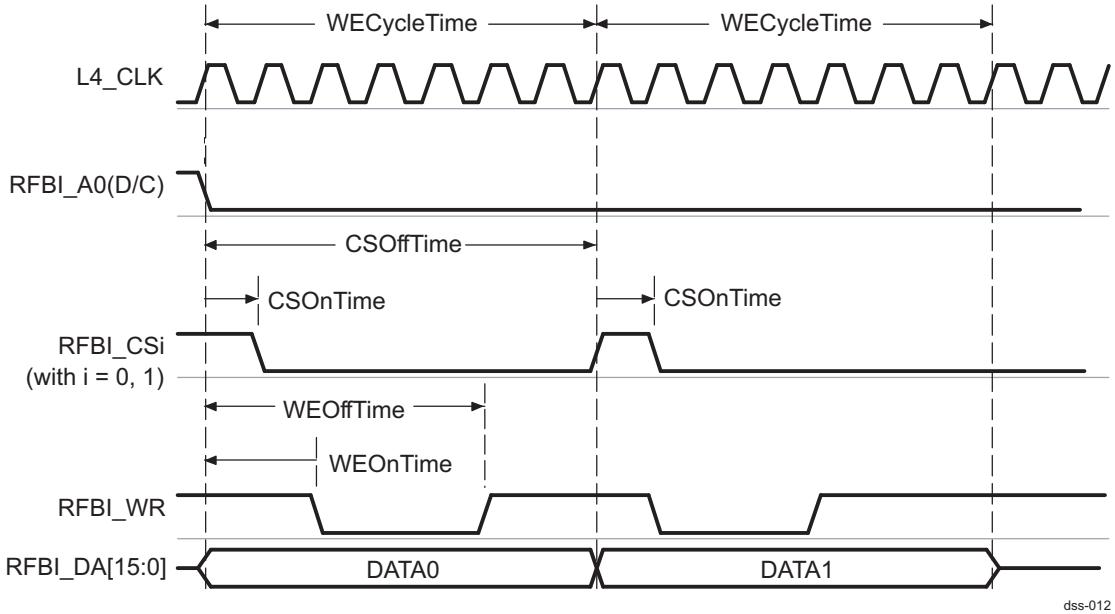


Figure 12-15. Display Data Read

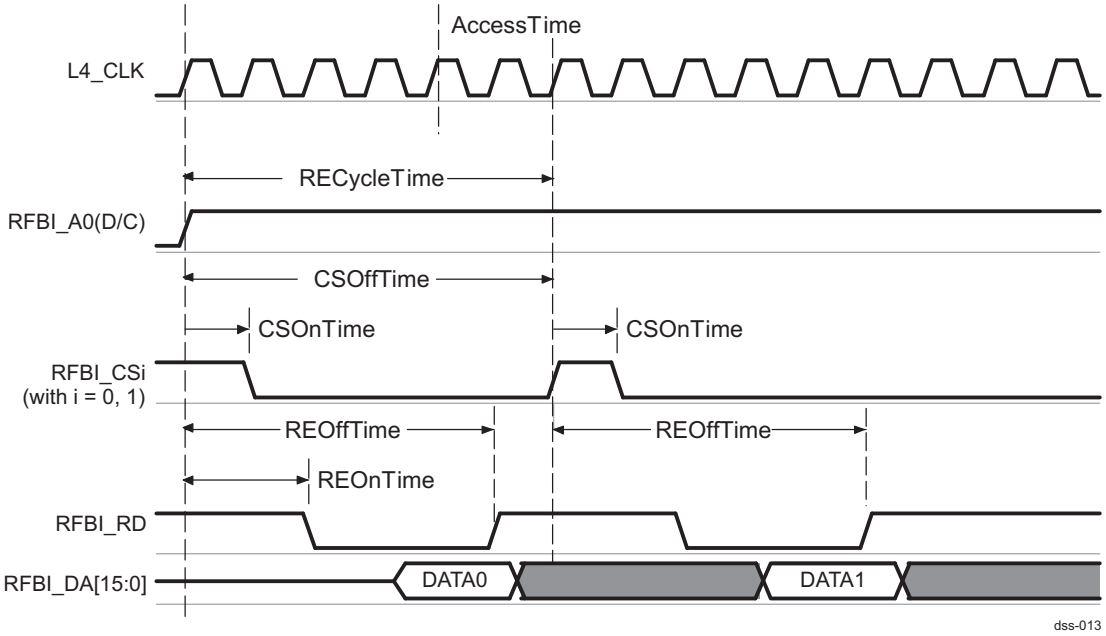
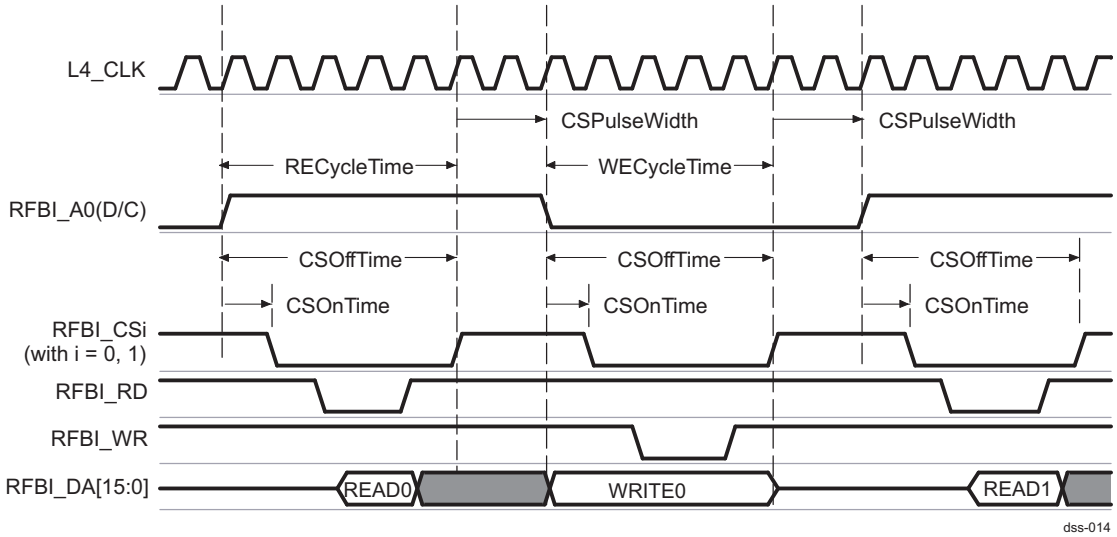


Figure 12-16. Read to Write and Write to Read



- Timing diagrams in bypass mode

  Figure 12-17 through Figure 12-32 show timing diagrams of synchronization signals and pixel clock in bypass mode for both passive matrix and active matrix panels. The display controller directly drives these signals, which are related to the programmable fields described in Table 12-6.

Table 12-6. Programmable Fields in Bypass Mode

Name	Register	Description
PPL	DSS.DISPC_SIZE_LCD[10:0] PPL bit field value + 1	Pixels per line (PPL)
LPP	DSS.DISPC_SIZE_LCD[26:16] LPP bit field value + 1	Lines per panel
HBP	DSS.DISPC_TIMING_H[31:20] HBP bit field value + 1	Horizontal back porch
HFP	DSS.DISPC_TIMING_H[19:8] HFP bit field value + 1	Horizontal front porch
HSW	DSS.DISPC_TIMING_H[7:0] HSW bit field value + 1	Horizontal synchronization pulse width
VBP	DSS.DISPC_TIMING_V[31:20] VBP bit field value	Vertical back porch
VFP	DSS.DISPC_TIMING_V[19:8] VFP bit field value	Vertical front porch
VSW	DSS.DISPC_TIMING_V[7:0] VSW bit field value + 1	Vertical synchronization pulse width
ONOFF	DSS.DISPC_POL_FREQ[17] ONOFF bit	DISPC_HSYNC and DISPC_VSYNC pixel clock control
RF	DSS.DISPC_POL_FREQ[16] RF bit	DISPC_HSYNC and DISPC_VSYNC pixel clock edge control
IEO	DSS.DISPC_POL_FREQ[15] IEO bit	Invert DISPC_ACBIAS
IPC	DSS.DISPC_POL_FREQ[14] IPC bit	Invert DISPC_PCLK
IHS	DSS.DISPC_POL_FREQ[13] IHS bit	Invert DISPC_HSYNC
IVS	DSS.DISPC_POL_FREQ[12] IVS bit	Invert DISPC_VSYNC

- Active matrix timing configuration 1
  - DSS.DISPC_POL_FREQ[17] ONOFF bit = 0
  - DSS.DISPC_POL_FREQ[16] RF bit = 0

    The DISPC_HSYNC and DISPC_VSYNC signals are driven on the opposite edge of DISPC_PCLK from the pixel data.
  - DSS.DISPC_POL_FREQ[15] IEO = 0

    The DISPC_ACBIAS signal is active high.
  - DSS.DISPC_POL_FREQ[14] IPC = 0

    The pixel data are driven on the rising edge of DISPC_PCLK.

– DSS.DISPC_POL_FREQ[13] IHS = 0
The DISPC_HSYNC signal is active high.
– DSS.DISPC_POL_FREQ[12] IVS = 0
The DISPC_VSYNC signal is active high.

Figure 12-17. Active Matrix Timing Diagram of Configuration 1 (Start of Frame)

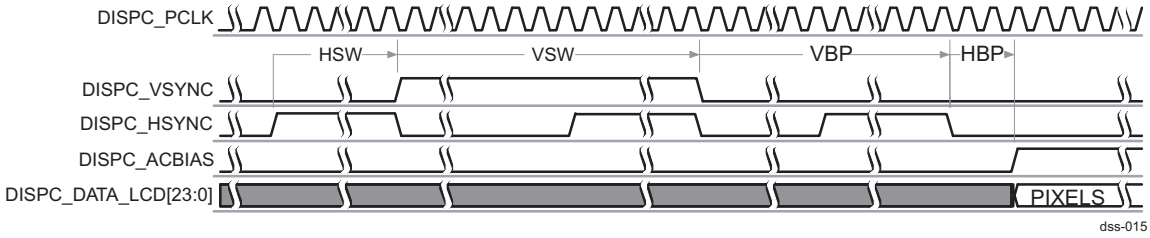


Figure 12-18. Active Matrix Timing Diagram of Configuration 1 (Between Lines)

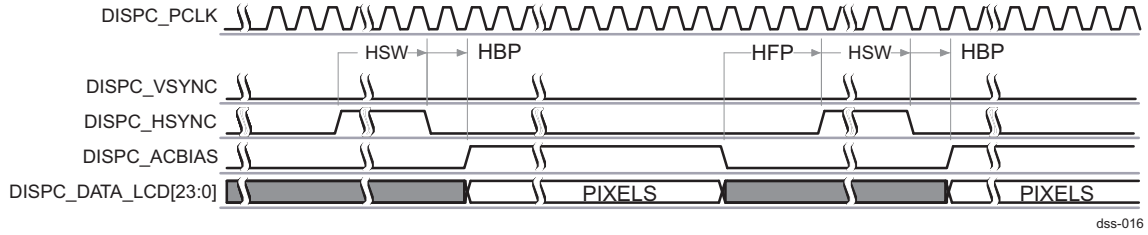


Figure 12-19. Active Matrix Timing Diagram of Configuration 1 (Between Frames)

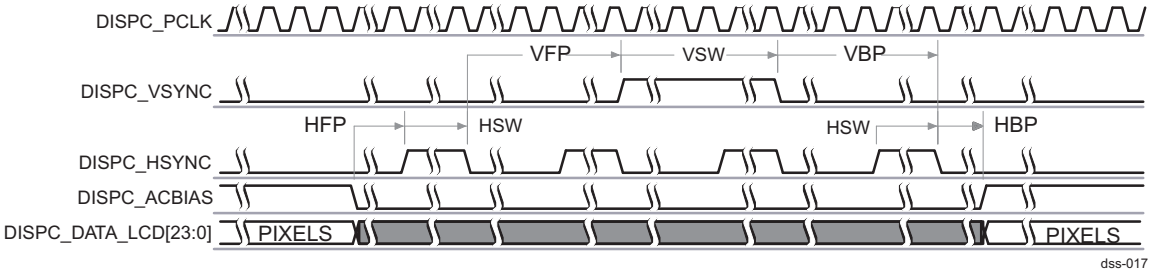
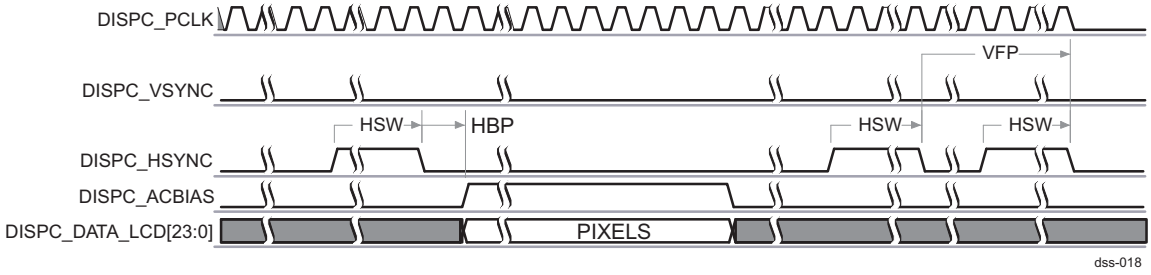


Figure 12-20. Active Matrix Timing Diagram of Configuration 1 (End of Frame)



- Active matrix timing configuration 2
  – DSS.DISPC_POL_FREQ[17] ONOFF bit = 1
  – DSS.DISPC_POL_FREQ[16] RF bit = 1
  The DISPC_HSYNC and DISPC_VSYNC signals are driven on the rising edge of DISPC_PCLK.
  – DSS.DISPC_POL_FREQ[15] IEO = 1
  The DISPC_ACBIAS signal is active low.
  – DSS.DISPC_POL_FREQ[14] IPC = 1
  The pixel data is driven on the falling edge of DISPC_PCLK.

– DSS.DISPC_POL_FREQ[13] IHS = 1
The DISPC_HSYNC signal is active low.
– DSS.DISPC_POL_FREQ[12] IVS = 1
The DISPC_VSYNC signal is active low.

Figure 12-21. Active Matrix Timing Diagram of Configuration 2 (Start of Frame)

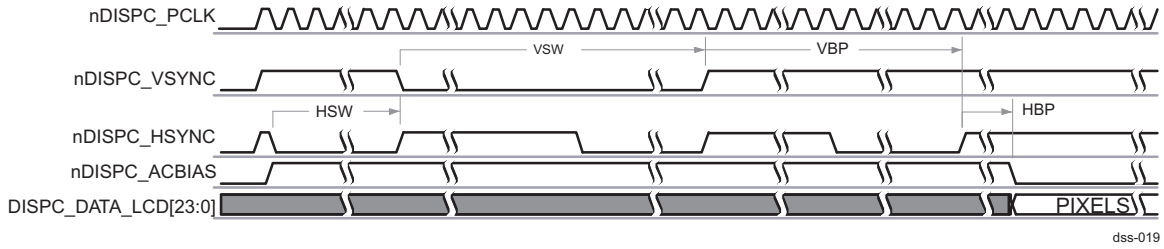


Figure 12-22. Active Matrix Timing Diagram of Configuration 2 (Between Lines)

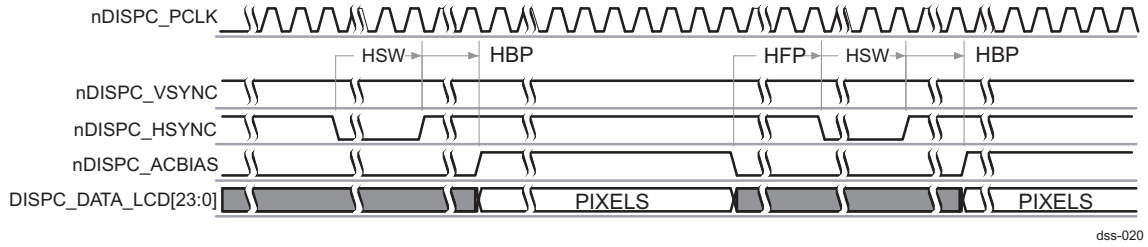


Figure 12-23. Active Matrix Timing Diagram of Configuration 2 (Between Frames)

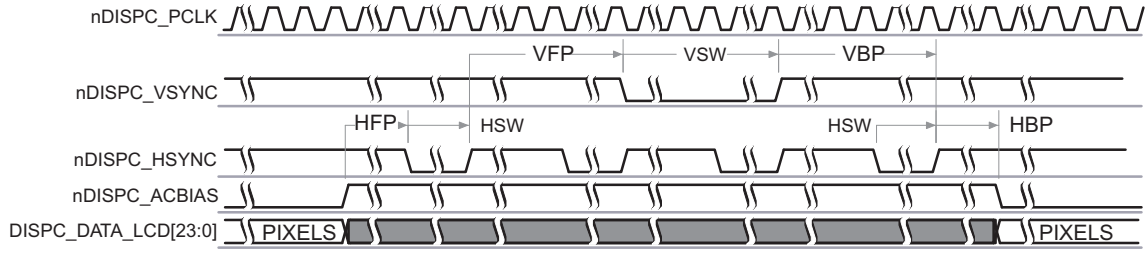
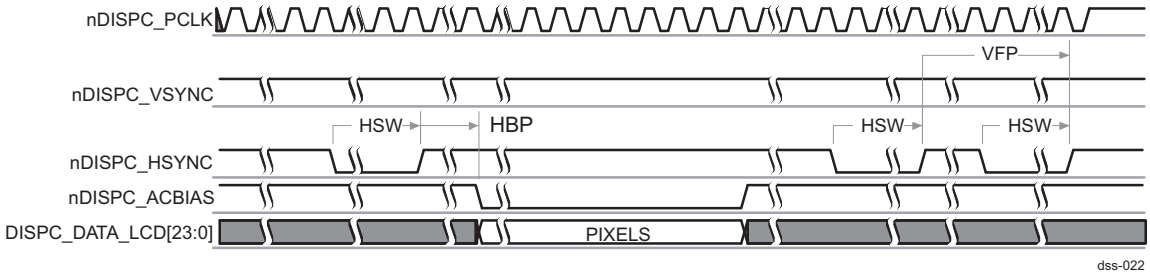


Figure 12-24. Active Matrix Timing Diagram of Configuration 2 (End of Frame)



- Active matrix timing configuration 3
  – DSS.DISPC_POL_FREQ[17] ONOFF bit = 1
  – DSS.DISPC_POL_FREQ[16] RF bit = 1
  The DISPC_HSYNC and DISPC_VSYNC signals are driven on the rising edge of DISPC_PCLK.
  – DSS.DISPC_POL_FREQ[15] IEO = 0
  The DISPC_ACBIAS signal is active high.
  – DSS.DISPC_POL_FREQ[14] IPC = 0

The pixel data are driven on the rising edge of DISPC_PCLK.

– DSS.DISPC_POL_FREQ[13] IHS = 0
  The DISPC_HSYNC signal is active high.
– DSS.DISPC_POL_FREQ[12] IVS = 0
  The DISPC_VSYNC signal is active high.

Figure 12-25. Active Matrix Timing Diagram of Configuration 3 (Start of Frame)

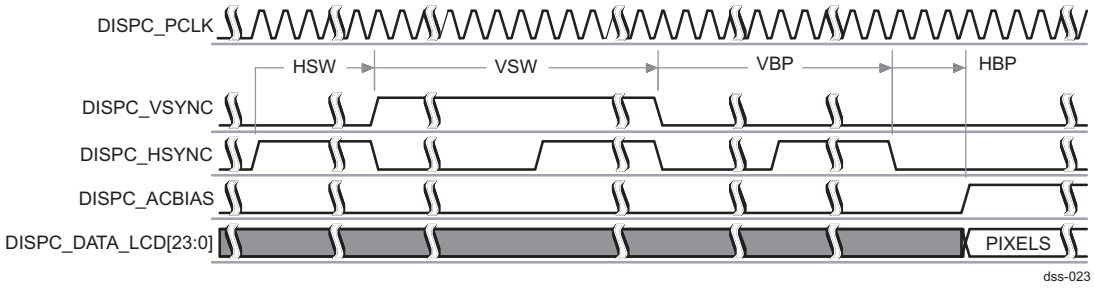


Figure 12-26. Active Matrix Timing Diagram of Configuration 3 (Between Lines)

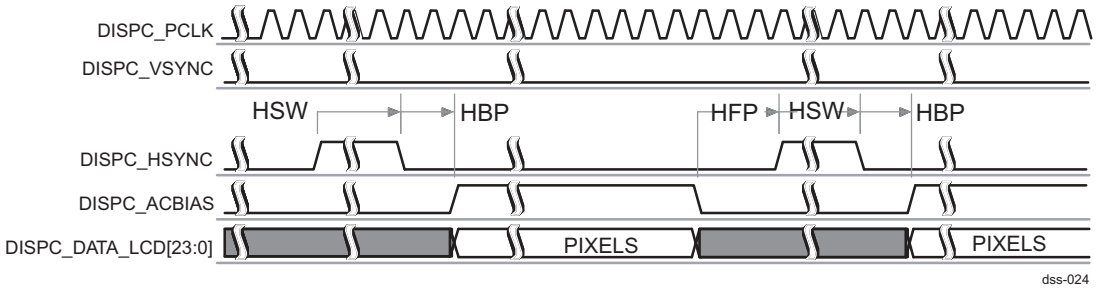


Figure 12-27. Active Matrix Timing Diagram of Configuration 3 (Between Frames)

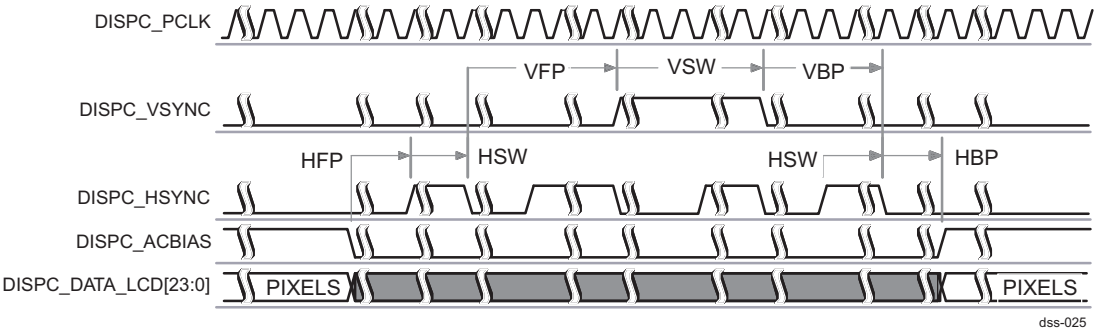
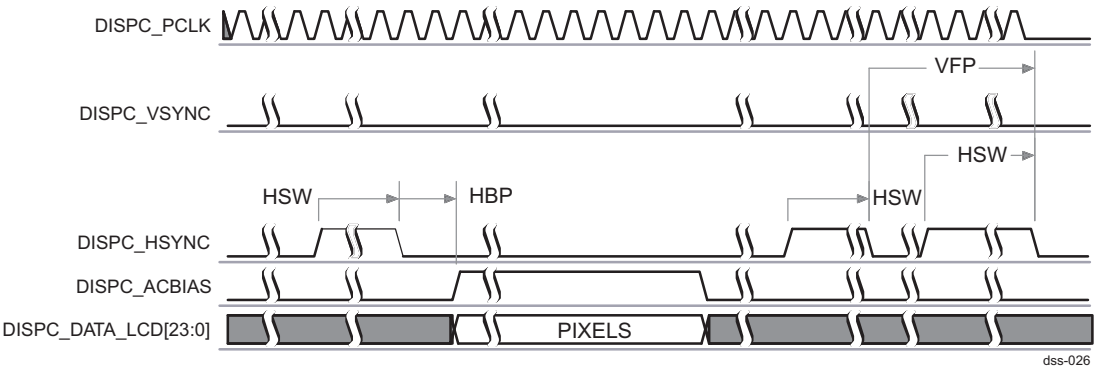


Figure 12-28. Active Matrix Timing Diagram of Configuration 3 (End of Frame)



- Passive matrix timing configuration
  - DSS.DISPC_POL_FREQ[17] ONOFF bit = 0
  - DSS.DISPC_POL_FREQ[16] RF bit = 0

    The DISPC_HSYNC and DISPC_VSYNC signals are driven on the opposite edge of DISPC_PCLK from the pixel data.
  - DSS.DISPC_POL_FREQ[15] IEO = 0

    The DISPC_ACBIAS signal is active high.
  - DSS.DISPC_POL_FREQ[14] IPC = 0

    The pixel data are driven on the rising edge of DISPC_PCLK.
  - DSS.DISPC_POL_FREQ[13] IHS = 0

    The DISPC_HSYNC signal is active high.
  - DSS.DISPC_POL_FREQ[12] IVS = 0

    The DISPC_VSYNC signal is active high.

Figure 12-29. Passive Matrix Timing Diagram (Start of Frame)

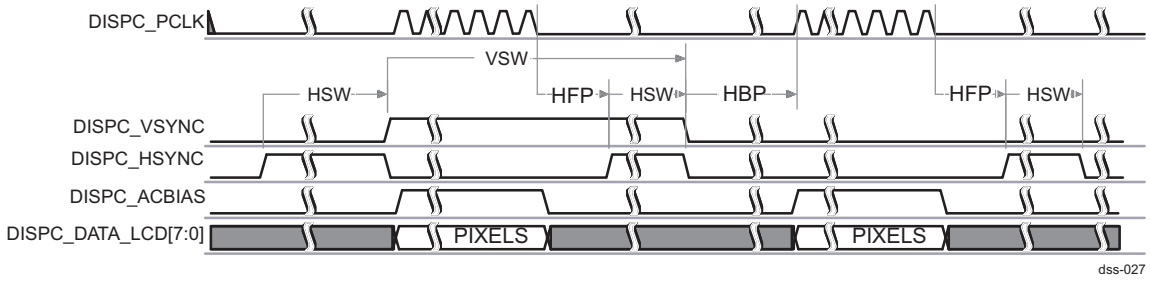


Figure 12-30. Passive Matrix Timing Diagram (Between Lines)

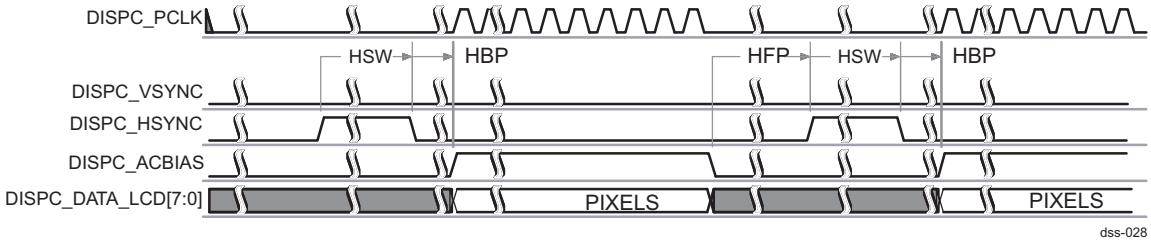


Figure 12-31. Passive Matrix Timing Diagram (Between Frames)

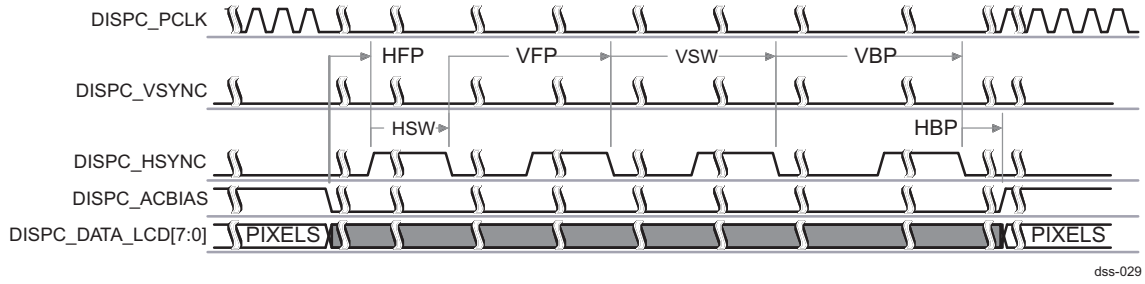
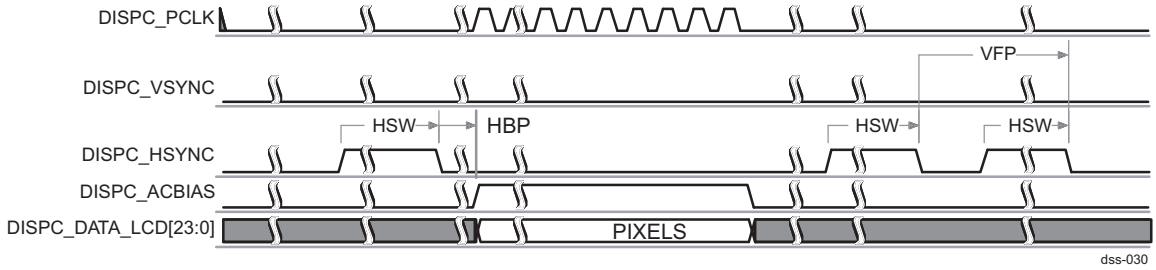


Figure 12-32. Passive Matrix Timing Diagram (End of Frame)



12.2.1.2 SDI Serial Interface

In the serial interface, the modules in the display subsystem path are the display controller and the SDI with its associated PLL. The display controller provides the required control signals to interface the memory frame buffer (either SDRAM or SRAM) directly to the external displays. The display controller is connected to the memory through the L3 interconnect and has its own DMA to read the data from the system memory.

The SDI module is an additional display output mode that facilitates the connection of displays over relatively few conductors.

The SDI module implements the TI FlatLink3G protocol and converts the conventional parallel display signals (including pixel data) from the display controller to serial form. Texas Instruments provides a global solution with OMAP device associated to a programmable 27-bit serial display interface receiver - SN65LVDS302 receiver for 3-data pair, SN65LVDS304 receiver for 2-data pair, and SN65LVDS306 receiver for 1-data pair.

The SDI module contains a PLL to multiply the pixel clock by an appropriate factor. The resulting serialized data is then transmitted on 1 to 3 differential data pairs and an additional clock pair that has a reference transition at the boundary between pixels.

NOTE: Only one 9-bit RFBI driven display can be run at the same time as the SDI.

Figure 12-33 shows a typical connection between the SDI module and a compliant panel display.

Figure 12-33. Typical SDI Connection

CAUTION

When routing the PCB, ensure that the connections between the OMAP device and the LCD panel meet FlatLink3G specifications, which are available in the 27-bit serial display interface receiver data sheet.

NOTE:

- The SDI pins are multiplexed in mode 1 with some LCD parallel pins. See Chapter 6, *System Control Module*, for more details on pad multiplexing.
- The connection from the vss_sdi pin of the device to the PCB digital ground must be as short as possible.
- The vdds_sdi dedicated power supply is provided by a power IC. Texas Instruments provides a global solution with the OMAP device associated with a power IC.

Table 12-7 describes the interface signals between the SDI module of the OMAP device and the LCD panel.

Table 12-7. Interface Signals Between the SDI Module and LCD Panel

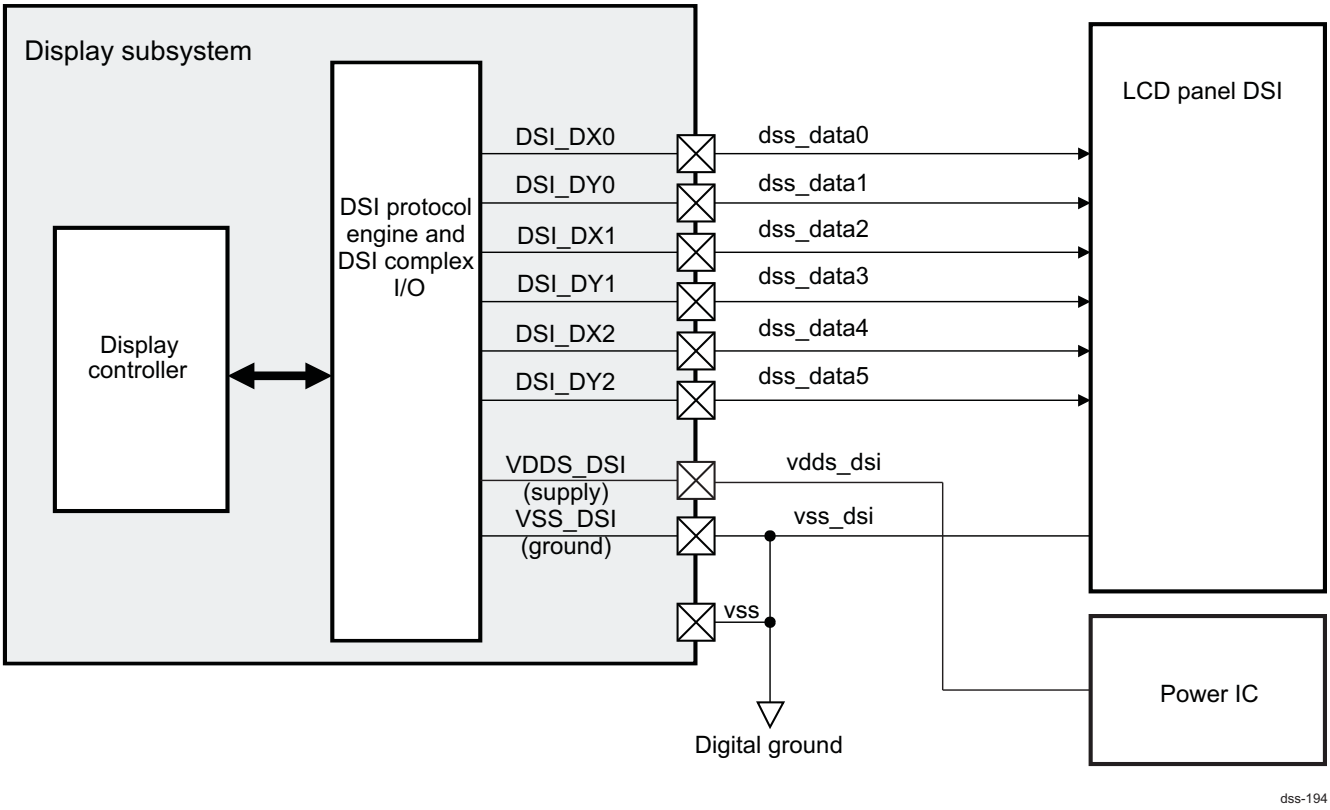
Signal Name	Type[1]	Description
SDI_DATA1N	O	First pair of differential data signals
SDI_DATA1P		
SDI_DATA2N	O	Second pair of differential data signals
SDI_DATA2P		
SDI_DATA3N	O	Third pair of differential data signals
SDI_DATA3P		
SDI_CLKN	O	Pair of differential clock signals
SDI_CLKP		
VDDS_SDI	PWR	Dedicated vdds power supply for SDI module
VSS_SDI	PWR	Dedicated vss ground for SDI module

(1) O=Output PWR=Power

12.2.1.3 DSI Serial Interface

Figure 12-34 shows a typical connection between the DSI modules and a compliant panel display.

Figure 12-34. Typical DSI Connection



NOTE: The DSI pins are multiplexed in mode 1 with some LCD parallel pins. See Chapter 6, *System Control Module*, for more details on pad multiplexing.

12.2.2 LCD Support With MIPI DSI 1.0 Protocol and Data Format

This section summarizes the MIPI DSI1.0 protocol and data format.

12.2.2.1 Physical Layer

Table 12-8 lists the DSI interface I/O.

Table 12-8. I/O Description for DSI Serial Interface

Signal Name	I/O[1]	Description	Value at Reset	
dsi_dx0	line 1	I/O	Serial data/clock input/output	N/A
dsi_dy0				
dsi_dx1	line 2	I/O	Serial data/clock input/output	N/A
dsi_dy1				
dsi_dx2	line 3	I/O	Serial data/clock input/output	N/A
dsi_dy2				

(1) I/O = Input/Output

NOTE: Each serial line (line 1, line 2, and line 3) can be used as clock lane or data lane. All polarities are supported. The MIPI DSI 1.0 protocol requires at least one clock line and one data lane.

Lanes support four operating modes:

- HS mode: High-speed transmit mode
- LP mode: Low-power transmit mode (also called low-power state [LPS])
- ULPS: Ultra-low power state used between two transmissions
- Off mode: Lane is off.

12.2.2.1.1 Data/Clock Configuration

From the device point of view, the DSI interface consists of six input/output signals representing three differential signals: The serial clock and one or two serial data. The minimum configuration is one data pair and one clock pair.

- The data signal carries the bit-serial data. The DSI transmitter in the host sends the data in-quadrature with the DDR clock in high speed mode; otherwise, the clock is extracted from the received data in low-speed mode. The data is transmitted byte-wise least significant bit (LSB) first.
- The clock signal carries the DDR clock signal in high speed transmission.
- Software users must configure the order of the data lanes to indicate the byte order while splitting the byte stream for each DSI_PHY into bytes.

Table 12-9 details all the DSI lanes configuration.

Table 12-9. DSI Lane Configuration

DSI DSI_PHY Lane Configuration	Data/Clock Lane Position			Description
	1	2	3	
Mode CLK + DATA1	CLK	DATA1	Not used	Single data lane
	CLK	Not used	DATA1	
	DATA1	CLK	Not used	
	Not used	CLK	DATA1	
	DATA1	Not used	CLK	
	Not used	DATA1	CLK	
Mode CLK + DATA1 + DATA2	CLK	DATA1	DATA2	Two data lanes
	CLK	DATA2	DATA1	
	DATA1	CLK	DATA2	
	DATA2	CLK	DATA1	
	DATA1	DATA2	CLK	

Table 12-9. DSI Lane Configuration (continued)

DSI DSI_PHY Lane Configuration	Data/Clock Lane Position			Description
	1	2	3	
	DATA2	DATA1	CLK	

NOTE:

- The byte on Dn is sent before the byte on Dn+1, all the combinations of data and clock are supported through programming of the DSS.DSI_COMPLEXIO_CFG1 register. The CLOCK_POSITION and CLOCK_POL bit fields configure which lane transmits the clock and define its polarity. Four bit fields (DATA1_POSITION, DATA1_POL, DATA2_POSITION, and DATA2_POL) configure the data lanes and their polarity. The DATA2_POSITION bit field can be set to 0; in this case, only the data lane defined in the DATA1_POSITION bit field is used, and data is transmitted on only one clock lane and one data lane.
- The configuration of the DSI complex I/O (number of data lanes, position, differential order) should not be changed while DSS.DSI_CLK_CTRL[20] LP_CLK_ENABLE bit is set to 1. For the hardware to recognize a new configuration of the complex I/O (done in DSS.DSI_COMPLEXIO_CFG1 register), it is recommended to follow this sequence: First set the DSS.DSI_CTRL[0] IF_EN bit to 1, next reset the DSS.DSI_CTRL[0] IF_EN to 0, then set DSS.DSI_CLK_CTRL[20] LP_CLK_ENABLE to 1, and finally, set again the DSS.DSI_CTRL[0] IF_EN bit to 1. If the sequence is not followed, the DSI complex I/O configuration is undetermined.
- Only DATA1 is bidirectional in command mode. The low-power received information is always sent by the display panel using DATA1. Since any lane of the DSI complex I/O can be configured as data lane DATA1, all lanes of the complex I/O are bidirectional

12.2.2.1.2 ULPS

Each lane can be put in ultra-low power state (ULPS) by software configuration. The ULPS mode requires all the following conditions:

- The lane must be in stop state.
- For data lanes, no data must be pending in the DSI module.
- For data lane 1, no BTA should have been sent. The DSI module should have control of the bus.

The control of each lane is independently controlled by the DSS.DSI_COMPLEXIO_CFG2 register.

12.2.2.2 Video Port (VP) Interface

NOTE: The signals described in this section are internal and not bounded outside the device. This section aims at helping software users understand the internal connections between the display controller (DISPC) and the DSI protocol engine.

Table 12-10 summarizes the video interface signals. This interface is used to connect the display controller to the DSI protocol engine to send real time data streams. Note that only the active matrix timings are supported by DSI protocol engine. The HSYNC/VSYNC/DE/DATA signals are driven on the rising or falling edge of the pixel clock (VP_PCLK).

Table 12-10. Video Interface for DSI Protocol Engine

Signal Name	Type[(1)]	Description
VP_HSYNC	I	Horizontal sync signal
VP_VSYNC	I	Vertical sync signal
VP_DATA[23:0]	I	Parallel output data: Bits 0 to 23

(1) I = Input, O = Output

Table 12-10. Video Interface for DSI Protocol Engine (continued)

Signal Name	Type[1]	Description
VP_PCLK	I	Pixel clock. In case of STALL configuration, it is used to indicate when new data is on the data bus during the clock period of VP_CLK. The VP_PCLK is generated from VP_CLK through division. The clock ratio is defined in the DSS.DSI_CTRL[4] VP_CLK_RATIO bit and must be aligned with the configuration of the clock divisor in the display controller (DSS.DISPC_DIVISOR[7:0] PCD bit field).
VP_DE	I	Data enable
VP_STALL	O	The stall signal must be deasserted to receive pixel and asserted to stop receiving pixel. (It can be used only while the display controller is configured in STALL mode; in that mode, HSYNC and VSYNC are not generated).
VP_CLK	I	Display controller internal clock. It is a free-running clock.

NOTE:

- The polarities of VP_HSYNC and VP_VSYNC are programmable by setting the DSS.DSI_CTRL register.
- The maximum frequency for VP_CLK is 173 MHz at nominal voltage, and 96 MHz at low voltage.
- Clocks VP_CLK and VP_PCLK can have the same frequency.
- The number of bits to be captured on the video port is defined in the DSS.DSI_CTRL[7:6] VP_DATA_BUS_WIDTH bit field.
- VP_DE is connected to the dss_acbias signal in the display controller, and its polarity can be controlled by setting the DISPC_POL_FREQ[15] IEO bit.

The data received on the video port can be stored into the line buffer memories or sent directly on the DSI interface in two cases:

- The line buffer size is too small compared to the line from the display controller.
- There is no line buffer instantiated. If there is no line buffer, the burst mode, defined as frequency burst mode, can not be used. Only the transparency burst mode is supported.

NOTE: The DSS.DSI_CTRL[13:12] LINE_BUFFER bit field defines the number of lines to be used for transferring data from the video port to the DSI link.

12.2.2.2.1 Video Port Used for Video Mode

If the video port is used for video mode, the VP_STALL is not used. Table 12-11 lists the active signals on the video port.

Table 12-11. Video Interface in the Context of Video Mode

Signal Name	Type[1]	Description
VP_HSYNC	I	Horizontal sync signal
VP_VSYNC	I	Vertical sync signal
VP_DATA[23:0]	I	Parallel output data: bits 0 to 23
VP_PCLK	I	Pixel clock.
VP_DE	I	Data enable
VP_CLK	I	It is a free running clock used as the display controller functional clock. The maximum frequency is 173 MHz at nominal voltage, and 96 MHz at low voltage.

(1) I = Input, O = Output

Three video modes are available:

- No line buffer: The data received on the video port are directly output of the DSI port without buffering. The ration of VP_CLK and the DSI high-speed (HS) clock period must ensure the same throughput on the two ports (the two clocks must be generated using the same PLL; the subsystem must provide such configuration).
- One line buffer:
  - The data are first stored in the line buffer; once all the data for one line are received, the DSI protocol engine sends the whole line. Software must adjust timings to allow for storing all line data into the line buffer before sending it to DSI outputs. The synchronization packets are never stored into the line buffer.
- Two line buffers:
  - One line is stored when the second line is output on the DSI port. It allows burst capability. While receiving the first line of the frame, there is no RGB packets sent because the line buffers are empty. To send the last line of the frame, a dummy line must be provided by the display controller to flush the line buffers. The synchronization packets are never stored into the line buffer.

NOTE: If more active lines are received on the video port than the number defined in the DSS.DSI_VM_TIMING3[15:0] VACT bit field, the extra lines are discarded by the DSI protocol engine. These lines are treated as blanking lines.

Figure 12-35 through Figure 12-37 show these three video modes.

NOTE:

- When HSync start and Hsync end short packets are not generated (HSA does not exist), HBP signal must be different than 0.
- The software must ensure that VBP is always defined so that there is at least one HSYNC during VBP.
- In blanking low-power mode (BL-LP), two options are possible:
  - The lane remains in ULPS, and the DSI_CTRL[20] BLANKING_MODE bit is set to 0x0.
  - Dummy bytes are sent during LP with the DSI_CTRL[20] BLANKING_MODE bit set to 0x1; the number of sent bytes is determined by the DSI_VM_TIMING6[15:0] BL_LP_INTERLEAVING bit field.

Figure 12-35. DSI Video Mode Without Burst (No-Line Buffer)

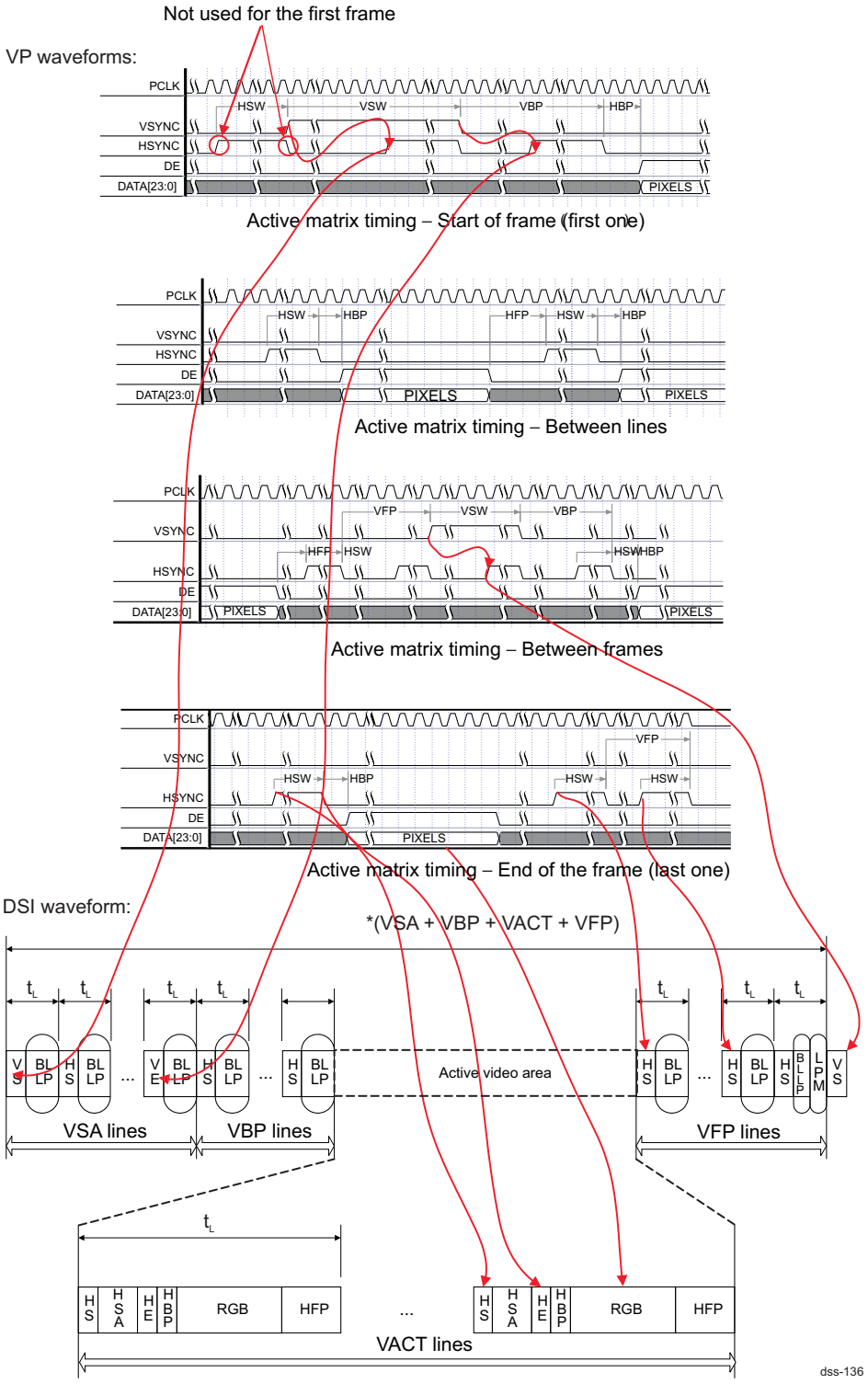


Figure 12-36. DSI Video Mode Without Burst (One-Line Buffer)

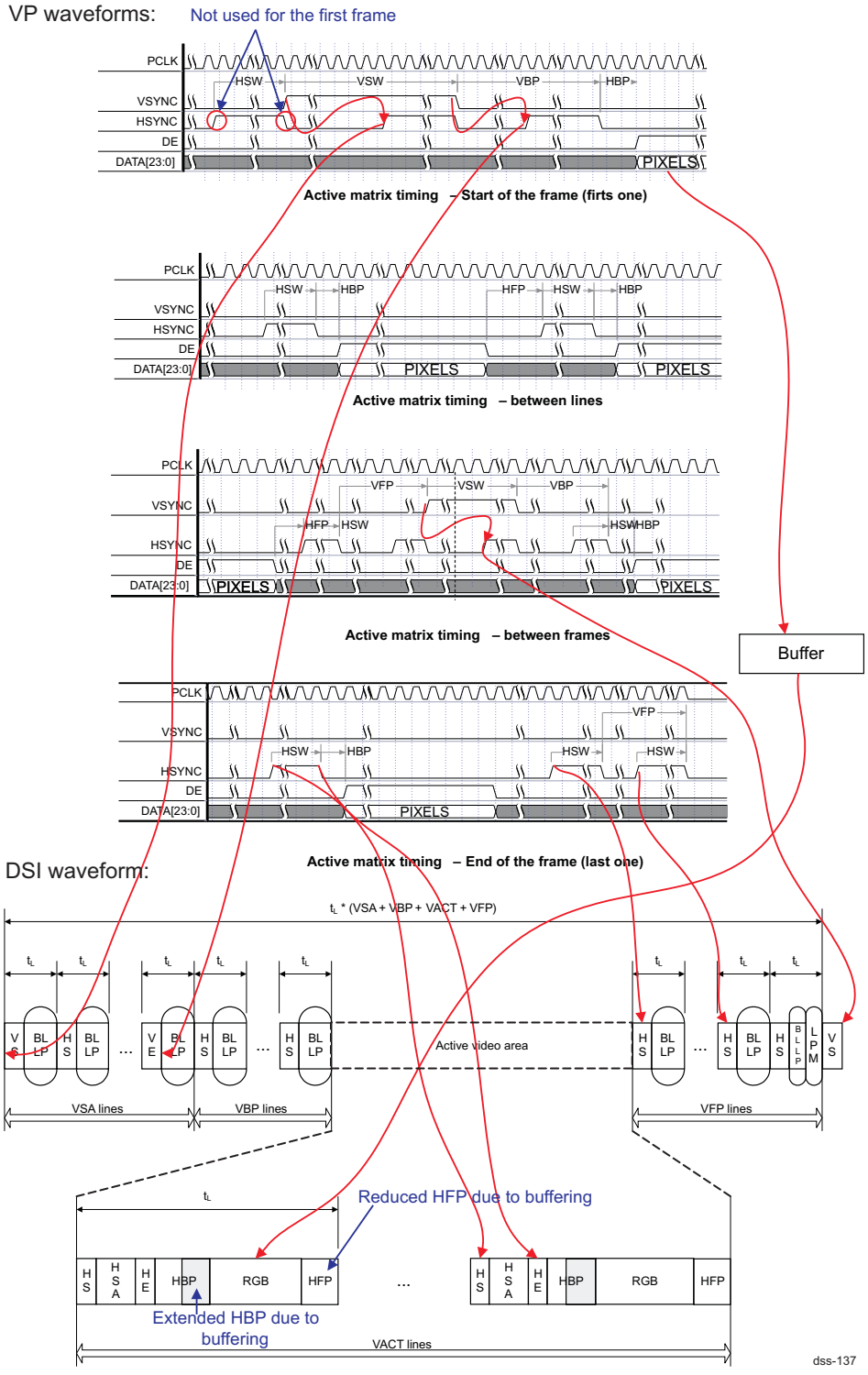
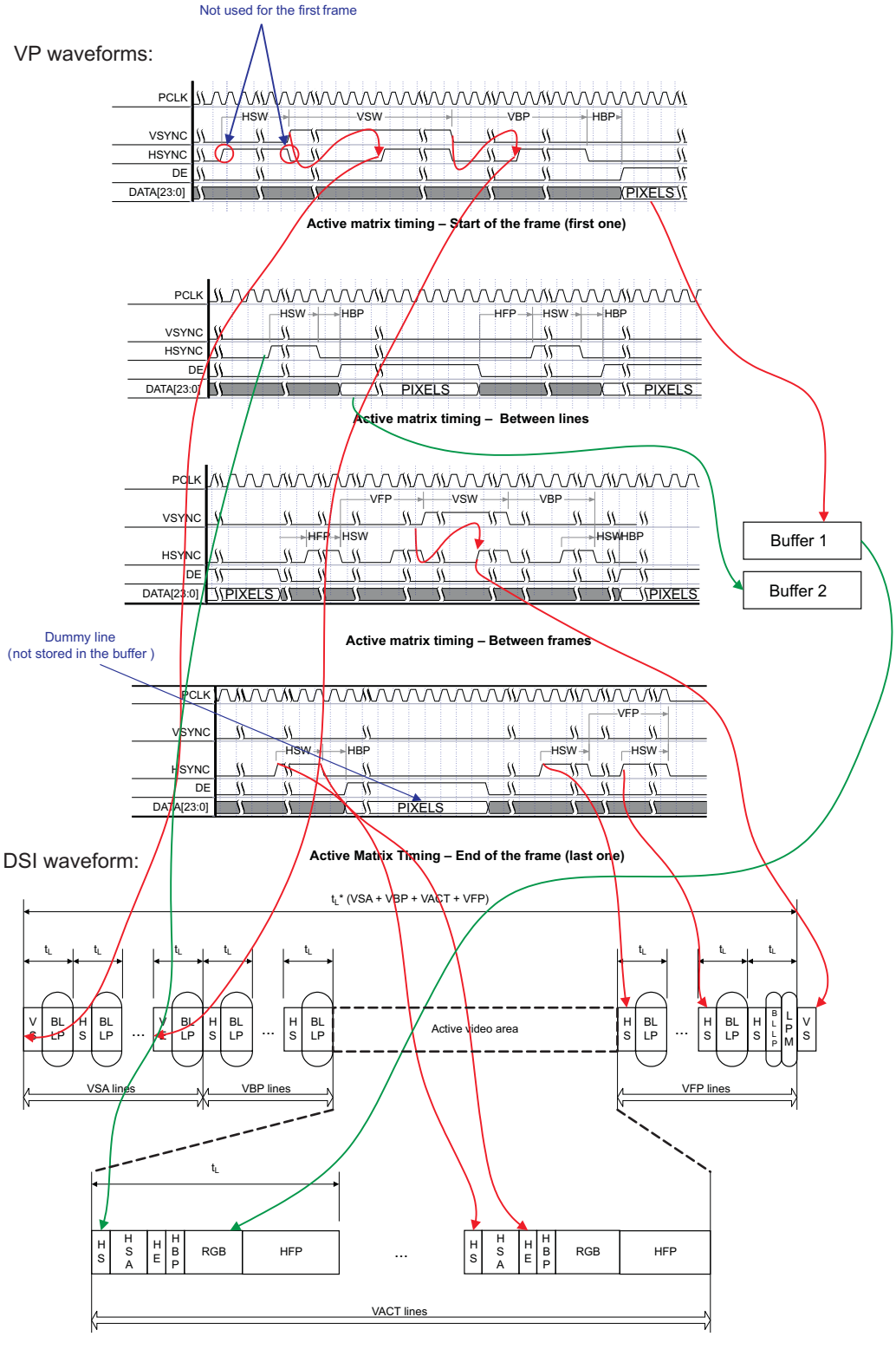


Figure 12-37. DSI Video Mode With Burst (Two-Line Buffers)



dss-138

If the signal VP_DE is not asserted during enough VP_PCLK cycles to be able to capture the number of bytes defined in the word count of the header, the module must send the valid data received on the video port followed by bytes of 0s to match the required number of bytes to transmit. The VP_PCLK must be present during all extra cycles where the DSI protocol engine is expecting pixels.

If the VP_DE signal is asserted for too many VP_PCLK cycles, the module should stop capturing the data on the video port while the number of bytes to capture, as defined in the word count field of the header, is reached.

The HS must check that the received synchronization events on the video port (VSYNC and HSYNC) are within the synchronization window based on expected timings. If the timings (internal and received) are out of sync, the interrupt for out-of-sync should be generated and the interface must be disabled (DSS.DSI_CTRL[0] IF_EN bit is automatically reset by hardware). The unsynchronization window is defined by the DSS.DSI_VM_TIMING2[27:24] WINDOW_SYNC bit field.

12.2.2.2.2 Video Port Used on Command Mode

If the video port is used for command mode, the VP_HSYNC, VP_VSYNC, and VP_DE signals are not used. Table 12-12 describes the active signals on the video port.

Table 12-12. Video Interface in the Context of Command Mode

Signal Name	Type[1]	Description
VP_DATA[23:0]	I	Parallel output data: bits 0 to 23
VP_PCLK	I	One pulse is generated every time new data is output on the data bus
VP_STALL	O	The stall signal must be deasserted to receive pixel and asserted to stop receiving pixel. (It can be used only while the display controller is configured in STALL mode; in that mode, HSYNC and VSYNC are not generated).
VP_CLK	I	Display controller internal clock: It is a free running clock used as the display controller functional clock. The maximum frequency is 173 MHz at nominal voltage and 96 MHz at low voltage.

(1) I = Input, O = Output

NOTE: The stall signal must be deasserted to receive pixels and asserted to stop receiving pixels.

Figure 12-38 and Figure 12-39 show the VP_STALL signal assertion and deassertion on rising and falling edges, respectively.

NOTE: In DSI command mode, the display controller must be configured in stall mode by setting the DSS.DISPC_CONTROL[11] STALLMODE bit to 1.

Figure 12-38. Stall Timing With Pixel on Rising Edge

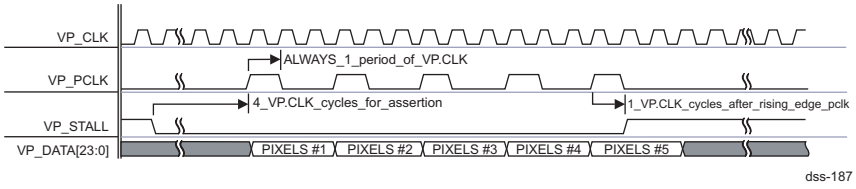
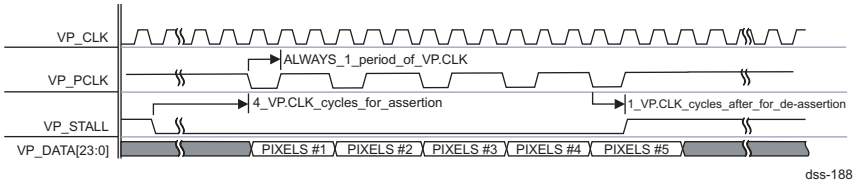


Figure 12-39. Stall Timing With Pixel on Falling Edge



To stop the transfer, the VP_STALL signal must be asserted when the last data is output. The data can be output on the rising or falling edge of the VP_PCLK through registers in the display controller module. The case with data output on falling edge of VP_PCLK is supported by the DSI protocol engine.

The VP_PCLK clock is generated from VP_CLK; these two clocks are balanced. Assertion and deassertion of VP_PCLK is done on the rising edge of VP_CLK. The width of the VP_PCLK pulse depends on the configuration of the clock divisor in the display controller (DSS.DISPC_DIVISOR[7:0] PCD bit field). In the DSI protocol engine, the information is defined in the DSS.DSI_CTRL[4] VP_CLK_RATIO bit and should be aligned with the display controller configuration.

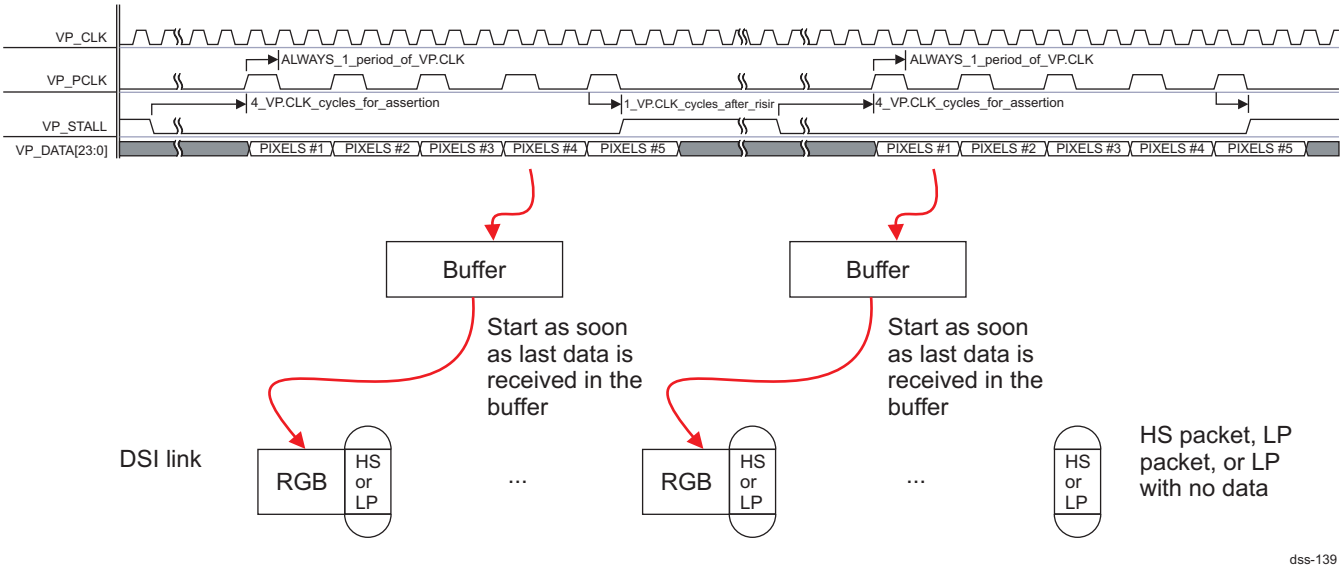
Deassertion of the VP_STALL signal should occur at least 4 VP_CLK cycles before assertion of VP_PCLK. Assertion of VP_STALL should occur one cycle VP_CLK after deassertion of VP_PCLK for the last pixel to be transferred. The VP_CLK clock is generated by the display controller under software control. It can be kept running between assertion and deassertion of VP_STALL.

The word count (WC) defined in the DSS.DSI_VCn_LONG_PACKET_HEADER register for the virtual channel (VC)associated with the video port, indicates the number of bytes to receive (one line or two lines can be used, depending on the WC and size of the line buffer). The total size defined in the WC of the header register can not exceed the size of the line buffer multiplied by the number of buffer lines.

The stall assertion/deassertion depends on the number of bytes to be received considering the size of the video port bus defined in the DSS.DSI_CTRL[7:6] VP_DATA_BUS_WIDTH bit field.

Figure 12-40 shows the data flow in command mode using the video port.

Figure 12-40. Data Flow in Command Mode Using the Video Port



Two command modes are available:

- One line buffer: The data are stored in the line buffer before being sent.
- Two line buffers: The two lines are used if the word count defined in the DSS.DSI_VCn_LONG_PACKET_HEADER register is bigger than the line size; otherwise, one line buffer is used.

NOTE: In command mode, the video port can only be used in one or two line buffer configuration. The no-line buffer configuration is not allowed.

The packets can be sent using high-speed or low-speed.

NOTE: The DCS command in the payload can be inserted automatically using the DSI_CTRL[24] DCS_CMD_ENABLE bit. If TE is used, hardware automatically inserts the DCS Write Start command for the first packet of the frame transfer and the DCS Write Continue command for all subsequent packets.

12.2.2.2.3 Burst Mode

When the burst mode is enabled, the video port receives data from the display controller at the pixel clock. The DSI protocol engine buffers the data in its own line FIFO (double-line buffer of 1024 x 24-bit pixels maximum). The read speed of the line can be twice the pixel clock to increase the blanking time of the video mode and to allow command mode traffic to be interleaved during the blanking period. The burst mode uses a dual-line buffer.

The DSI port can output data from one line buffer while the second one is accessed by the video port. The two processes are concurrent but they do not access the same line at the same time. The DSI transfer can start only when the whole video port line is transferred into a line buffer. The switch is controlled by the VP_HS signal on the video port side and by internal signal on the DSI port indicating that the last data for the current line has been written into the line buffer.

NOTE: The line buffers are used to store the pixels only. The synchronization codes are not stored in the line buffers. They should be sent according to the video port timings.

12.2.2.3 Multilane Layer

Peripherals do not typically have high bandwidth requirements for returning data to the host processor. To keep designs simple and improve interoperability, all DSI-compliant systems should only use Lane 1 in LP mode for returning data from a peripheral to the host processor.

12.2.2.3.1 SoT and EoT in Multilane Configurations

Since a HS transmission is composed of an arbitrary number of bytes that may not be an integer multiple of the number of lanes, some lanes may run out of data before others. Therefore, the lane management layer, as it buffers up the final set of less-than-N bytes, deasserts its *valid data* signal into all lanes for which there is no further data. Although all lanes start simultaneously with parallel SoTs, each lane operates independently and may complete the HS transmission before the other lanes, sending an EoT one cycle (byte) earlier.

12.2.2.3.2 Lane Splitter

The lane splitter can split the byte stream into 2 lanes (for 1 lane, the splitter is bypassed). Figure 12-41 and Figure 12-42 show the byte position into each serial link for 1 and 2 data lane configurations. The byte stream always starts from lane 1. It finishes on one of the lanes, depending on the number of bytes to send and the number of lanes. The splitter module is only used when packets are sent using high-speed mode (HS). In low-power mode (LP), only one data lane is used (Lane 1).

Figure 12-41. Two Data Lane Configuration

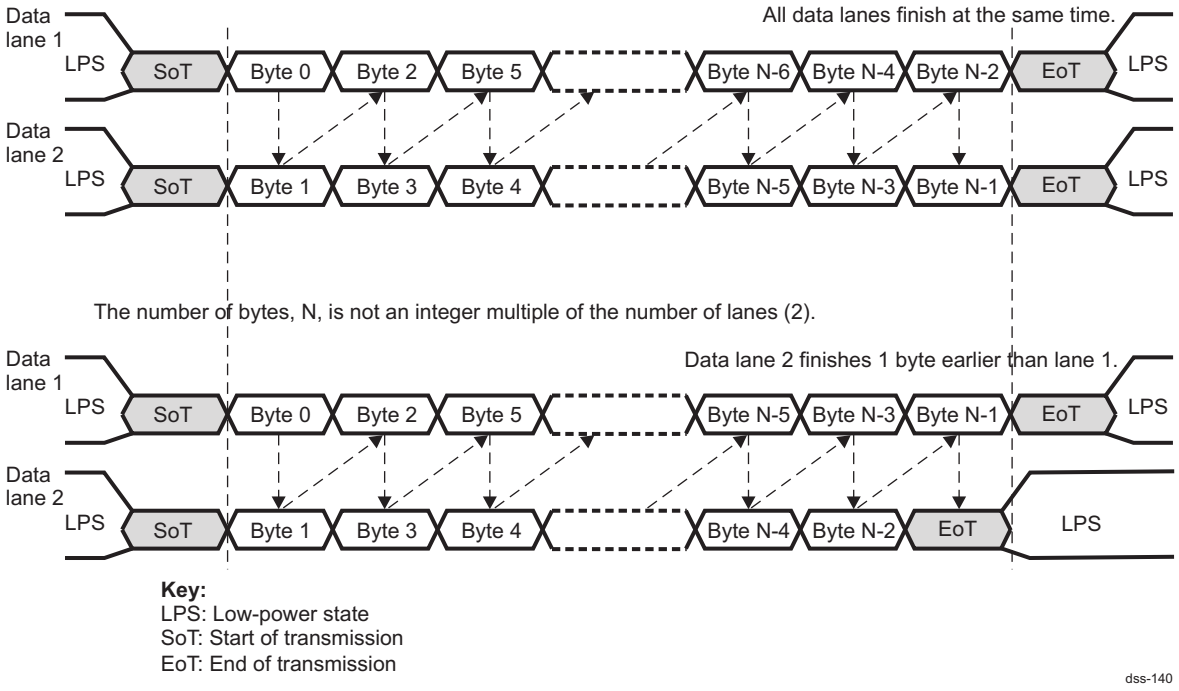
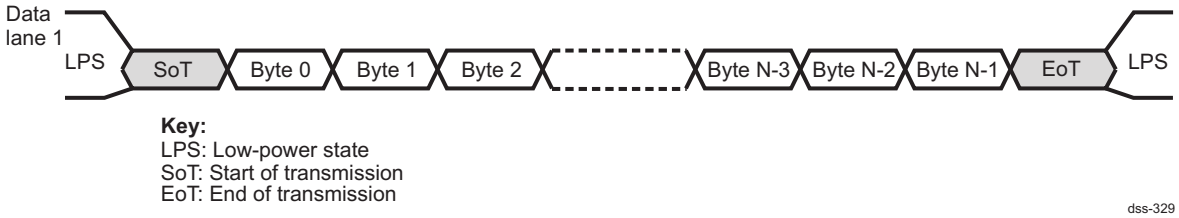
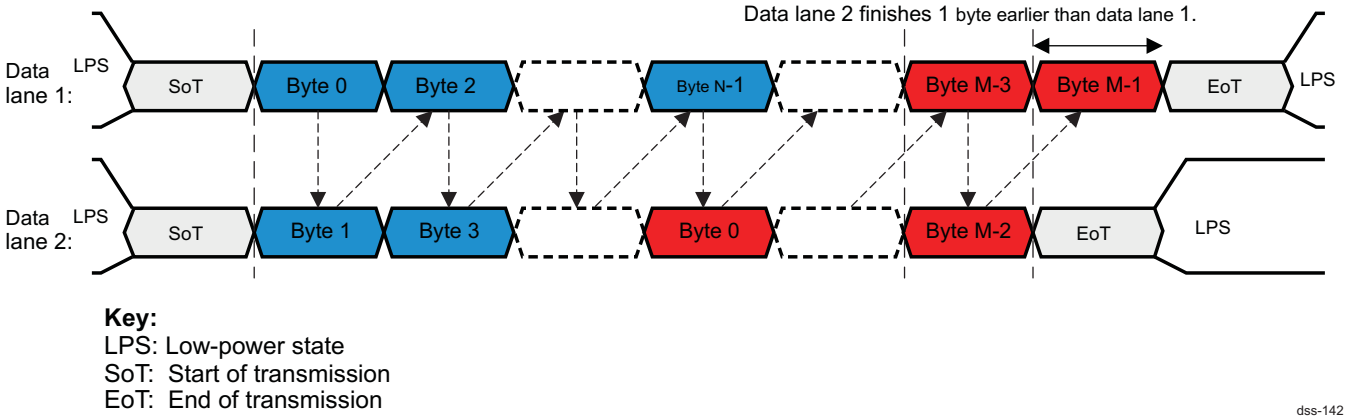


Figure 12-42. One Data Lane Configuration



In case of back-to-back packets, the byte stream is considered as a single packet by the splitter module. Figure 12-43 shows the example of two packets sent back to back. N bytes for the first packet and M bytes for the second one.

Figure 12-43. Two Packets Using Two-Data Lane Configuration (Example)



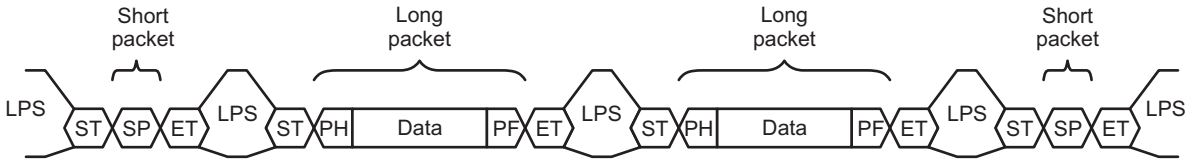
12.2.2.4 Protocol Layer

The low level protocol (LLP) is a byte-oriented protocol. It supports short and long packet formats. The DSI protocol layer defines how the display data is transported onto the physical layer. Packets can be sent using high-speed or low-speed mode. LLP is selected through DSI registers. The features of the DSI protocol layer are:

- Transport of arbitrary data (payload independent)
- 8-bit word size
- Support for up to four interleaved VCs on the same link
- Special packets for frame start, frame end, line start, and line end information
- Descriptor for the type, pixel depth, and format of application-specific payload data
- ECC for 1-bit or 2-bit error detection in the header
- 16-bit checksum code for error detection

Figure 12-44 shows the protocol layer with short and long packets.

Figure 12-44. Protocol Layer With Short and Long Packets



Key:

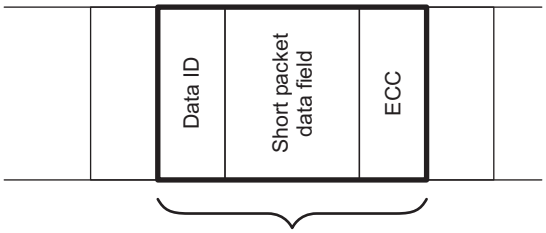
ST: Start of transmission	ET: End of transmission
PH: Packet header	PF: Packet footer
LPS: Low-power state	SP: Short packet

dss-143

12.2.2.4.1 Short Packet

Figure 12-45 shows the structure of the short packet. A short packet should contain an 8-bit data ID followed by two command or data bytes and an 8-bit ECC; a packet footer should not be present. Short packets should be four bytes in length. The ECC byte allows correction of single-bit errors and detection of 2-bit errors in the short packet.

Figure 12-45. Short Packet Structure



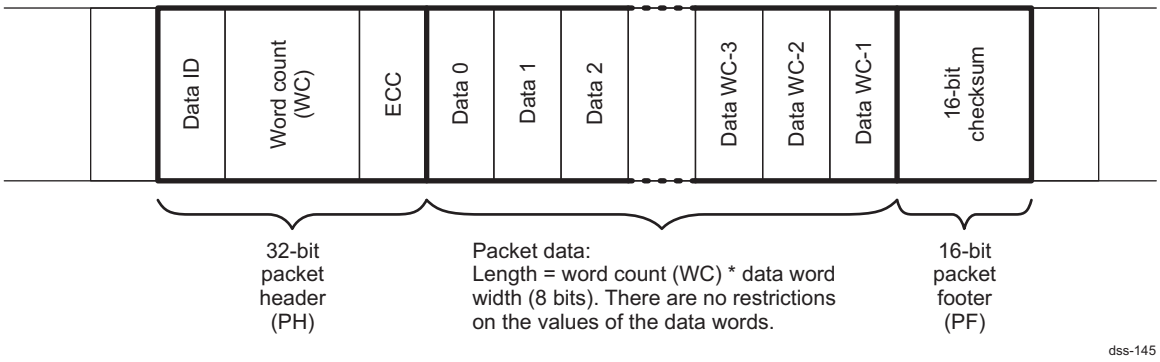
dss-144

NOTE: The short packets can be sent in low-power mode or in high-speed mode.

12.2.2.4.2 Long Packet

Figure 12-46 shows the structure of the low-level protocol long packet. A long packet must consist of three elements: A 32-bit packet header (PH), an application-specific data payload with a variable number of bytes, and a 16-bit packet footer (PF). The packet header is further composed of three elements: An 8-bit data identifier, a 16-bit word count, and 8-bit ECC. The packet footer has one element, a 16-bit checksum. Long packets can be from 6 to 65,541 bytes in length.

Figure 12-46. Long Packet Structure



- The data identifier defines the VC for the data and the DT for the application-specific payload data.
- The word count defines the number of bytes in the data payload between the end of the packet header and the start of the packet footer. Neither the packet header nor the packet footer should be included in the word count.
- The ECC byte allows single-bit errors to be corrected and 2-bit errors to be detected in the packet header. This includes the data identifier and the word count fields.

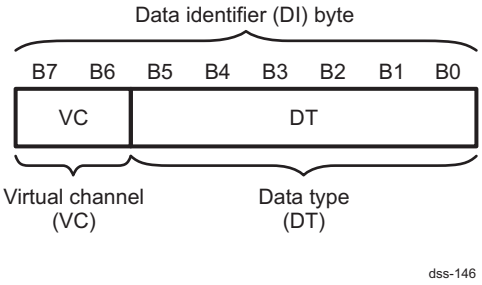
After the end of the packet header, the receiver reads the next word count * bytes of the data payload. There are no limitations on the value of a data word within the data payload block, that is, no embedded codes are used. Once the receiver has read the data payload, it reads the checksum in the packet footer. The host processor should always calculate and transmit a checksum in the packet footer. Peripherals are not required to calculate a checksum. Also note the special case of zero-byte data payload: If the payload has length 0, then the checksum calculation results in (0xFFFF). If the checksum is not calculated, the packet footer should consist of two bytes of all zeros (0x0000). In the generic case, the length of the data payload should be a multiple of bytes. In addition, each data format may impose additional restrictions on the length of the payload data, that is, multiple of four bytes. Each byte is transmitted LSB first. Payload data may be transmitted in any byte order, restricted only by data format requirements. Multibyte elements such as word count and checksum should be transmitted least-significant byte first.

NOTE: The long packets can be sent in low-power mode or in high-speed mode.

12.2.2.4.3 Data Identifier

The data identifier byte contains the virtual VC ID value and the DT value as illustrated in Figure 12-47. The VC ID is contained in the two MS bits of the data identifier byte. The DT value is contained in the six LSBs of the data identifier byte. DI[7:6]: These two bits identify the data as directed to one of four VCs. DI[5:0]: These six bits specify the DT.

Figure 12-47. Data Identifier Structure

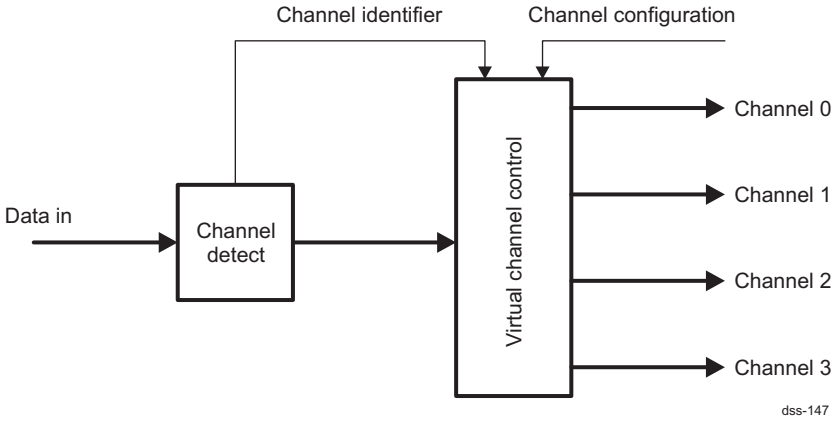


12.2.2.4.4 Virtual Channel ID - VC Field, DI[7:6]

The host can service up to four peripherals with tagged commands or blocks of data using the VC ID field of the header for packets targeted at different peripherals. The VC ID enables one serial stream to service two or more virtual peripherals by multiplexing packets onto a common transmission channel. Note that each packet sent in a single transmission have its own VC assignment and can be directed to different peripherals. The VC ID is defined in the DSS.DSI_VCn_SHORT_PACKET_HEADER and DSS.DSI_VCn_LONG_PACKET_HEADER registers for short and long packets, respectively. It should not be modified by hardware. There is one set of registers for each VC. Each set of registers defines the characteristics of the traffic between the host and the display associated with the VC.

Figure 12-48 shows the VC controller.

Figure 12-48. Virtual Channel Controller



12.2.2.4.5 Data Type Field DT[5:0]

The DT field specifies whether the packet is a long or short packet type and the packet format. The DT field, along with the word count field for long packets, informs the receiver on how many bytes to expect in the remainder of the packet. This is necessary because there are no special packet start/end sync codes to indicate the beginning and end of a packet. This permits packets to convey arbitrary data, but it also requires the packet header to explicitly specify the size of the packet.

12.2.2.4.6 Pixel Data Formats in Video Mode

The host can send different pixel formats in video mode. Table 12-13 summarizes the pixel formats supported by the DSI interface in video mode.

Table 12-13. Pixel Data Format in Video Mode

Mode	Description
RGB888 (using 24-bit container)	RGB888
RGB666 (using 24-bit container)	RGB666

Table 12-13. Pixel Data Format in Video Mode (continued)

Mode	Description
RGB666 (18-bit packet using 18-bit container)	RGB666_PACKET
RGB565 (using 16-bit container)	RGB565

12.2.2.4.7 Synchronization Codes

Each frame can be identified by two synchronization codes: One for the start of vertical synchronization pulse (VSSC) and one for the end of the vertical synchronization pulse (VSEC). Each line can be identified by two synchronization codes: One for the start of horizontal synchronization pulse (HSSC) and one for the end of the horizontal synchronization pulse (HSEC). The synchronization events may not be required by the display (peripheral): They are optional. Users can program which sync events are generated to the display from the timings received from the display controller in video mode. When data are received on the L4 interconnect slave port, the synchronization codes are not automatically generated by the protocol engine. They can be provided on the L4 interconnect port by writing to the registers with limited timing control. It is highly recommended to use the video port from the display controller to receive the synchronization events to automatically generate the short synchronization packets to the peripheral. When the DSI protocol engine detects that the VSYNC signal from the display controller transitions from inactive to active state, the VSSC short packet replaces the following HSSC corresponding to the following HSYNC synchronization short packet (if the generation is enabled). When the transition from active to inactive state is detected, the VSEC short packet is generated (if the generation is enabled) replacing the HSSC synchronization packet corresponding to the following HSYNC. When the DSI protocol engine detects that the HSYNC signal from the display controller transition from inactive to active state, the HSSC short packet is generated (if the generation is enabled). When the transition from active to inactive state is detected, the HSEC short packet is generated (if the generation is enabled). For the first frame, any HSYNC and data received on the video port prior to the first VSYNC should be ignored. Since the first VSYNC sent to the display is also recognized as a HSYNC for the first line, there should be no HSYNC sent for the first line. To send the synchronization codes, the DSI protocol engine uses the short packets. Table 12-14 summarizes the 6-bit DT synchronization code values.

Table 12-14. Synchronization Codes

Synchronization Code	Value	Comments
V sync start code (VSSC)	0x1	Optional
V sync end code (VSEC)	0x11	Optional
H sync start code (HSSC)	0x21	Optional
H sync end code (HSEC)	0x31	Optional

12.2.2.4.8 Blanking

To keep the DSI link in HS state while using the video mode, during blanking periods, the long blanking packets are sent to the display. The DSS.DSI_VM_TIMINGi (I between 1 and 7) registers define the size of the long blanking packets after:

- Horizontal sync start code (short packet)
- Horizontal sync end code (short packet)
- Vertical sync start code (short packet)
- Vertical sync end code (short packet)
- Pixels (long packet)

Table 12-15 defines the short packet values for the synchronization packets:

Table 12-15. Sync Short Packet Values

Virtual Channel ID	Sync Code	Header (1st Byte)	Header (2nd Byte): Data Field LSB	Header (3rd Byte): Data Field MSB	Header (ECC)
0x0	0x1	0x1	0x0	0x0	See note following this table
	0x11	0x11			
	0x21	0x21			
	0x31	0x31			
0x1	0x1	0x41			
	0x11	0x51			
	0x21	0x61			
	0x31	0x91			
0x2	0x1	0x81			
	0x11	0x81			
	0x21	0xA1			
	0x31	0xB1			
0x3	0x1	0xC1			
	0x11	0xD1			
	0x21	0xC1			
	0x31	0xF1			

NOTE:

- If the ECC is enabled by setting the DSS.DSI_VCn_CTRL[8] ECC_TX_EN bit to 1 for the VC in video mode, the ECC value is calculated; otherwise, 0x00 is used for the blanking long packets and sync short packets. If the CRC is enabled by setting the DSS.DSI_VCn_CTRL[7] CS_TX_EN bit to 1 for the VC in video mode, the check-sum value is calculated; otherwise, 0x00 is used for the blanking long packets.
- In other cases, when the DSS.DSI_VCn_CTRL[7] CS_TX_EN bit is set to 0, the value 0x00 is always used for the CRC (long packets). When the DSS.DSI_VCn_CTRL[8] ECC_TX_EN bit is set to 0, the value 0x00 is used for the ECC for short and long packets, except when the header is provided by the register, since the ECC field is available in the register. It can be used to generate invalid ECC values when the header is provided by the register.

The link [lane(s) and clock separately] can be put in ULPS mode. While using the blanking values formerly defined, the packets (short and long) are considered in HS mode.

Timing parameters VSA, VBP, VFP, HSA, HBP, HFP, VACT, and t_L are defined in the DSS.DSI_VM_TIMINGx (x between 1 and 7) register. HSA, HBP, HFP, and t_L are defined using the byte clock unit (TxByteClkHS) and also in low-power clock cycles (TxClkEsc). VSA, VBP, VFP, and VACT are defined in term of number of lines. When the HS blanking packets are sent during the blanking periods, the parameters are used to determine the blanking packet payload size (taking into account the 4-byte header and the 2-byte check sum).

The configuration of the display controller timing generator must be used when the display controller timings are used to generate the DSI HS video mode transfer.

Special care must be taken in the case of the last line of the frame. The LPS transition is required when the link is in HS mode for the whole frame.

When BTA is sent for the data packets, the following blanking period can not be used for sending any data from the TX FIFO. When the blanking period starts with one HS packet from one VC, it can only be followed by another HS packet from the same VC, or by trigger (BTA for example). When there is no more HS data to send for this VC, the lane is in LPS. When the blanking period starts with one LP packet from one VC, it can only be followed by another LP packet from the same VC, by another VC, by trigger (BTA

for example), or by extra LP NULL packets. If the trigger has been sent, it is not possible to send any more data. When there is no more data from the TX FIFO to send in LP mode or the trigger has been sent, the lane is put into LPS. If the lanes must be kept in HS mode during blanking periods (except for the last blanking period of the frame), the HS blanking packets must be used. In case one trigger is sent at the beginning of the blanking period, the rest of the blanking period is in ULPS.

Figure 12-49 and Figure 12-50 show a nonburst transfer in DSI video mode with, and without VE and HE, respectively.

Figure 12-49. DSI Video Mode: Nonburst Transfer With VE and HE

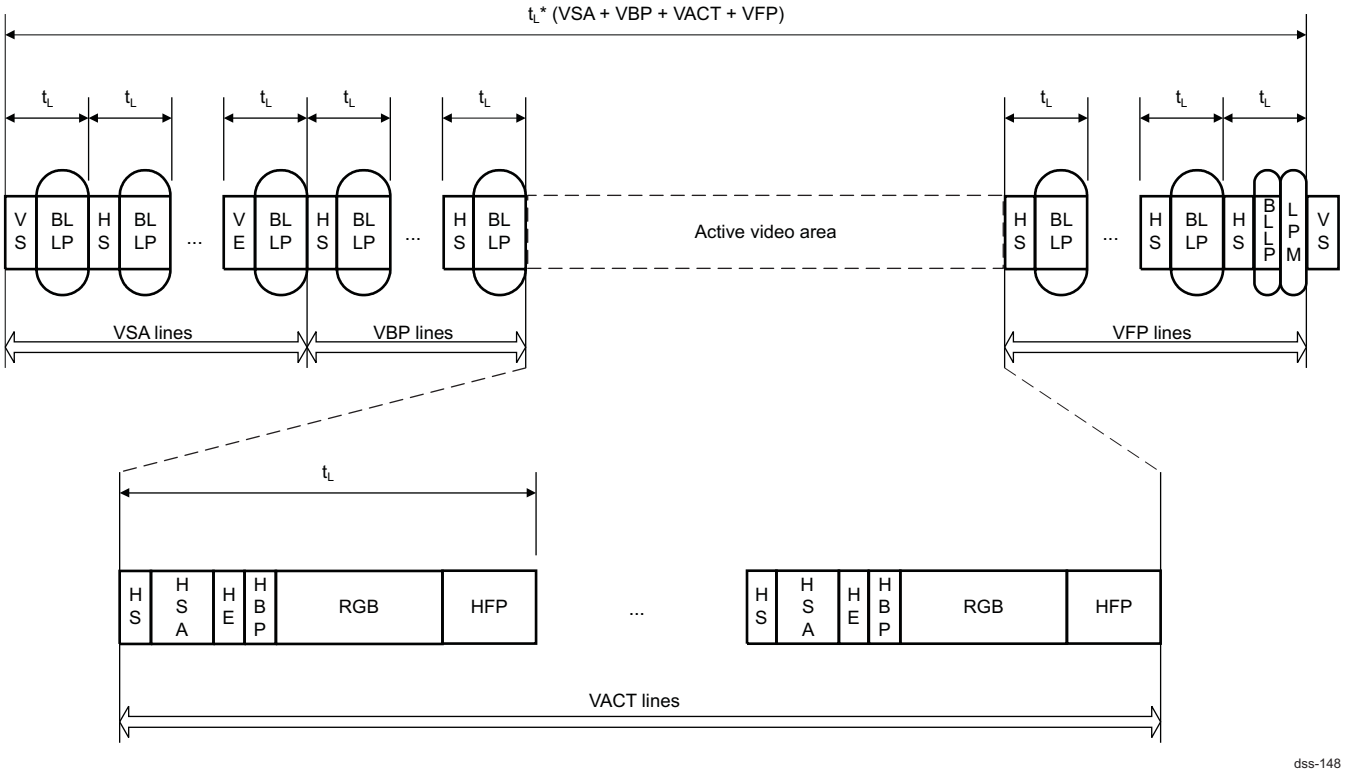
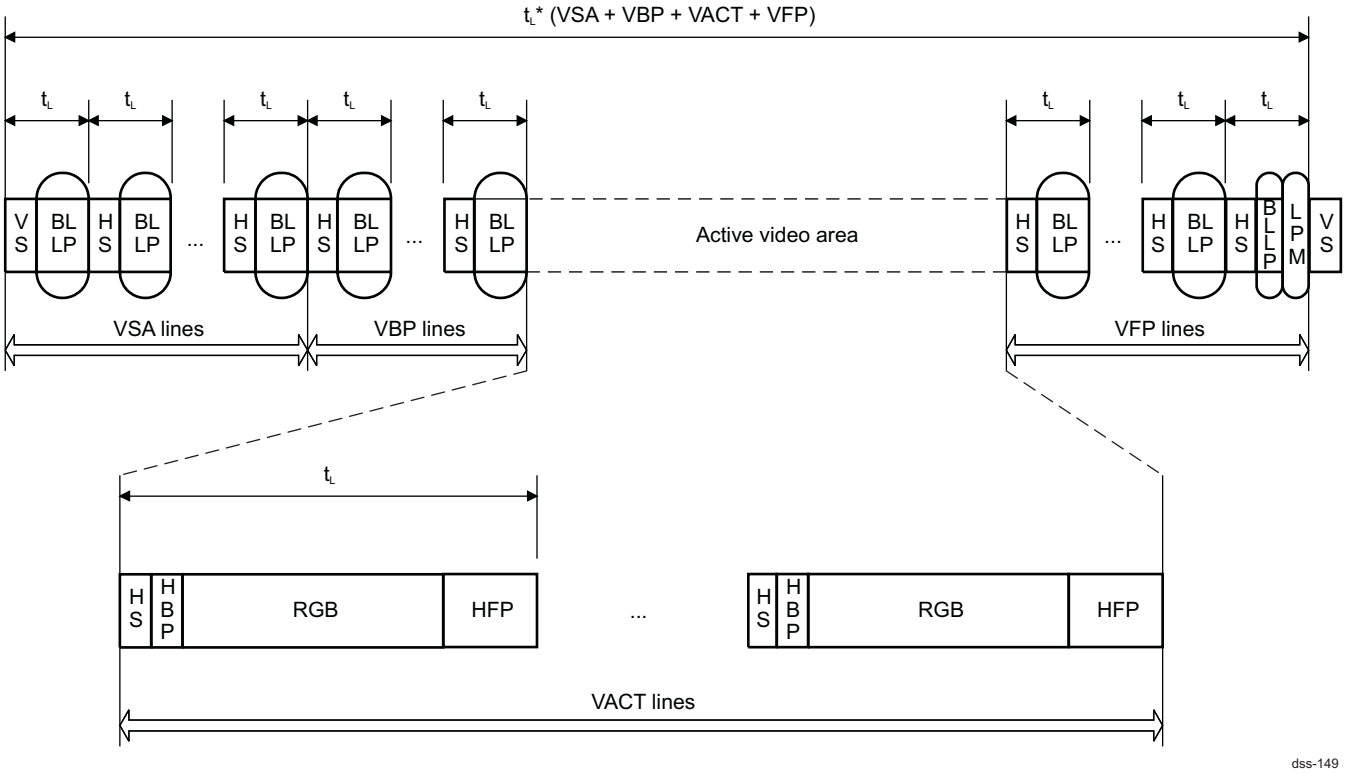
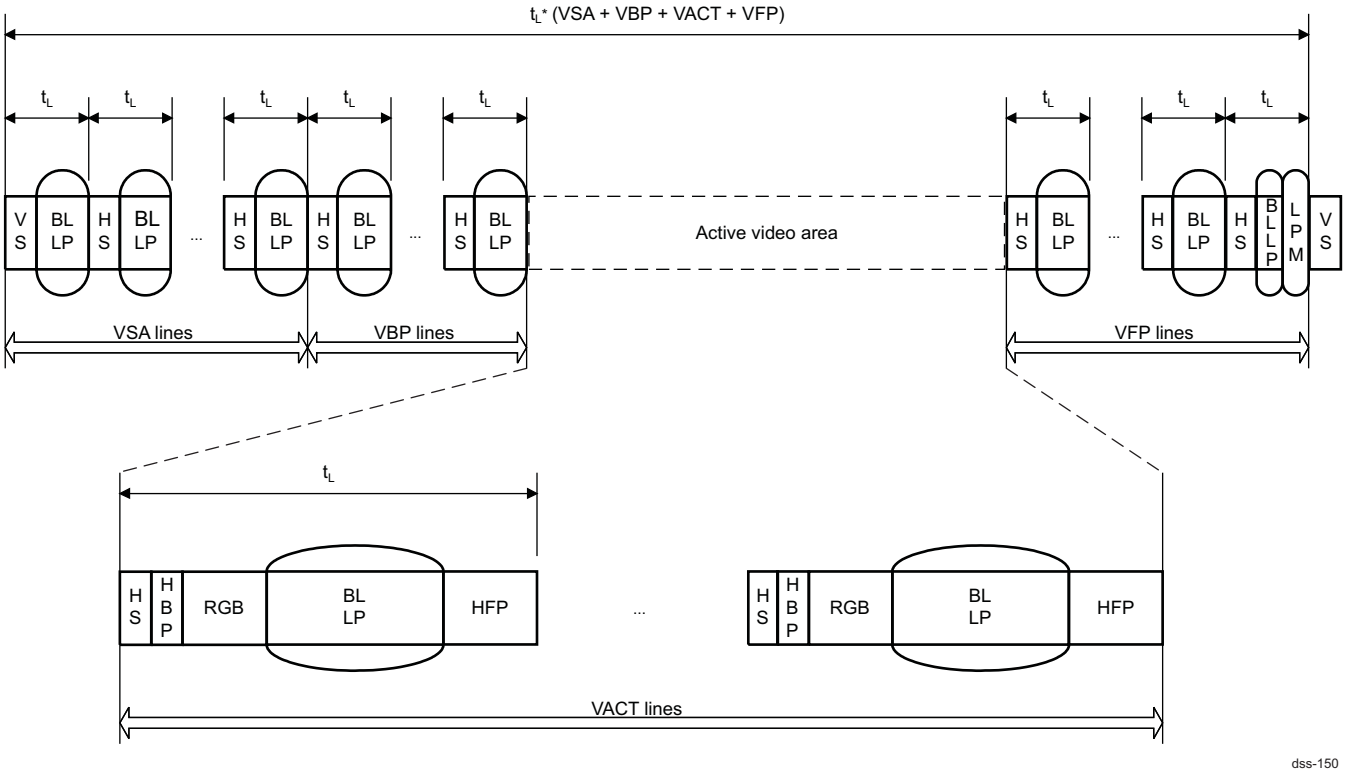


Figure 12-50. DSI Video Mode: Nonburst Transfer Without VE and HE



NOTE: HSA timing is not used and does not have to be programmed when HE short packet is not generated.

Figure 12-51. DSI Video Mode: Burst Transfer Without VE and HE



NOTE: HSA timing is not used and does not have to be programmed when HE short packet is not generated.

In Figure 12-50 and Figure 12-51, if HSync end short packet is not generated (HSA does not exist), HBP should be other than 0.

12.2.2.4.9 Frame Structures

Figure 12-52 shows the general DSI frame structure.

Figure 12-52. DSI General Frame Structure

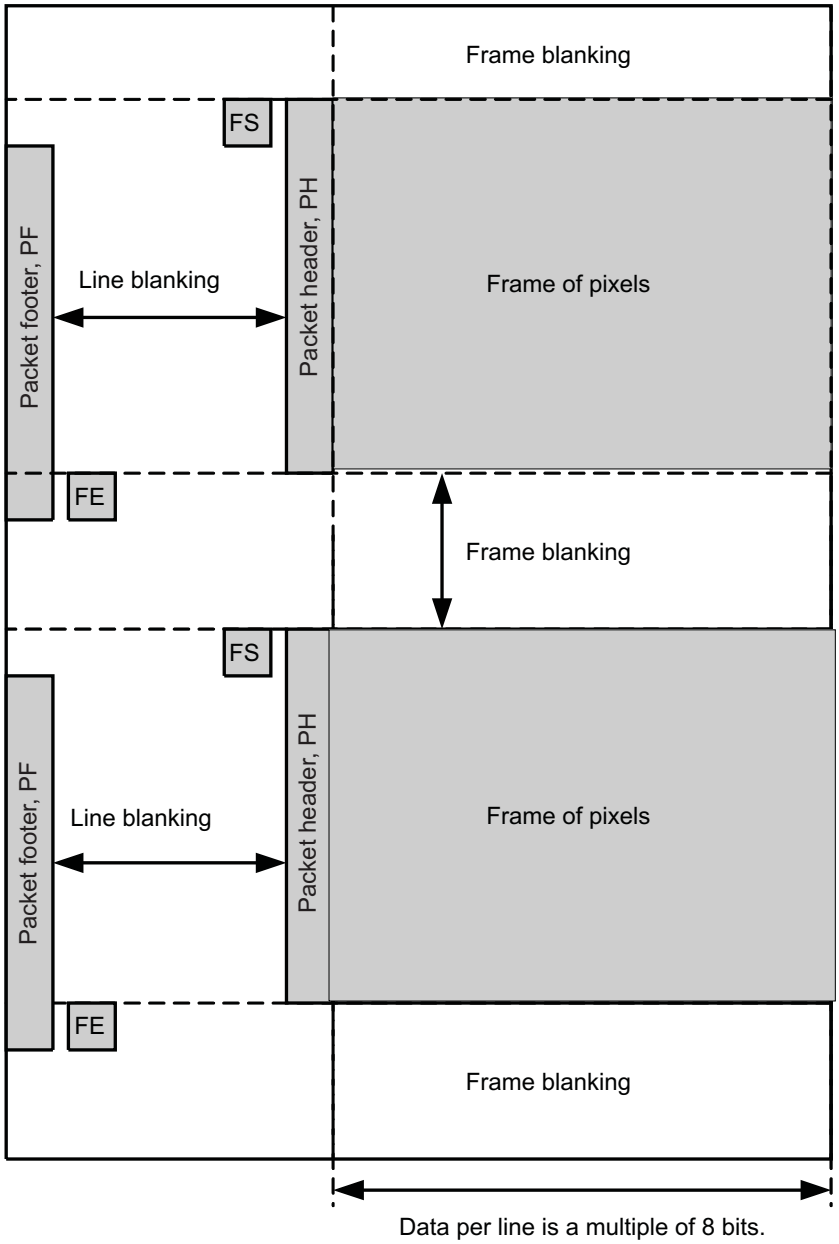


Figure 12-53 shows the general frame structure using burst mode.

Figure 12-53. DSI General Frame Structure Using Burst Mode

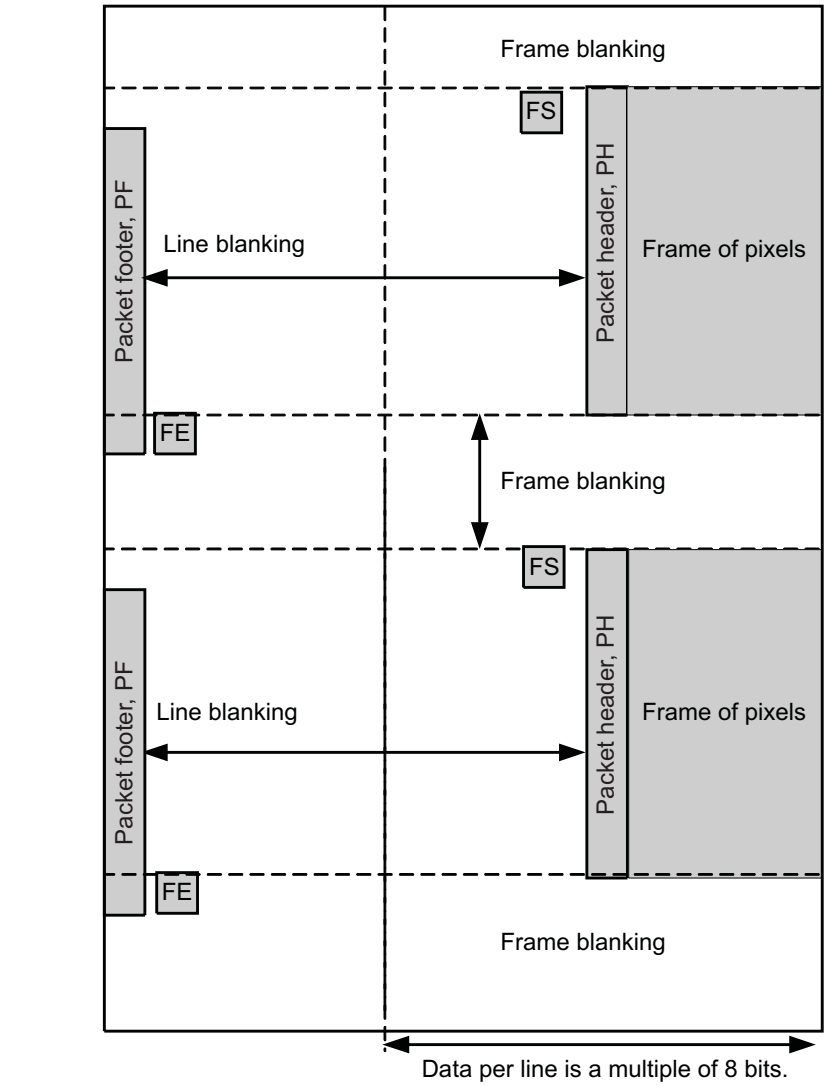
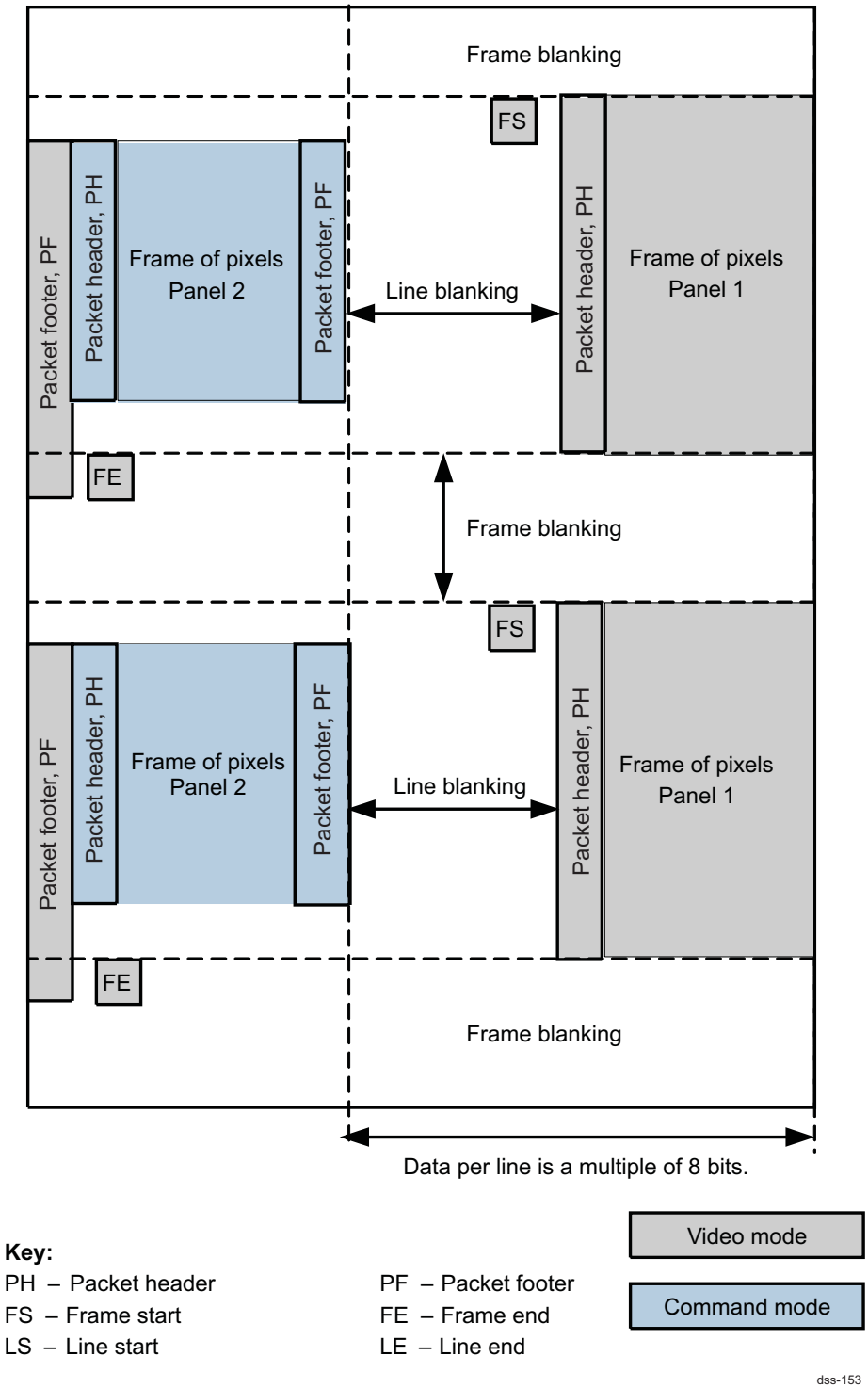


Figure 12-54 shows the general frame structure using burst mode and interleaving.

Figure 12-54. DSi General Frame Structure Using Burst Mode and Interleaving



12.2.2.4.10 Virtual Channels

The DSI protocol layer transports VCs. The purpose of VCs is to separate different data flows, which are interleaved in a same data stream. Each VC is identified by a unique channel identification number in the packet header. The channel identification number is encoded in 2-bits. The DSI protocol engine determines the channel identifier number to be used for generating the packet header and multiplexes the interleaved data streams. The DSI protocol engine supports multiple concurrent VCs: Up to 4. Table 12-16 summarizes the VC values used for each channel.

Table 12-16. Virtual Channel Values

Virtual Channel Number	Value
Virtual channel 0	0x0
Virtual channel 1	0x1
Virtual channel 2	0x2
Virtual channel 3	0x3

In the case of multiple displays connected to the single DSI port on the host, a hub may be used to root the data stream to the appropriate display based on the VC ID. Typically, VC ID 0x0 is used for the primary display and 0x1 for the secondary. The hub may have its own VC ID to provide communication capability between the host and the hub.

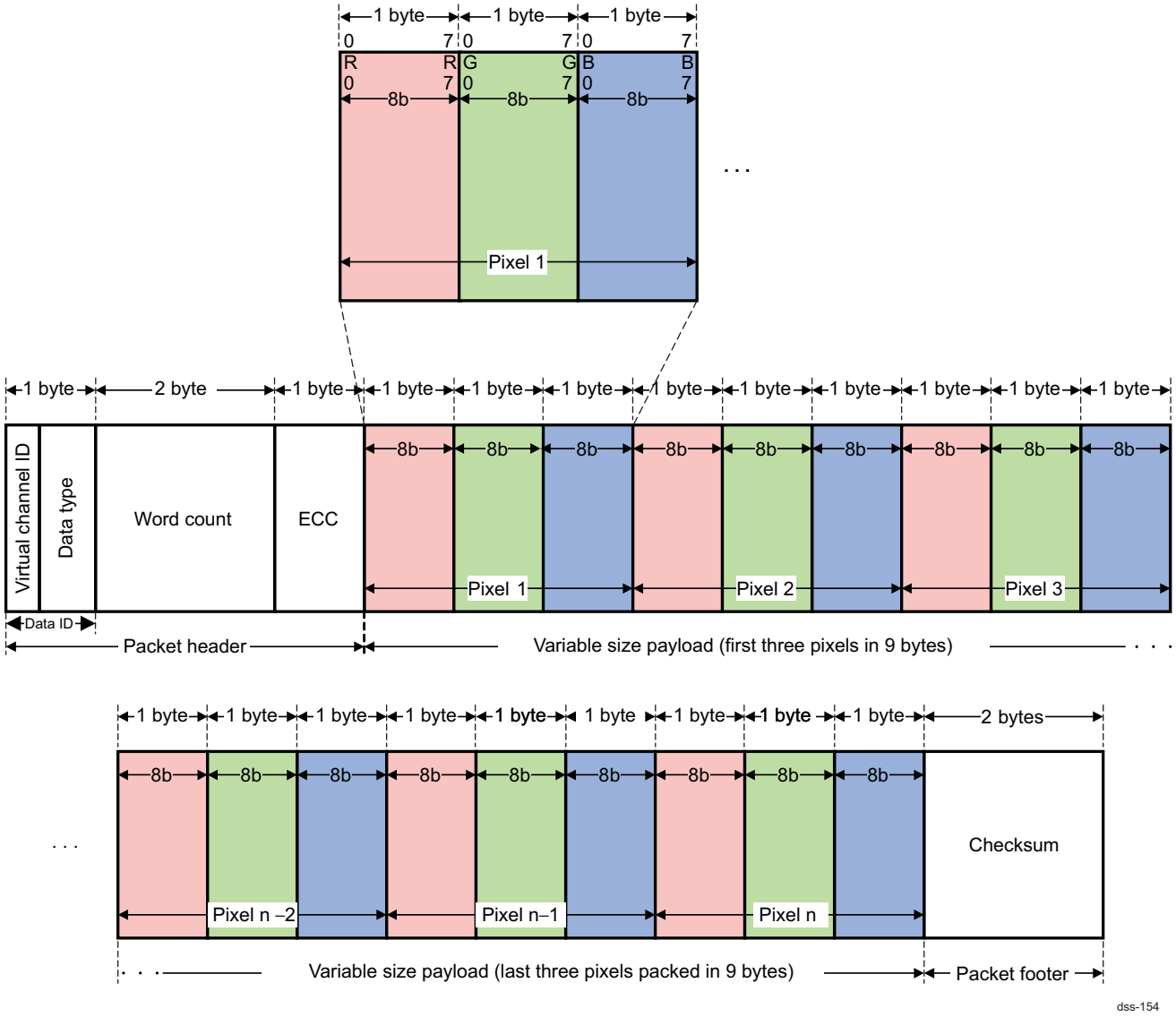
12.2.2.5 Pixel Data Formats

This section summarizes how the DSI supported pixel data formats in video mode are transmitted over the serial interface. For pixel formats in command mode. The DSI protocol engine can cope with all data formats given that the data line length sent through the DSI physical protocol is a multiple of a pixel. This condition is required for the DSI protocol engine to work properly.

12.2.2.5.1 24 Bits per Pixel - RGB Color Format, Long Packet

Figure 12-55 shows the RGB888 format.

Figure 12-55. 24 Bits per Pixel RGB Color Format, Long Packet

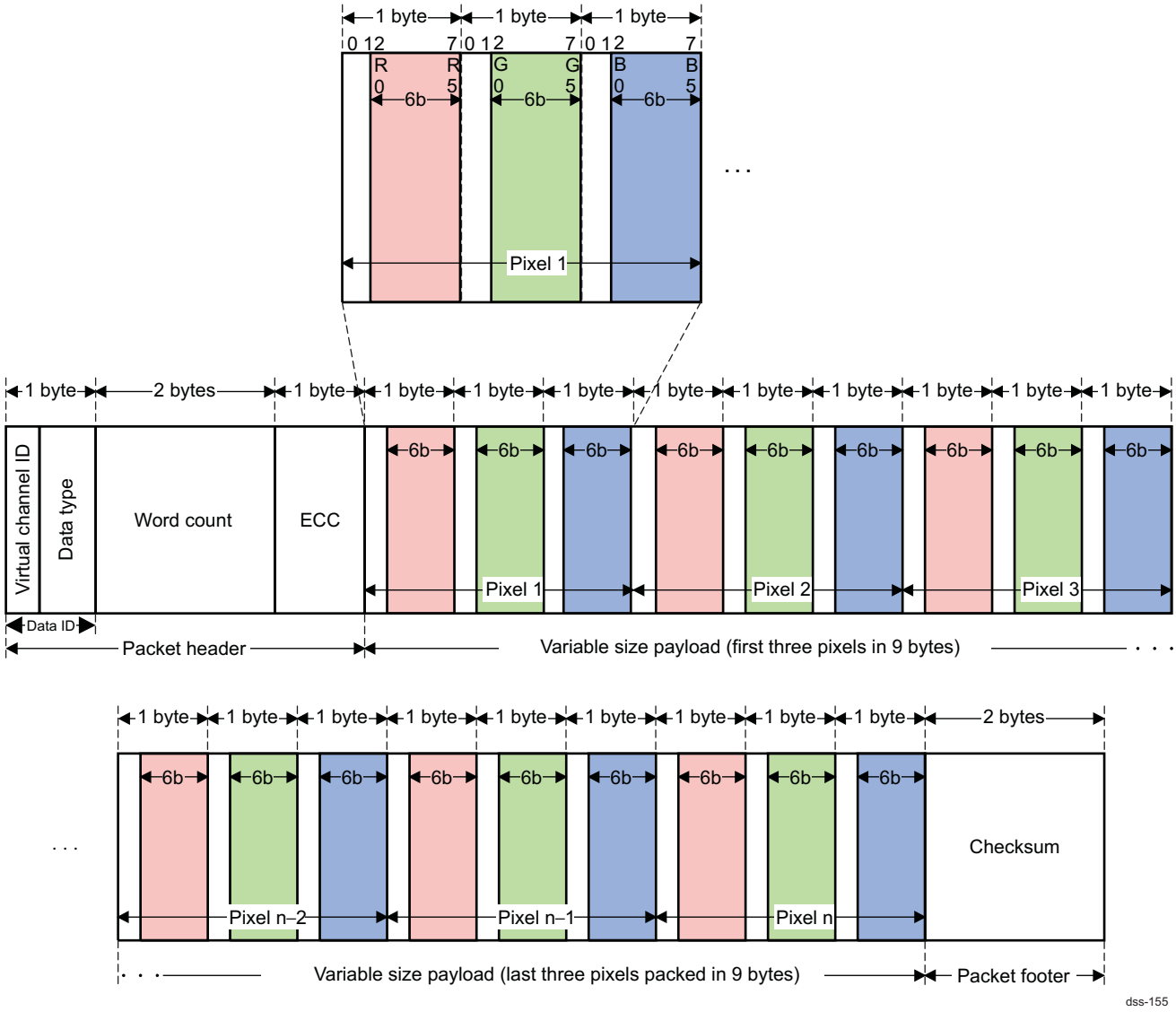


Packed pixel stream 24-bit format is a long packet used to transmit image data formatted as 24-bit pixels to a video mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes, and a two-byte checksum. The pixel format is red (8 bits), green (8 bits), and blue (8 bits), in that order. Each color component occupies one byte in the pixel stream; no components are split across byte boundaries. Within a color component, the LSB is sent first, the MSB last.

12.2.2.5.2 18 Bits per Pixel (Loosely Packed) - RGB Color Format, Long Packet

Figure 12-56 details the RGB666 format.

Figure 12-56. 18 Bits per Pixel (Loosely Packed) RGB Color Format, Long Packet

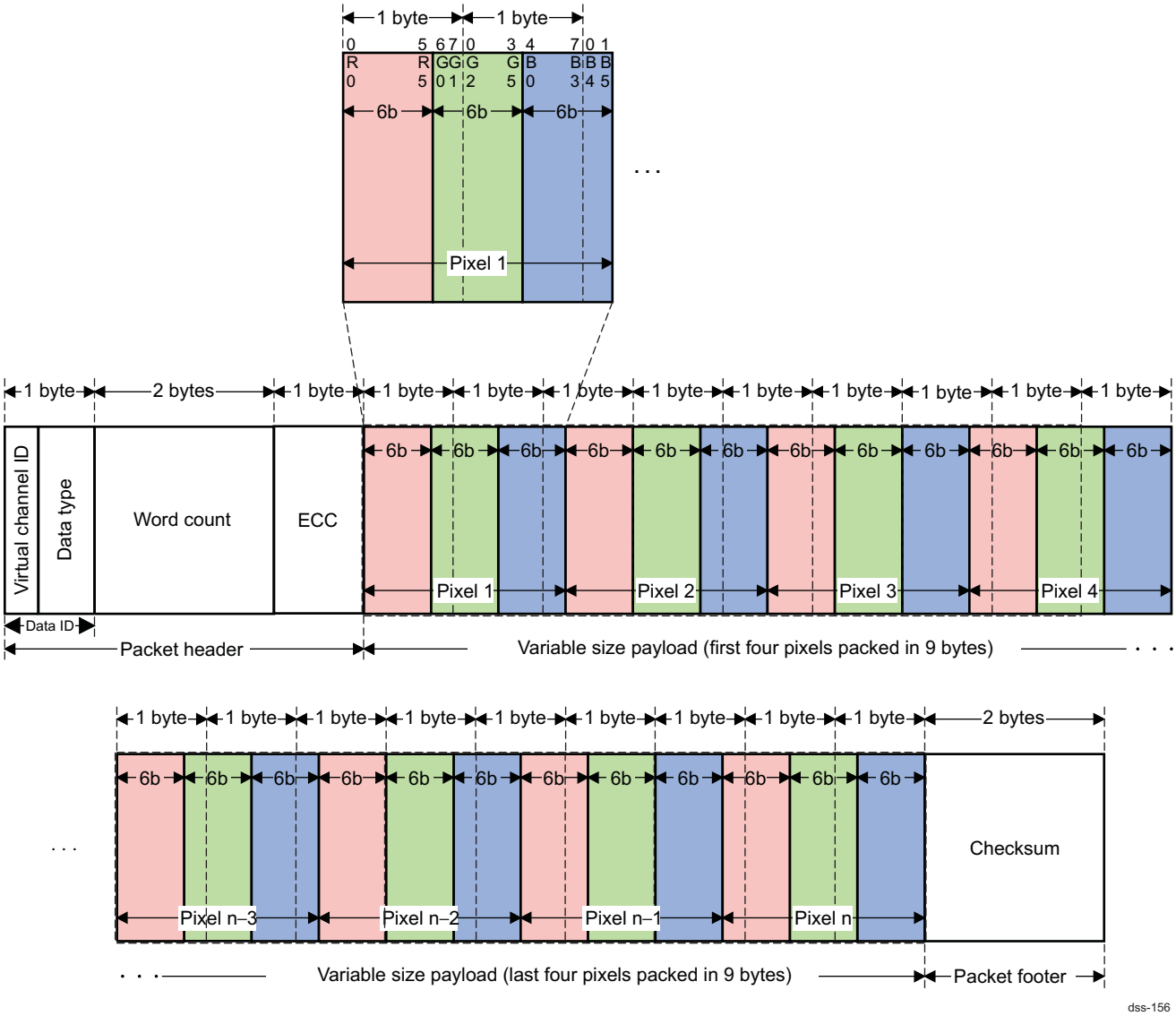


In the 18-bit pixel loosely-packed format, each R, G, or B color component is six bits but is shifted to the upper bits of the byte, such that the valid pixel bits occupy bits [7:2] of each byte. Bits [1:0] of each payload byte representing active pixels are ignored. As a result, each pixel requires three bytes as it is transmitted across the link. This requires more bandwidth than the packed format, but requires less shifting and multiplexing logic in the packing and unpacking functions on each end of the link. This format is used to transmit RGB image data formatted as pixels to a video mode display module that displays 18-bit pixels. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes, and a two-byte checksum. The pixel format is red (6 bits), green (6 bits), and blue (6 bits) in that order. Within a color component, the LSB is sent first, the MSB last.

12.2.2.5.3 18 Bits per Pixel (Packed) - RGB Color Format, Long Packet

Figure 12-57 details the RGB666_PACKED format.

Figure 12-57. 18 Bits per Pixel (Packed) RGB Color Format, Long Packet

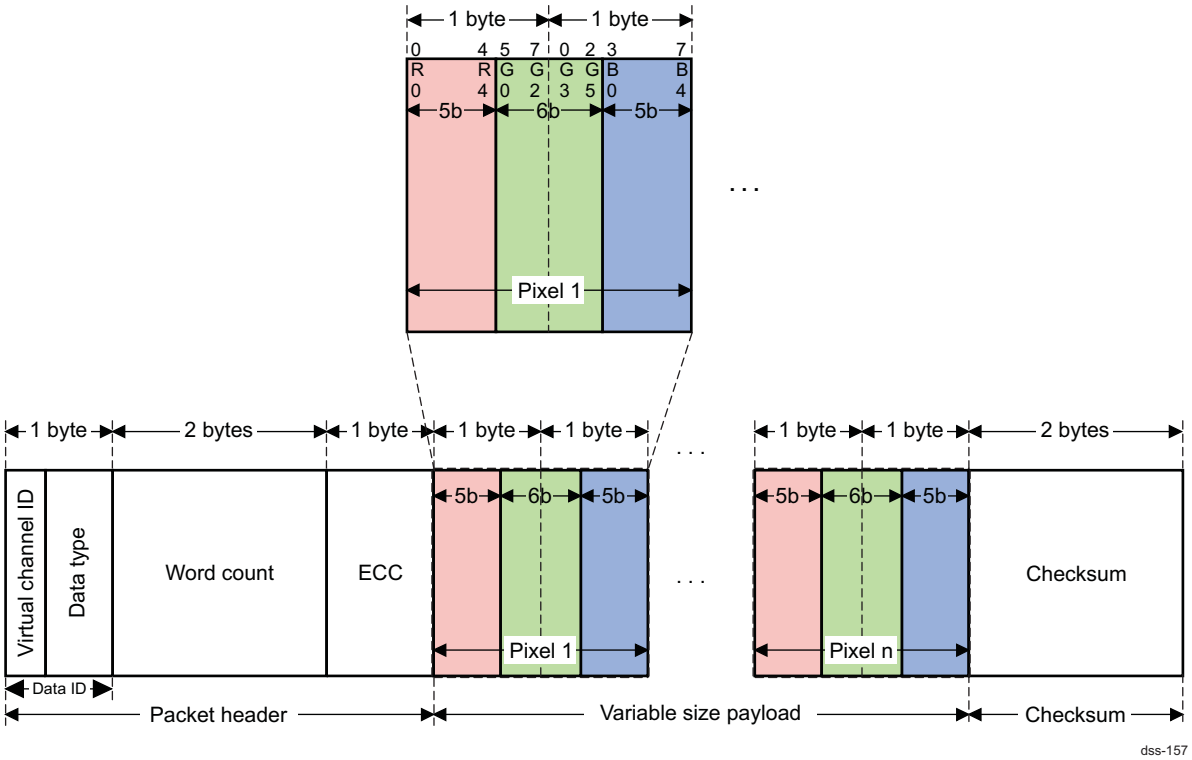


Packed pixel stream 18-bit format (packed) is a long packet. It is used to transmit RGB image data formatted as pixels to a video mode display module that displays 18-bit pixels The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes, and a two-byte checksum. Pixel format is red (6 bits), green (6 bits), and blue (6 bits), in that order. Within a color component, the LSB is sent first, the MSB last. With this format, it is strongly recommended that the total line width be a multiple of four pixels (nine bytes).

12.2.2.5.4 16 Bits per Pixel - RGB Color Format, Long Packet

Figure 12-58 details the RGB565 format.

Figure 12-58. 16 Bits per Pixel RGB Color Format, Long Packet



Packed pixel stream 16-bit format is a long packet used to transmit image data formatted as 16-bit pixels to a video mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes, and a two-byte checksum. Pixel format is five bits red, six bits green, five bits blue, in that order. Note that the *green* component is split across two bytes. Within a color component, the LSB is sent first, the MSB last.

12.2.3 LCD Output With TI FlatLink3G Data Format for the SDI Module

The parallel-to-serial conversion of the 24-BPP pixel data depends on the number of data channels in use.

Figure 12-59 through Figure 12-61 summarize the pixel order for each data format of the TI FlatLink3G LCD panel.

The following apply to Figure 12-59 through Figure 12-61:

- VS: Vertical synchronization
- HS: Horizontal synchronizations
- DE: Data enable
- CP: Pixel parity. Parity of the whole 30-bit data is odd
- F1, F0: Reserved bits for the TI FlatLink3G protocol
- Clock channel: Differential clock signal between SDI_CLKP and SDI_CLKN

Figure 12-59. 24 Bits Per Pixel With One Data Channel

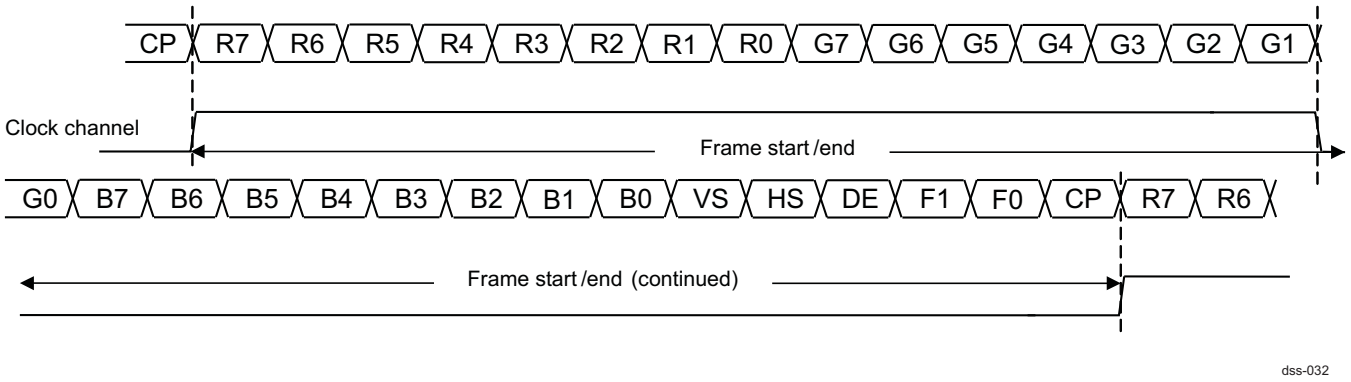


Figure 12-60. 24 Bits Per Pixel With Two Data Channels

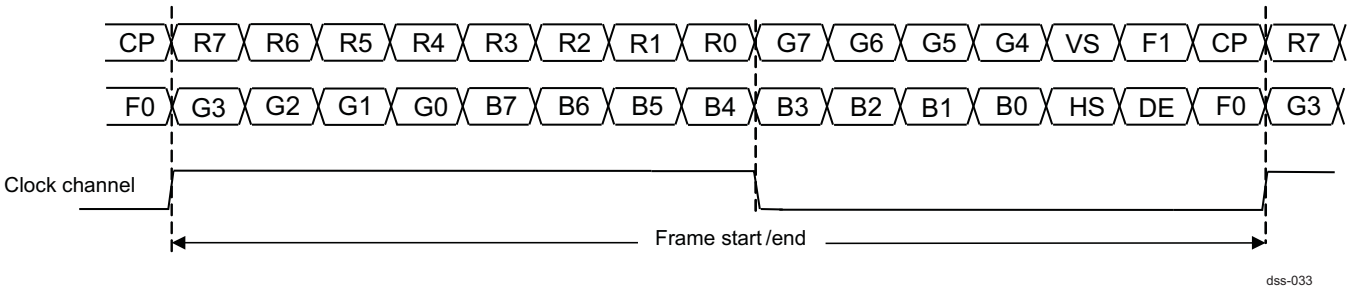
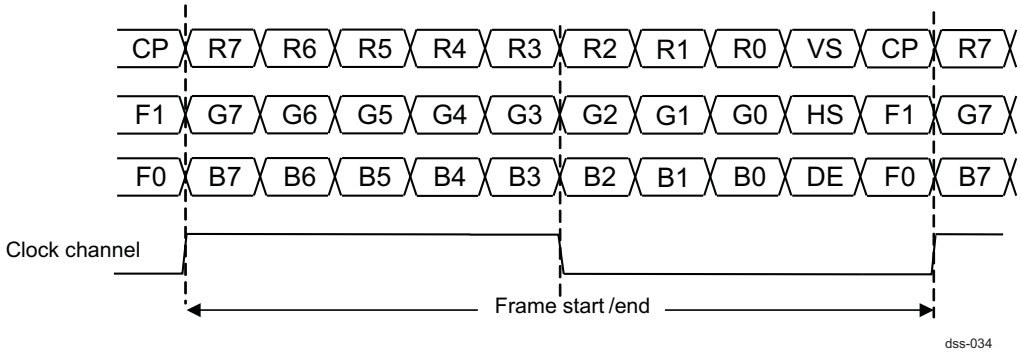


Figure 12-61. 24 Bits Per Pixel With Three Data Channels



NOTE: The range of the data rates permitted is 120 Mbps per pair minimum (4 MHz pixel clock, one data pair) to 650 Mbps per pair maximum (65 MHz pixel clock, three data pairs).

12.2.4 TV Display Support

The TV display path includes the following modules:

- Display controller
- Video encoder
- Dual 10-bit DAC with video amplifiers

The display controller module receives synchronization signals from the video encoder and synchronously sends pixel data to the video encoder with these signals. The digital output of the display controller is always a 24-bit RGB value based on a pixel request from the video encoder.

The video encoder converts RGB video signals to conform to the NTSC/PAL standard analog video. The video encoder includes an integrated synchronization signal generator and a 2-channel video digital-to-analog converter (DAC) with video amplifiers, data manager, luma stage, chroma stage, modulator, and a control interface.

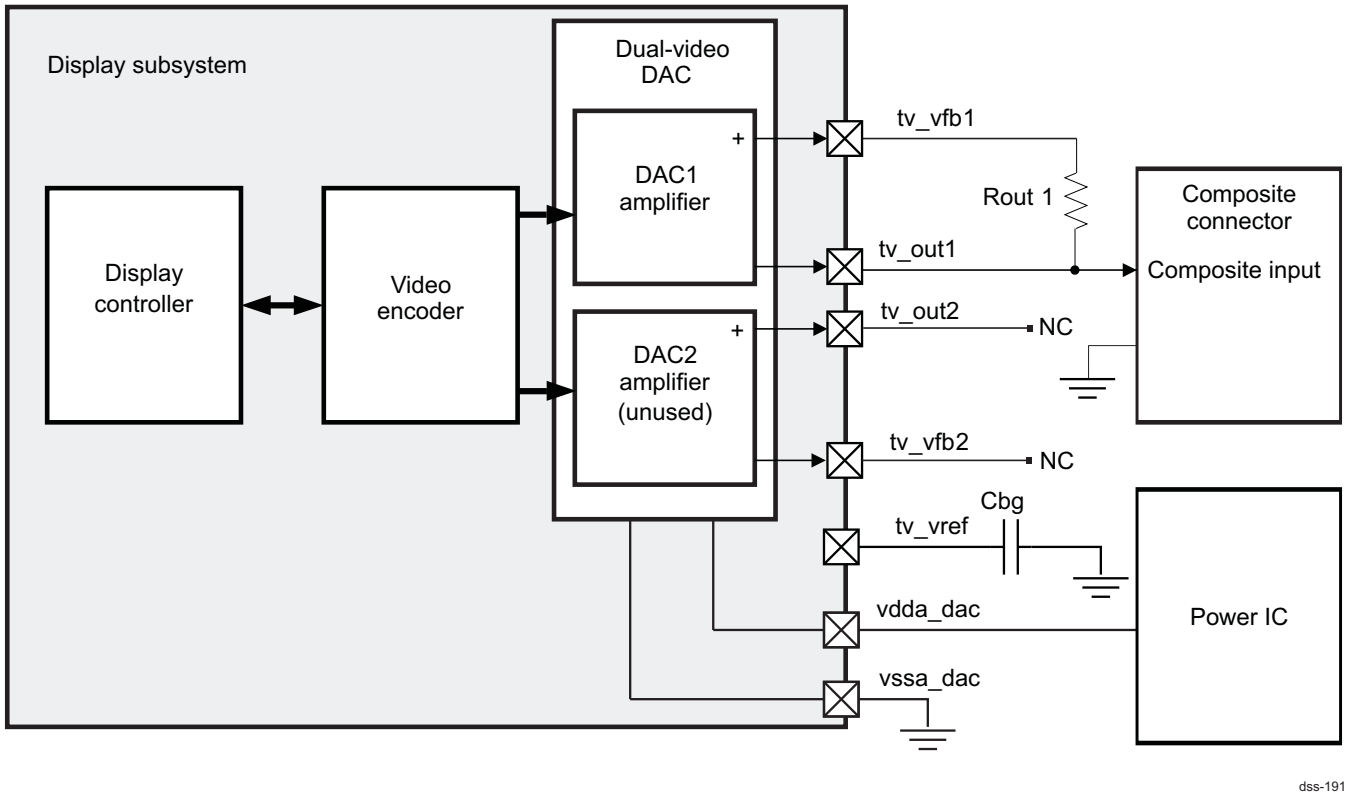
The video encoder also provides the synchronization signals to the display controller: VSYNC, active VIDeo (AVID), and field ID (FID).

Figure 12-62 is a block diagram of the TV display interface (s-video mode).

Figure 12-62. TV Display Interface (s-video mode)

Figure 12-63 is a block diagram of the TV display interface (composite mode).

Figure 12-63. TV Display Interface (Composite Mode)



NOTE: In composite video mode, the video DAC2 chroma output must be disabled by setting the DSS.VENC_OUTPUT_CONTROL[2] CHROMA_ENABLE bit to 0.

Table 12-17 describes the interface signals to/from the TV set for TV display support.

Table 12-17. TV Display Interface Pins

Pin Name	Type[1]	Description
tv_out1	O	Analog luma or composite video output. An external resistor is connected between this node and the tv_vfb1 pin. The nominal value of Rout1 is 1650 Ω. Note that this is the output node that drives the load (75 Ω).
tv_out2	O	Analog chroma video output. An external resistor is connected between this node and the tv_vfb2 pin. The nominal value of Rout2 is 1650 Ω. This is the output node that drives the load (75 Ω).
tv_vfb1	O	Amplifier feedback node. An external resistor is connected between this node and tv_out1. The nominal value of Rout1 is 1650 Ω.

(1) O = Output, Power = Power pin

Table 12-17. TV Display Interface Pins (continued)

Pin Name	Type[1]	Description
tv_vfb2	O	Amplifier feedback node. An external resistor is connected between this node and tv_out2. The nominal value of Rout2 is 1650 Ω.
tv_vref	O	Reference output voltage from internal bandgap. A decoupling capacitor must be connected for optimum performance. tv_vref is generated internally to the OMAP device.
vdda_dac	Power	Analog supply voltage for the dual video DAC
vssa_dac	Power	Analog ground for the dual video DAC

CAUTION

- tv_out1 and tv_out2 are very high-frequency analog signals and must be routed with extreme care. As a result, the path of these signals must be as short as possible, and as isolated as possible from other interfering signals.
- During board design, the onboard traces and parasites must be matched for the two channels. tv_vfb1 and tv_vfb2 pins are the most sensitive pins in the TV out system. The onboard parasitic capacitance associated with these two pins should be less than 0.5 pF. Low onboard resistance is required for the traces that connect the Rout1/Rout2 to the tv_vfb1/tv_vfb2 and TV OUT pins (tv_out1 and tv_out2). The resistance on those trace affects output impedance matching. Therefore, Rout1 and Rout2 resistors are suggested to be placed as close as possible to the OMAP device pins. The onboard traces lead to the TV OUT pins must have a characteristic impedance of 75 Ω starting from the closest possible place to the OMAP device pin output. For typical values of Rout1, Rout2, Cout, and Cbg, see the device data manual.
- If the TV output is not used, the analog pins tv_ref and vdda_dac must be grounded. The tv_out1, tv_out2, tv_vfb1, and tv_vfb2 pins can be grounded or left unconnected. To avoid current leakage, the following bits must be set to 0:
  - DSS.DSS_CONTROL[5] DAC_POWERDN_BGZ
  - DSS.VENC_OUTPUT_CONTROL[2:0]
  - PRCM.CM_FCLKEN_DSS[2] EN_TV
  - CONTROL.CONTROL_DEVCONF[18] TVOUTBYPASS

Texas Instruments provides a global solution with an OMAP device associated with a power IC. The power pin vdda_dac is software controlled by the power IC.

12.2.4.1 TV Output and Data Format

The output data to the TV set are the analog composite data from the DAC. The following video standards are supported:

- NTSC-J, M
- PAL-B, D, G, H, I
- PAL-M

12.2.4.2 Digital-to-Analog Converter

The DAC includes the following main features:

- 1.0-V-to-1.3-V digital power supply, 1.8-V analog power supply
- 10-bit resolution
- DNL within 1 LSB and INL within 1 LSB

- Sample rate of up to 60 megasamples per second (MSPS)
- Support composite/S-video dc or ac coupled output
- Full-scale voltage output: 0.88 Vpp with a 75-Ω load
- Internal TV detect feature
- Signal-to-noise ratio (SNR) is 54 dB (taking into account the ac coupling)
- Suitable for low-power wireless applications; total power: 10 mA (no load in dc-coupled mode)
- Power-down mode with less than 2.5-µA standby current
- Differential gain error and differential phase error: 1.5% and 1%, respectively

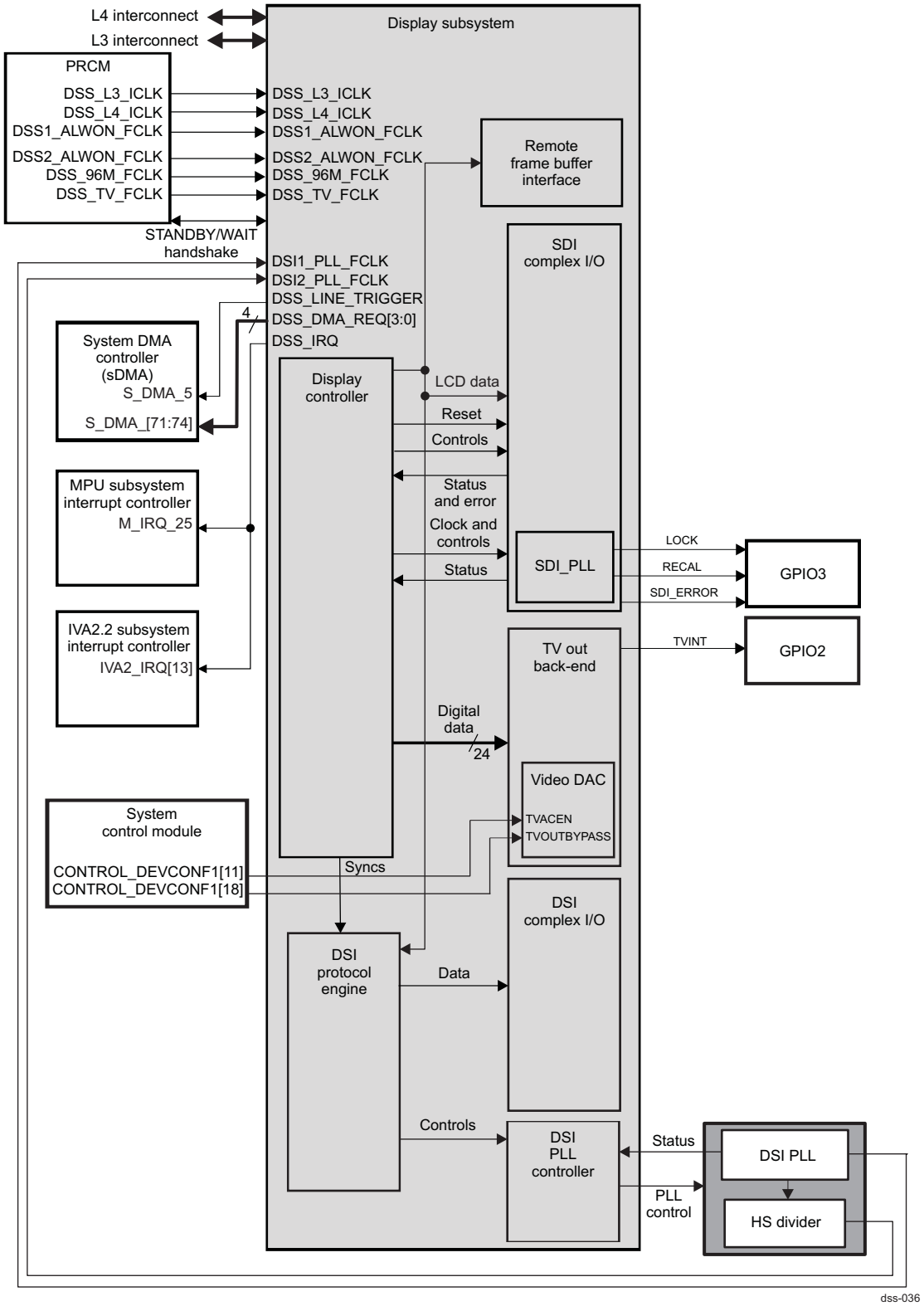
NOTE: To enhance the TV color display, it is highly recommended to set the DSS.DSS_CONTROL[4] DAC_DEMEN bit.

12.3 Display Subsystem Integration

This section describes the integration of the display subsystem and details clocks, resets, hardware requests, and power modes.

Figure 12-64 shows the integration of the display subsystem in the device.

Figure 12-64. Display Subsystem Integration



12.3.1 Clocking, Reset, and Power-Management Scheme

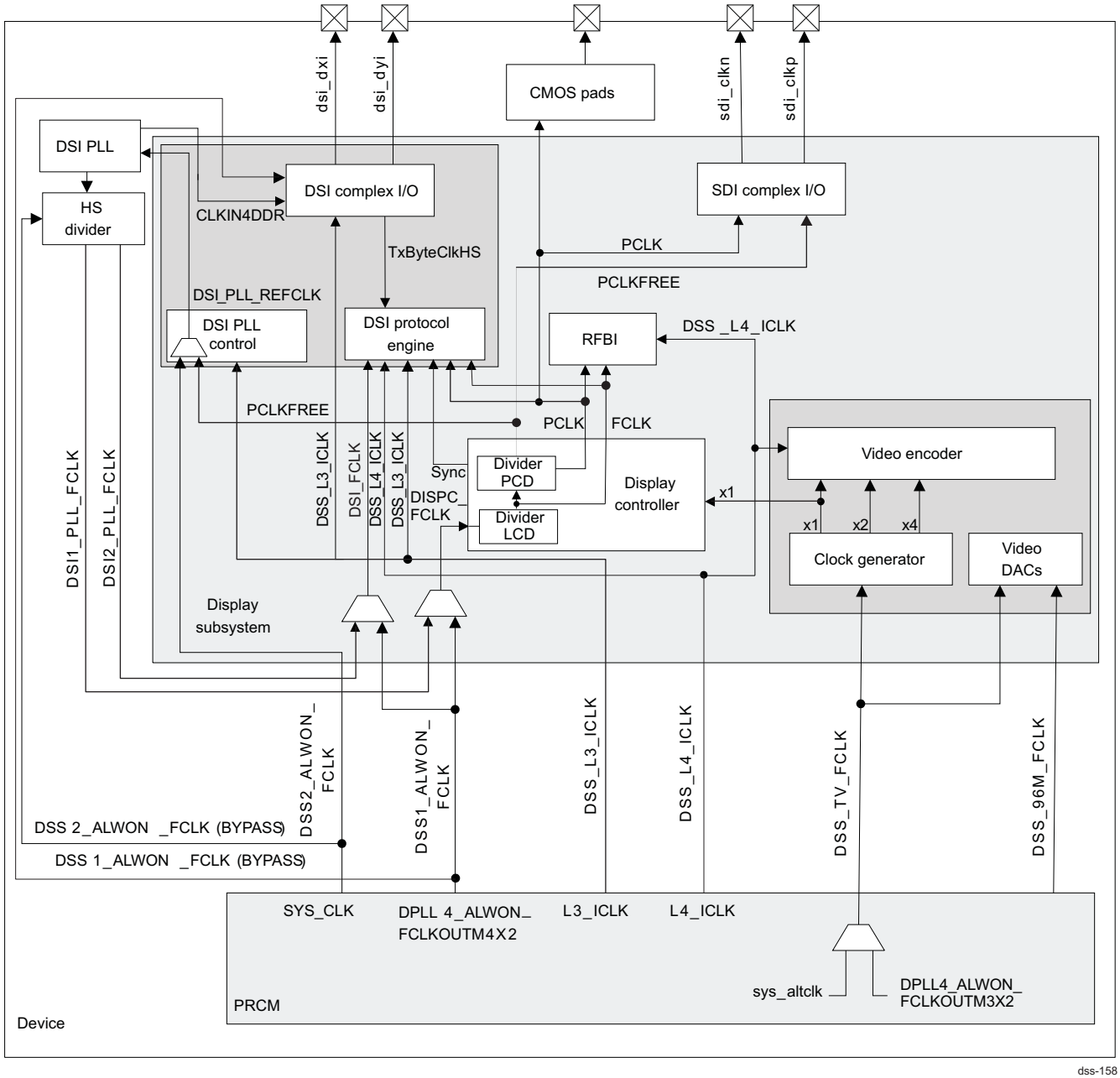
12.3.1.1 Clocks

The power, reset, and clock management (PRCM) module provides six clock signals to the display subsystem: The L3 interface clock (DSS_L3_ICLK) and the L4 interface clock (DSS_L4_ICLK), with frequencies equal to the L3 interconnect clock and the L4 interconnect clock, respectively; two configurable functional clocks (DSS1_ALWON_FCLK and DSS2_ALWON_FCLK); and two other functional clocks (DSS_TV_FCLK and DSS_96M_FCLK).

The DSI PLL provides two functional clock signals to the display subsystem: DSI1_PLL_FCLK and DSI2_PLL_FCLK.

Figure 12-65 details the clock tree for the display subsystem.

Figure 12-65. Display Subsystem Clock Tree



NOTE: A synchronization signal is sent by the display controller (DISPC) to the DSI protocol engine. This signal, named DISPC_UPDATE_SYNC, is used to inform the DSI protocol engine that it must be unsynchronized with the display controller.

Table 12-18 describes the different clocks with their possible frequency values.

Table 12-18. Display Subsystem Clocks

Clock Signal	Attribute	Module	Frequency	Comment
DSS_L3_ICLK	L3 interface clock	DSS	(See , *Power, Reset, and Clock Management*)	L3 interface clock from PRCM
DSS_L4_ICLK	L4 interface clock	DSS	(See , *Power, Reset, and Clock Management*)	L4 interface clock from PRCM
DSS1_ALWON_FCLK	Functional clock	DISPC, DSI protocol engine	Up to 173 MHz at nominal voltage (OPP3), and up to 96 MHz at low voltage (OPP2)	From PRCM: DPLL4 (source: DPLL4_ALWON_FCLK)
DSS2_ALWON_FCLK	Functional clock	DSI PLL	12/13/16.8/19.2/26/38.4 MHz	From PRCM: SYS_CLK
DSI1_PLL_FCLK	Functional clock	DISPC	Up to 173 MHz at nominal voltage (OPP3), and up to 96 MHz at low voltage (OPP2)	From DSI PLL and HS divider
DSI2_PLL_FCLK	Functional clock	DSI protocol engine	Up to 173 MHz at nominal voltage (OPP3), and up to 96 MHz at low voltage (OPP2)	From DSI PLL and HS divider
DSS_TV_FCLK	Functional clock	DSS, video mode DAC	54 MHz or	From PRCM: DPLL4 (source: DPLL4_ALWON_FCLK) or
			sys_alt_clk (up to 59 MHz)	External input clock (See , *PRCM*)
DSS_96M_FCLK	Functional clock	Video DAC	96 MHz	From PRCM: 96M_FCLK

To enable or disable each functional clock, set the following bit (1: Enable, 0: Disable):

- PRCM.CM_FCLKEN_DSS[0] EN_DSS1 bit to enable DSS1_ALWON_FCLK
- PRCM.CM_FCLKEN_DSS[1] EN_DSS2 bit to enable DSS2_ALWON_FCLK
- PRCM.CM_FCLKEN_DSS[2] EN_TV bit to enable DSS_TV_FCLK and DSS_96M_FCLK

To enable or disable the DSS_L3_ICLK and DSS_L4_ICLK interface clocks, write (1: Enable, 0: Disable) to the PRCM.CM_ICLKEN_DSS[0] EN_DSS bit.

NOTE: Note that it is not possible to gate/stop L3 clock and keep L4 clock running.

- L3 and L4 interface clock

  The DSS_L3_ICLK clock is only used by the display controller interface to fetch the pixel data. The DSS_L4_ICLK is used to access the L4 interconnect for configuring all the display subsystem registers.

  NOTE: A clock generated internally from the L3 interface clock allows the submodules to be configured and is the functional clock for the RFBI. The SDI module and all the display subsystem registers are configured through the display subsystem register.

- Display controller functional clocks

  The display controller can use either the DSS1_ALWON_FCLK or the DSI1_PLL_FCLK functional clock.

To select the DSS1_ALWON_FCLK functional clock (the default clock selected after reset), write 0 in the DSS.DSS_CONTROL[0] DISPC_CLK_SWITCH bit; to select the DSI1_PLL_FCLK functional clock, write 1 in the DSS.DSS_CONTROL[0] DISPC_CLK_SWITCH bit.

NOTE: The DSS1_ALWON_FCLK and DSI1_PLL_FCLK functional clocks must be active (the PRCM.CM_FCLKEN_DSS[0] EN_DSS1 and DSI PLL programmed correctly) to switch from one functional clock to another. The new functional clock is effective when the next vertical blanking interval occurs. This is true only if the DSS.DISPC_CONTROL[5] GOLCD bit is set to 1.

Depending on the DPLL4 input clock frequency, the DSS1_ALWON_FCLK can be adjusted by setting the PRCM.CM_CLKSEL_DSS[4:0] CLKSEL_DSS1 bit field.

- DSI PLL functional clock (DSI_PLL_REFCLK)

  The DSI PLL controller module can use either the DSS2_ALWON_FCLK (from PRCM) or the PCLKFREE (from DISPC) functional clock. To select the DSS2_ALWON_FCLK functional clock (default clock selected after reset), write 0 in the DSS.DSI_PLL_CONFIGURATION2[11] DSI_PLL_CLKSEL bit; to select the PCLKFREE functional clock, write 1 in the DSS.DSI_PLL_CONFIGURATION2[11] DSI_PLL_CLKSEL bit.

- DSI protocol engine functional clocks (DSI_FCLK)

  The DSI protocol engine can use either the DSS1_ALWON_FCLK (from PRCM) or the DSI2_PLL_FCLK (from DSI PLL) functional clock. To select the DSS1_ALWON_FCLK functional clock (default clock selected after reset), write 0 in the DSS.DSS_CONTROL[1] DSI_CLK_SWITCH bit; to select the DSI2_PLL_FCLK functional clock, write 1 in the DSS.DSS_CONTROL[1] DSI_CLK_SWITCH bit.

  NOTE: It is possible to switch between these two clocks, even when both of them are not active.

- There are five clock domains in the DSI module:
  - Byte clock domain:

    TxByteClkHS is generated from the bit clock and converted into a byte clock. The maximum frequency is 100 MHz (all OPPs). It is generated by the DSI complex I/O.
  - Functional clock domain

    The DSI_FCLK is the functional clock for the DSI protocol engine module. The maximum frequency is 173 MHz (nominal voltage) and 96 MHz (low voltage). It should always be equal to or higher than the byte (TxByteClkHS), L4 interconnect (DSS_L4_ICLK), and video port (VP_CLK) clocks. The software must configure the clocks correctly.
  - L4 interface clock domain

    The DSS_L4_ICLK is used in the L4 interconnect port domain. The maximum frequency is 166 MHz at nominal voltage and 83 MHz at low voltage.
  - The video port domain

    The pixel clock (PCLK) on the video port is used by the video port domain to capture the pixels from the display controller. The maximum frequency of VP_CLK used as the functional clock for the video port domain is 173 MHz at nominal voltage and 96 MHz at low voltage.
  - Serial Configuration Port (SCP) and Power Control (PWR) interfaces

    The DSS_L4_ICLK is the functional clock.

NOTE:

- There is no clock domain for RxClkEsc because it is used as an enable and not as a clock by the DSI protocol engine module
- The clock domains are asynchronous (except for L4 interconnect port and SCP/PWR because both of them use DSS_L4_ICLK). The clocks used for the L4 interconnect port and SCP/PWR interface should be balanced.
- If video mode is used, the display controller functional clock must be generated using a clock from the DSI PLL.

- Video encoder functional clock

  The DSS_TV_FCLK is divided into three balanced clocks, depending on the clock mode selected (see Table 12-19).

Table 12-19. Possible Digital Clock Division for the Video Encoder

Clock Output	Clock Mode	
	Clock mode 0	Clock mode 1
Video encoder clock 4x	DSS_TV_FCLK	DSS_TV_FCLK or 0 (gated)
Video encoder clock 2x	DSS_TV_FCLK/2	DSS_TV_FCLK
Video encoder clock 1x	DSS_TV_FCLK/4	DSS_TV_FCLK/2

The clock mode is defined by the the DSS_CONTROL[2] VENC_CLOCK_MODE bit:

- In the case of clock mode 1, the DSS_CONTROL[3] VENC_CLOCK_4X_ENABLE bit is used to control clock gating.
- In the case of clock mode 0, the VENC_CLOCK_4X_ENABLE bit must be set to 0x1 by software.

NOTE: After reset, clock mode 0 is selected by default and the DSS_TV_CLK clock is disabled. The DSS_TV_CLK / 4 in mode 0, or the DSS_TV_CLK / 2 in mode 1, is used in the DISPC module to send data to the video encoder.

NOTE: Clock mode 1 can be used for power-saving purposes, or if a 27-MHz external clock is provided to the video encoder.

DSS_TV_FCLK can be adjusted depending on the DPLL4 input clock frequency by setting the PRCM.CM_CLKSEL_DSS[12:8] CLKSEL_TV bit field. If the DPLL4 is selected, the DSS_TV_FCLK is provided by the DPLL4_ALWON_FCLKOUTM3X2 clock.

NOTE: If the DSS_TV_FCLK is not provided by DPLL4 but rather by the sys_alt_clk pin, an external clock generator must be connected to this pin. In this case, a 54-MHz clock is needed for PAL or NTSC 601, a 49.09-MHz clock is needed for NTSC square pixel, and a 59-MHz clock is needed for PAL square pixel.

- Dual video DAC clocks

  The dual video DAC uses two distinct clocks: The DSS_TV_FCLK and the DSS_96M_FCLK. The video data are latched on the positive edge of the DSS_TV_FCLK clock. On the other hand, the video DAC uses the DSS_96M_FCLK fixed-frequency clock internally as the clock input for the switch capacitor resistor.

- SDI functional clock

  The display controller provides the functional clocks PCLK (pixel clock) and PCLKFREE (free-running pixel clock) of the SDI module, which is derived from the display controller functional clock. The SDI module functional clock drives the phase-locked loop SDI_PLL that multiplies and divides the PCLK frequency by two programmable factors for the serial connection.

NOTE: When using the SDI, configure the PCLK in the free-running mode by setting the DSS.DISPC_CONTROL[27] PCLKFREEENABLE bit to 1.

12.3.1.2 Resets

12.3.1.2.1 Hardware Reset

The display subsystem receives its reset signal DSS_RST (the reset signal of the display subsystem [DSS] power domain) from the PRCM module.

12.3.1.2.2 Software Reset

The display subsystem can receive a software reset propagated through all of the submodules and used to initialize the display subsystem. To apply the reset, write to the DSS.DSS_SYSCONFIG[1] SOFTRESET bit (1: Reset; 0: Normal). The DSS.DSS_SYSSTATUS[0] RESETDONE bit indicates that the software reset is complete when its value is 1.

NOTE: The display controller, the DSI protocol engine, and the RFBI modules also have their own software reset functionality. To access this reset, access the DSS.DISPC_SYSCONFIG[1] SOFTRESET bit for the display controller, the DSS.DSI_SYSCONFIG[1] SOFTRESET bit for the DSI protocol engine, and the DSS.RFBI_SYSCONFIG[1] SOFTRESET bit for the RFBI module.

To properly reset these modules, 0x2 is the only valid value to write to these registers.

CAUTION

All the interface and functional clocks, even for the TV output, must be provided to the display subsystem to update the RESETDONE status bit correctly.

12.3.1.3 Power Domain

The display subsystem modules are on the display subsystem (DSS) power domain and on the VDD2 voltage domain, except for the dual video DACs, which are on the analog vdda_dac voltage domain.

12.3.1.4 Power Management

12.3.1.4.1 Clock Activity Mode

The display controller clocks can be configured in one of the following clock activity modes:

- DSS.DISPC_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x0 (reset value): The interface and functional clocks can be switched off.
- DSS.DISPC_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x1: The functional clocks can be can be switched off and the interface clocks are maintained during the wake-up period.
- DSS.DISPC_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x2: The interface clocks can be can be switched off and the functional clocks are maintained during the wake-up period.
- DSS.DISPC_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x3: The interface and functional clocks are maintained during the wake-up period.

The DSI protocol engine clocks can be configured in one of the following clock activity modes:

- DSS.DSI_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x0 (reset value): The interface and functional clocks can be switched off.
- DSS.DSI_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x1: The functional clocks can be switched off and the interface clocks are maintained during the wake-up period.
- DSS.DSI_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x2: The interface clocks can be switched

off and the functional clocks are maintained during the wake-up period.

- DSS.DISPC_SYSCONFIG[9:8] CLOCKACTIVITY bit field set to 0x3: The interface and functional clocks are maintained during the wake-up period.

The DSS power domain clock activity status is logged in the PRCM.CM_CLKSTST_DSS[0] CLKACTIVITY_DSS status bit. When set to 0, there is no domain clock activity. When set to 1, the DSS power domain clock is active.

NOTE: The display subsystem interface clock can be dependent on the DSS power domain state. This is configured with PRCM.CM_AUTOIDLE_DSS[0] AUTO_DSS bit:

- When the AUTO_DSS bit is set to 0 (reset value): The display subsystem interface clock is not related to the DSS power domain state transition.
- When the AUTO_DSS bit is set to 1: The display subsystem interface clock is automatically enabled or disabled along with the DSS power domain state transition.

12.3.1.4.2 Autoidle Mode

The RFBI, display controller, DSI protocol engine, and L4 interfaces can internally gate their clocks to decrease power consumption if no transaction is present on the related bus. The following bits must be set to enable this functionality:

- DSS.DSS_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the display subsystem
- DSS.RFBI_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the RFBI
- DSS.DISPC_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the display controller
- DSS.DSI_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the DSI protocol engine
- DSS.DISPC_CONFIG[9] FUNCGATED bit (1: Functional clocks gated enabled; 0: Functional clocks gated disabled) for the display controller

NOTE: All the bits listed above (except for the FUNCGATED bit) are set to 1 by default. It is highly recommended to set all the bits to 1 to save power.

12.3.1.4.3 Idle Mode

The display controller, DSI protocol engine, and RFBI can be configured into one of the following acknowledgment modes:

- Force-idle mode: The module immediately enters the idle state on receiving a low-power mode request from the PRCM module. In this mode, the software must ensure that there are no asserted output interrupts before requesting this mode to go into the idle state. Set the DSS.DISPC_SYSCONFIG[4:3] SIDLEMODE bit field to 0x0 (reset value) for display controller, set the DSS.DSI_SYSCONFIG[4:3] SIDLEMODE bit field to 0x0 (reset value) for DSI protocol engine, and, finally, the DSS.RFBI_SYSCONFIG[4:3] SIDLEMODE bit field to 0x0 (reset value) for RFBI.
- No-idle mode: The module never enters the idle state. Set the DSS.DISPC_SYSCONFIG[4:3] SIDLEMODE bit field to 0x1 for display controller, set the DSS.DSI_SYSCONFIG[4:3] SIDLEMODE bit field to 0x1 for DSI protocol engine, and, finally, the DSS.RFBI_SYSCONFIG[4:3] SIDLEMODE bit field to 0x1 for RFBI.
- Smart-idle mode:
  - Display controller: After receiving a low-power-mode request from the PRCM module, the display controller module enters the idle state when all the following conditions are satisfied:
    - All asserted output interrupts are acknowledged (no interrupt pending).
    - The display controller does not use anymore the L4 interface clock (DSS_L4_ICLK).
  - DSI protocol engine: After receiving a low-power-mode request from the PRCM module, the DSI protocol engine enters the idle state when all the following conditions are satisfied:
    - All asserted output interrupts are acknowledged (no interrupt pending).

- The DSI protocol engine does not use the L4 interface clock (DSS_L4_ICLK) anymore.
- The SCP and PWR transactions are complete.
- No data remains in the TX FIFO (data waiting in the FIFO to be sent to the peripheral).

To configure the display subsystem in smart-idle mode, set the DSS.DISPC_SYSCONFIG[4:3] SIDLEMODE bit field to 0x2 for display controller, set the DSS.DSI_SYSCONFIG[4:3] SIDLEMODE bit field to 0x2 for DSI protocol engine, and, finally, the DSS.RFBI_SYSCONFIG[4:3] SIDLEMODE bit field to 0x2 for RFBI.

Once the idle handshake protocol is over:

- The DSS L4 interface clock (DSS_L4_ICLK) can be shutdown at any time.
- Any transaction on the L4 configuration port is ignored.

12.3.1.4.4 SDI Idle Mode

To save power consumption, the PLL (SDI_PLL) of the SDI module can be configured in one of the operation modes shown in Table 12-20 when not locked.

Table 12-20. SDI PLL Operation Modes

SDI_PLL Operation Mode	Stop mode Low power[1]	Stop mode Low power[1]	Idle bypass Low power	Idle bypass Fast relock	Limp
	Output clock stopped Lowest power standby	Output clock stopped Fastest start-up time	Output clock at PCLK rate Lowest power standby	Output clock at PCLK rate Fastest start-up time	Output at full speed
DSS.DSS_PLL_CONTROL[0] SDI_PLL_IDLE	0	0	1	1	1 or PCLK stops
DSS.DSS_PLL_CONTROL[20] SDI_PLL_LOWCURRSTBY	1	0	1	0	X
DSS.DSS_PLL_CONTROL[17] SDI_PLL_STOPMODE	1	1	1	1	0

(1) Recommended

12.3.1.4.5 Wake-Up Mode

The Display Controller (DISPC) supports the wake-up protocol. The mode is selected by programming the appropriate value in the DSS.DISPC_SYSCONFIG[2] ENWAKEUP bit. The wake-up signal is asserted when the DISPC is in idle mode and when anyone of the following events occur:

- Graphics pipe is enabled and data fetch is not completed for graphics window, and the number of data bytes in FIFO is less than the low threshold programmed value.
- Video1 pipe is enabled and data fetch is not completed for video1 window, and the number of data bytes in FIFO is less than the low threshold programmed value.
- Video2 pipe is enabled and data fetch is not completed for video2 window, and the number of data bytes in FIFO is less than the low threshold programmed value.
- The current pixel is the last pixel displayed on the LCD panel if it is not the last frame.
- The current pixel is the last pixel displayed on the digital panel if it is not the last frame.

  If software users set the DSS.DISPC_CONFIG[17] FIFOFILLING bit, when one of the active pipe reaches the low threshold and should refill the FIFO for the current frame, the other pipes also refill their own FIFOs, even if the low threshold has not been reached. This is used to improve the probability of increasing the time when there is no access to the L3 interconnect (MStandby asserted, affects power savings).

  Once the wake-up signal is asserted, the WAKEUP interrupt request is generated. The wake-up signal is deasserted when the idle request is no longer activated.

12.3.1.4.6 Standby Mode

As part of the system-wide power-management scheme, the display controller can enter standby mode. To configure the display controller, write the DSS.DISPC_SYSCONFIG[13:12] MIDLEMODE bit field (00: Forced standby; 01: No standby; 10: Smart standby) in one of the following standby modes:

- Forced standby mode (default mode): The module enters standby mode when the module is disabled.
- No standby mode: The module never enters standby mode.
- Smart standby mode: The module enters standby state when the DISPC module is disabled or when all the three following events occur:
  - Graphics pipe is disabled or graphics pipe is enabled but data fetch completed for graphics window, or graphics pipe is enabled and data fetch is not completed and number of data bytes in FIFO is greater than the high threshold programmed value.
  - Video1 pipe is disabled or video1 pipe is enabled but data fetch completed for video1 window, or video1 pipe is enabled and data fetch is not completed and number of data bytes in FIFO is greater than the high threshold programmed value.
  - Video2 pipe is disabled or video2 pipe is enabled but data fetch completed for video2 window, or video2 pipe is enabled and data fetch is not completed and number of data bytes in FIFO is greater than the high threshold programmed value.

When in standby mode, the display controller does not generate transactions on the L3 master port. Standby is active when the PRCM module confirms this mode.

The display subsystem standby mode activity can be monitored with the PRCM.CM_IDLEST_DSS[0] ST_DSS status register. When this register is read to 0, the display subsystem is accessible and the interface clock running; when it is read to 1, the display subsystem is in standby mode.

12.3.1.4.6.1 Conditions to Exit Standby Mode

The following conditions allow the subsystem to exit standby mode:

- Forced standby mode: Standby mode is exited when the display controller is enabled.
- Smart standby mode: Standby mode is exited when any one of the following events occurs:
  - Graphics pipe is enabled and data fetch is not completed for graphics window, and number of data bytes in FIFO is less than the low threshold programmed value.
  - Video1 pipe is enabled and data fetch is not completed for video1 window, and number of data bytes in FIFO is less than the low threshold programmed value.
  - Video2 pipe is enabled and data fetch is not completed for video2 window, and number of data bytes in FIFO is less than the low threshold programmed value.

12.3.1.4.6.2 Standby Transition Dependency

The sleep transition of the DSS power domain can be dependent or not with respect to MPU domain. This is configured by PRCM.CM_SLEEPDEP_DSS[1] EN_MPU bit:

- When the EN_MPU bit is set to 0 (reset value): The DSS power domain sleep dependency with the MPU power domain is disabled. The DSS power domain will not enter idle unless the MPU power has previously entered idle.
- When the EN_MPU bit is set to 1: The DSS power domain sleep dependency with the MPU power domain is enabled. The DSS power domain will enter idle regardless of the power domain state of the MPU.

The sleep transition of the DSS power domain may, or may not depend on the IVA2.2 domain, depending on the configuration of the PRCM.CM_SLEEPDEP_DSS[2] EN_IVA2 bit:

- When the EN_IVA2 bit is set to 0 (reset value): The DSS power domain sleep dependency on the IVA2.2 power domain is disabled.
- When the EN_IVA2 bit is set to 1: The DSS power domain sleep dependency on the IVA2.2 power domain is enabled.

12.3.1.4.6.3 Standby Procedure Description

When the display subsystem initiates a standby procedure, it also initiates an standby/wait handshake protocol with the PRCM module that lets the PRCM cut the display subsystem clocks. Depending on the PRCM setting, two modes are available:

- Manual mode
  - DSS1_ALWON_FCLK is shut down when the PRCM.CM_FCLKEN_DSS[0] EN_DSS1 bit is set to 0 and the display subsystem is in standby mode.
  - DSS2_ALWON_FCLK is shut down when the PRCM.CM_FCLKEN_DSS[1] EN_DSS2 bit is set to 0 and the display subsystem is in standby mode.
  - DSS_L3_ICLK and DSS_L4_ICLK are controlled together. They are shut down when the PRCM.CM_ICLKEN_DSS[0] EN_DSS bit is set to 0 and the display subsystem is in standby mode.

CAUTION

Do not stop DSS1_ALWON_FCLK, or DSS2_ALWON_FCLK clock (if used) if the display subsystem is not disabled.

CAUTION

DSS_TV_FCLK does not depend on the display subsystem standby state. Ensure correct clock management for DSS_TV_FCLK.

The clocks are reactivated when the related bits are set to 1 and the display subsystem exits from standby. For more information, see , *Power, Reset, and Clock Management*.

- Hardware- or software-supervised mode

  The DSS-power state-transition between active and inactive states can be either hardware or software supervised. This is programmed with the PRCM.CM_CLKSTCTRL_DSS[1:0] CLKTRCTRL_DSS bit field:
  - When the CLKTRCTRL_DSS bit field is set to 0x0 (reset value), the automatic transition is disabled
  - When the CLKTRCTRL_DSS bit field is set to 0x1, the software-supervised sleep transition is started on the DSS power domain.
  - When the CLKTRCTRL_DSS bit field is set to 0x2, the software-supervised wake-up transition is started on the DSS power domain.
  - When the CLKTRCTRL_DSS bit field is set to 0x3, the automatic transition is enabled. Any transition on the DSS power domain is supervised by the hardware.

12.3.1.4.6.4 Display Subsystem Standby Mode, Power-Saving Use Cases

- Setup
  - Set the display subsystem in smart standby mode.
  - Manually enable DSS_L3_ICLK and DSS_L4_ICLK.
  - Manually enable DSS1_ALWON_FCLK or DSS2_ALWON_FCLK.
  - Manually enable DSS_TV_FCLK if the video encoder is used.
  - Set the PRCM.CM_CLKSTCTRL_DSS[1:0] CLKTRCTRL_DSS bit field to 0x3 (autocontrol mode supervised by hardware).
- Shut down the display subsystem.
  - Disable the display subsystem.
  - Manually disable DSS1_ALWON_FCLK, DSS2_ALWON_FCLK, DSS_TV_FCLK, DSS_L3_ICLK, and DSS_L4_ICLK.

For more details on low-power programming settings, see Section 12.6.2.

12.3.2 Hardware Requests

12.3.2.1 DMA Requests

The display controller, the DSI protocol engine, and the RFBI generate some DMA requests to the sDMA. Figure 12-66 details the DMA tree.

Figure 12-66. Display Subsystem DMA Tree

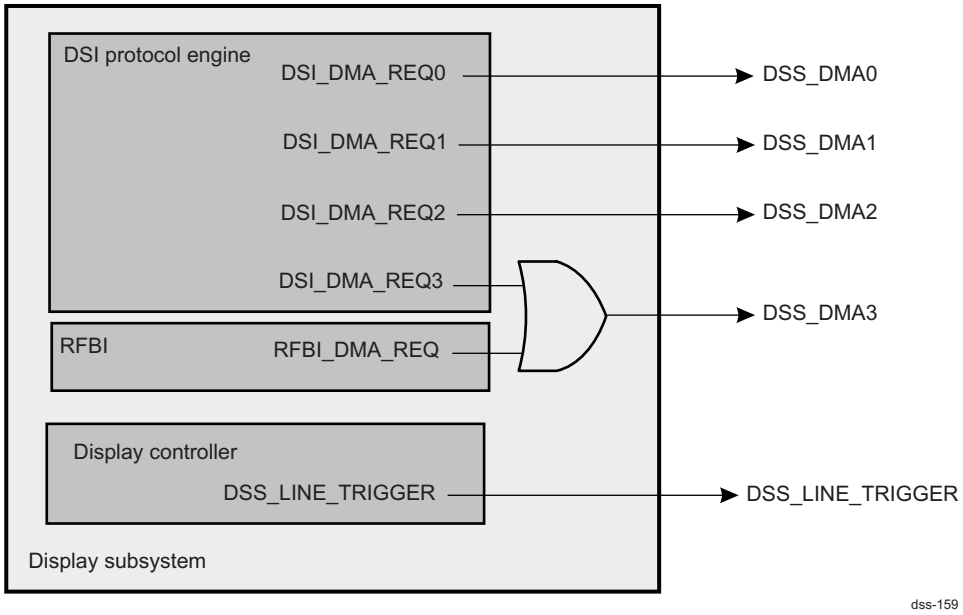


Table 12-21. DSS DMA Requests Description

DMA Request Name	DSS Module	Mapping	Description
DSS_LINE_TRIGGER	DISPC	S_DMA_5	See Section 12.3.2.1.1
DSI_DMA_REQ0	DSI protocol engine	S_DMA_71	See Section 12.3.2.1.2
DSI_DMA_REQ1	DSI protocol engine	S_DMA_72	See Section 12.3.2.1.2
DSI_DMA_REQ2	DSI protocol engine	S_DMA_73	See Section 12.3.2.1.2
DSI_DMA_REQ3	DSI protocol engine	S_DMA_74	See Section 12.3.2.1.2
RFBI_DMA_REQ	RFBI	S_DMA_74	See Section 12.3.2.1.3

NOTE: The DMA requests from the RFBI module (RFBI_DMA_REQ) and the DSI protocol engine (DSI_DMA_REQ3) are merged on line DSS_DMA3. The software must only use DSS_DMA3 on one module at a time (RFBI or DSI protocol engine).

12.3.2.1.1 Display Controller DMA Request (Line Trigger)

One DMA synchronization line (DSS_LINE_TRIGGER) is connected to the sDMA by the sDMA controller (S_DMA_5) input line. This DMA request is not a classical one but a synchronization signal from the display subsystem to the sDMA informing the sDMA that a programmable number of lines are output to the LCD, and that the system memory can be updated. This request is related to an interrupt event described in Section 12.3.2.2, *Interrupt Requests*. This allows the sDMA channel to be synchronized with the display subsystem internal DMA controller. In other words, it allows to synchronize a memory to memory frame buffer update based on the scan line of the frame buffer in system memory (SDRAM or SRAM) by the display controller. The DSS_LINE_TRIGGER DMA request is generated at a programmable line number defined in DSS.DISPC_LINE_NUMBER[10:0] LINENUMBER bit field.

12.3.2.1.2 DSI Protocol Engine DMA Request

The DSI DMA requests are used to allow automatic transfer by the sDMA or MPU (with less efficiency and through-put capability) from the DSI RX FIFO to the system memory and from the system memory to the DSI TX FIFO. Two independent DMA requests for RX FIFO and TX FIFO for the same VC are supported.

12.3.2.1.3 RFBI DMA Request

The RFBI_DMA_REQ is used to receive data into the RFBI FIFO. The DMA request is always generated when there is enough room in the FIFO to accept the full burst.

12.3.2.2 Interrupt Requests

The DSI protocol engine, the DSI complex I/O (DSI_IRQ), and the display controller (DISPC_IRQ) generate one interrupt request each. The DSI_IRQ and DISPC_IRQ lines are merged together in a single interrupt line.

One interrupt line (DSS_IRQ) is connected to two interrupt controllers:

- MPU interrupt controller (M_IRQ_25 input line)
- IVA interrupt handler (IVA2_IRQ[13] input line)

Figure 12-67 shows the interrupt tree for the DSI protocol engine and DSI complex I/O in detail.

Figure 12-67. DSI Interrupt Tree

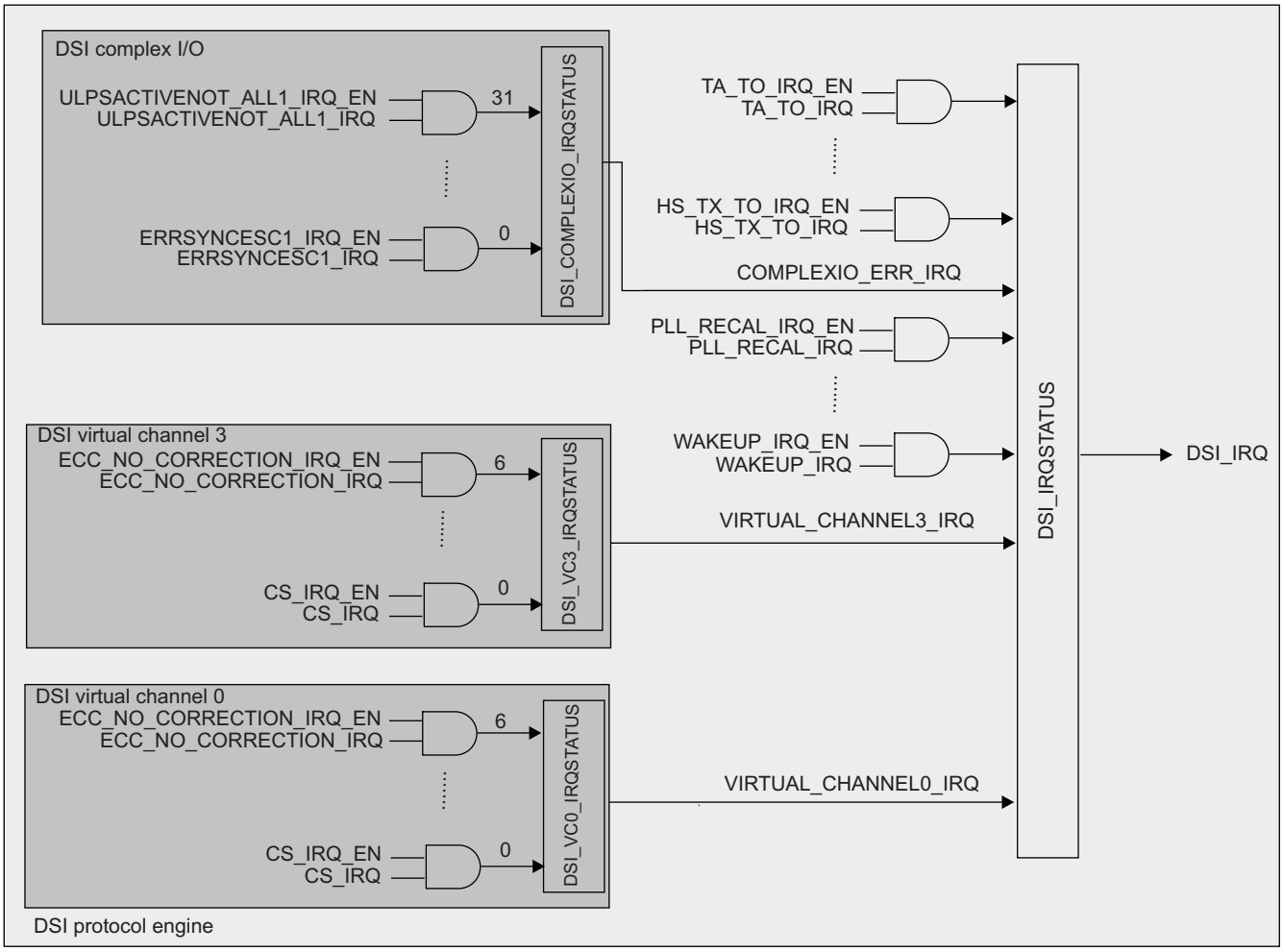
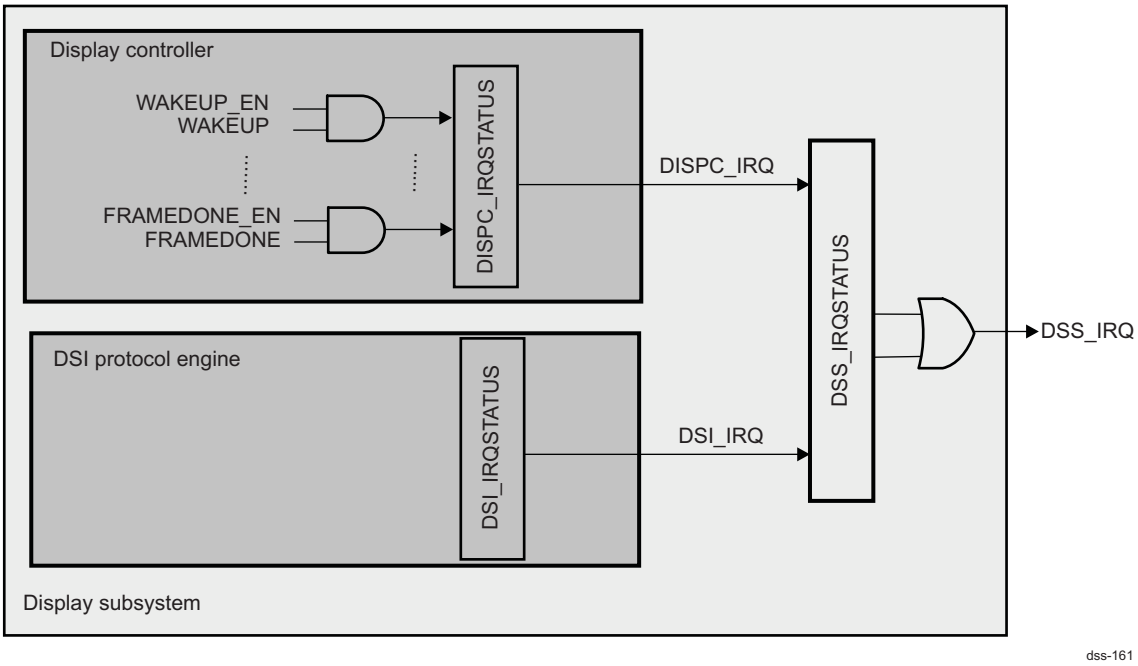


Figure 12-68 details the interrupt tree for the DISPC and the display subsystem.

Figure 12-68. DISPC and DSS Interrupts Tree



12.3.2.2.1 DISPC Interrupt Request

The interrupt line indicates when one or more events are detected by the hardware. Each event is independently maskable by setting the DSS.DISPC_IRQENABLE register.

To check when a particular interrupt event occurs and to reset a particular event, the DSS.DISPC_IRQSTATUS register must be accessed. This register regroups all the status of the module internal events that generate an interrupt (read 0: No interrupt occurred; read 1: Interrupt occurred; write 1: Status bit reset). See Section 12.7, *Display Subsystem Register Manual*, for more information on checking and clearing interrupt events.

Table 12-22 lists the display subsystem interrupt events.

Table 12-22. Display Subsystem Interrupts

Interrupt Name	Description
FRAMEDONE	Active frame is complete and LCD output is disabled.
VSYNC	VSYNC interrupt occurred at the end of the frame.
EVSYNC_EVEN[(1)]	EVSYNC_EVEN interrupt occurred at the end of the frame. (EVSYNC is received and the field polarity is even.)
EVSYNC_ODD[(1)]	EVSYNC_ODD interrupt occurred at the end of the frame. (EVSYNC is received and the field polarity is odd.)
ACBIASCOUNTSTATUS	The ac-bias transition counter decremented to 0.
PROGRAMMEDLINENUMBER	The LCD reached the user-programmed line number.
GFXFIFOUNDERFLOW	The input graphics FIFO goes underflow.
GFXENDWINDOW	The screen reached the end of the graphics window. All data for the graphics window are fetched from memory and displayed on the screen.
PALETTEGAMMALOADING	The palette/gamma table is loaded.
OCPERROR	L3 interconnect sent SResp = ERR.
VID1FIFOUNDERFLOW	The input video1 FIFO goes underflow.
VID1ENDWINDOW	The screen reached the end of video1 window. All data for the video window are fetched from the memory and displayed on the screen.

(1) EVYNC interrupts (EVSYNC_EVEN and EVSYNC_ODD) are external interrupts received by the display controller and generated by the video encoder (VENC) module.

Table 12-22. Display Subsystem Interrupts (continued)

Interrupt Name	Description
VID2FIFOUNDERFLOW	The input video2 FIFO goes underflow.
VID2ENDWINDOW	The screen reached the end of video2 window. All data for the video window are fetched from the memory and displayed on the screen.
SYNCLOST	Interrupt occurs when VSYNC width/front or back porches are not wide enough to load the pipelines with data (LCD output).
SYNCLOSTDIGITAL	Interrupt occurs when the display controller is not ready to output data when a digital request occurs. This interrupt informs that the timings of the NTSC/PAL video encoder are not set correctly.
WAKEUP	Occurs when the wakeup signal is asserted

NOTE: To clear a synchronization lost interrupt, follow this sequence:

1. Clear the DSS.DISPC_CONTROL[0] LCDENABLE (LCD: SYNCLOST interrupt) or DSS.DISPC_CONTROL[1] DIGITALENABLE (TV: SYNCLOSTDIGITAL interrupt) bits.

   Check the interrupts.

   LCD: Verify that a FRAMEDONE interrupt occurs.

   TV : Verify that EVSYNC_EVEN or EVSYNC_ODD interrupts occur.
2. Set the DSS.DSS_SYSCONFIG[1] SOFTRESET bit to reset the display subsystem.
3. Set the display subsystem registers again.

NOTE: The SYNCLOSTDIGITAL interrupts, which occur before the first VSYNC pulse signal (from the video encoder), should not be considered.

After the first VSYNC pulse signal, the SYNCLOSTDIGITAL interrupt status bit must be cleared by writing 1 in the DSS.DISPC_IRQSTATUS[15] SYNCLOSTDIGITAL bit; then the SYNCLOSTDIGITAL interrupt can be enabled by setting the DSS.DISPC_IRQENABLE[15] SYNCLOSTDIGITAL bit.

12.3.2.2.2 DSI Interrupt Request

The DSI protocol engine requires a single interrupt line, DSI_IRQ. The DSS.DSI_IRQSTATUS register indicates the general interrupt events. Refer to Table 12-23. Each VC and complex I/O has a dedicated interrupt register: DSS.DSI_VCn_IRQSTATUS and DSS.DSI_COMPLEXIO_IRQSTATUS respectively. Refer to Table 12-24 and Table 12-25.

Table 12-23 indicates the DSI global interrupt events.

Table 12-23. DSI Global Interrupts

Interrupt Name	Description
RESYNCHRONIZATION_IRQ	Resynchronization in video mode
TA_TO_IRQ	Turn-around timer expired
LDO_POWER_GOOD_IRQ	Signal LDOPWRGOOD from the DSI_PHY changes its state for the supply VDDALDODSIPLL from up to down or down to up.
SYNC_LOST_IRQ	Synchronization with video mode port is lost (video mode only)
ACK_TRIGGER_IRQ	Acknowledge trigger is received
TE_TRIGGER_IRQ	Tearing effect trigger is received
WAKEUP_IRQ	Occurs when the SWakeUp signal is asserted
HS_TX_TO_IRQ	High speed TX Time-out Interrupt
LP_RX_TO_IRQ	Low speed RX Time-out Interrupt

Table 12-23. DSI Global Interrupts (continued)

Interrupt Name	Description
COMPLEXIO_ERR_IRQ	Error signaling from complex I/O: The interrupt is triggered when any error is received from the complex I/O (events are defined in DSI_COMPLEXIO_IRQSTATUS).
PLL_RECAL_IRQ	PLL recal event (assertion of DSIRecal signal from the DSI PLL Control module)
PLL_UNLOCK_IRQ	PLL unlock event (deassertion of DSILock signal from the DSI PLL Control module)
PLL_LOCK_IRQ	PLL lock event (assertion of DSILock signal from the DSI PLL Control module)
VIRTUAL_CHANNEL3_IRQ	Virtual channel #3 Error signaling from DSI Virtual Channel3: The interrupt is triggered when an error is received from DSI Virtual Channel3 (events are defined in DSI_VC3_IRQENABLE).
VIRTUAL_CHANNEL2_IRQ	Virtual channel #2 Error signaling from DSI Virtual Channel2: The interrupt is triggered when an error is received from DSI Virtual Channel2 (events are defined in DSI_VC2_IRQENABLE).
VIRTUAL_CHANNEL1_IRQ	Virtual channel #1 Error signaling from DSI Virtual Channel1: The interrupt is triggered when an error is received from DSI Virtual Channel1 (events are defined in DSI_VC1_IRQENABLE).
VIRTUAL_CHANNEL0_IRQ	Virtual channel #0 Error signaling from DSI Virtual Channel0: The interrupt is triggered when an error is received from DSI Virtual Channel0 (events are defined in DSI_VC0_IRQENABLE).

Table 12-24 indicates the DSI complex I/O interrupt events.

Table 12-24. DSI Complex I/O Interrupts

Interrupt Name	Description
ULPSActiveNot_ALL0_IRQ	All signals ULPSActiveNOT are 0
ULPSActiveNot_ALL1_IRQ	All the ULPSActiveNOT signals corresponding to the lanes with TXULPSExit being high are high
STATEULPS3_IRQ	Lane #3 in ultralow-power state
STATEULPS2_IRQ	Lane #2 in ultralow-power state
STATEULPS1_IRQ	Lane #1 in ultralow-power state
ERRCONTROL3_IRQ	Control error for lane #3
ERRCONTROL2_IRQ	Control error for lane #2
ERRCONTROL1_IRQ	Control error for lane #1
ERRESC3_IRQ	Escape entry error for lane #3(edge trigger interrupt)
ERRESC2_IRQ	Escape entry error for lane #2 (edge trigger interrupt)
ERRESC1_IRQ	Escape entry error for lane #1 (edge trigger interrupt)
ERRCONTENTIONLP1_1_IRQ	Contention LP1 error for lane #1
ERRCONTENTIONLP0_1_IRQ	Contention LP0 error for lane #1
ERRCONTENTIONLP1_2_IRQ	Contention LP1 error for lane #2
ERRCONTENTIONLP0_2_IRQ	Contention LP0 error for lane #2
ERRCONTENTIONLP1_3_IRQ	Contention LP1 error for lane #3
ERRCONTENTIONLP0_3_IRQ	Contention LP0 error for lane #3
ERRSYNCESC3_IRQ	Low power Data transmission synchronization error for lane #3
ERRSYNCESC2_IRQ	Low power Data transmission synchronization error for lane #2
ERRSYNCESC1_IRQ	Low power Data transmission synchronization error for lane #1

NOTE: The error contention signals for DX and DY signals of each lane are ORed together.

Table 12-25 indicates the DSI VCs interrupt events

Table 12-25. DSI Virtual Channel Interrupts

Interrupt Name	Description
ECC_CORRECTION_IRQ	Virtual channel - ECC has been used to correct a 1-bit error (short and long packet only). Indicates whether a 1-bit error correction occurred using the ECC.
PACKET_SENT_IRQ	Indicates that a packet has been sent. It is used when BTA manual mode is used
CS_IRQ	Virtual channel - Check-Sum of the payload mismatch detection
FIFO_RX_OVF_IRQ	RX FIFO overflow. The FIFO used on the L4 interconnect slave port for buffering the data received on the DSI link has overflowed
FIFO_TX_OVF_IRQ	TX FIFO overflow. The FIFO used on the L4 interconnect slave port for buffering the data received on the L4 interconnect slave port has overflowed
BTA_IRQ	Bus turnaround is received from the peripheral (the VC ID used for the last BTA request transfer to the peripheral is used to determine which VC is used to flag the interrupt)
ECC_NO_CORRECTION_IRQ	ECC error (short and long packets). No correction of the header because of more than 1-bit error
FIFO_TX_UDF_IRQ	TX FIFO underflow. The FIFO used on the slave port for buffering the data received on the L4 interconnect port has under-flowed in the middle of a packet transfer

12.4 Display Subsystem Functional Description

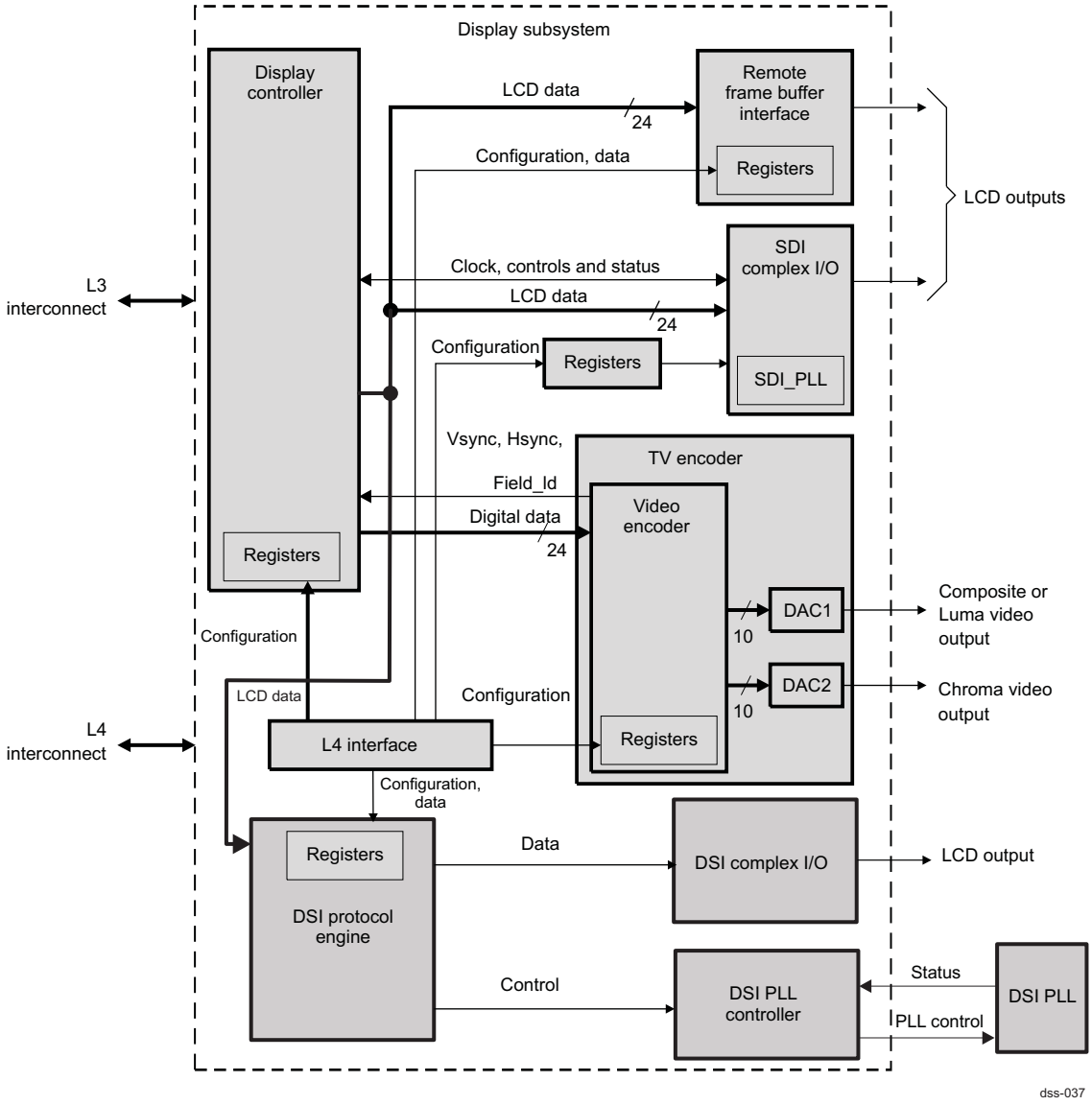
This section describes the functionalities of the LCD and TV display supports by describing the following modules: Display controller, DSI protocol engine, DSI PLL controller, DSI complex I/O, RFBI, SDI, and video encoder.

The functionalities of the display controller are common to both LCD and TV data paths; the RFBI and SDI functionalities are LCD-specific; and the video encoder functionalities are specific to the TV set.

12.4.1 Block Diagram

Figure 12-69 is a schematic of the display subsystem.

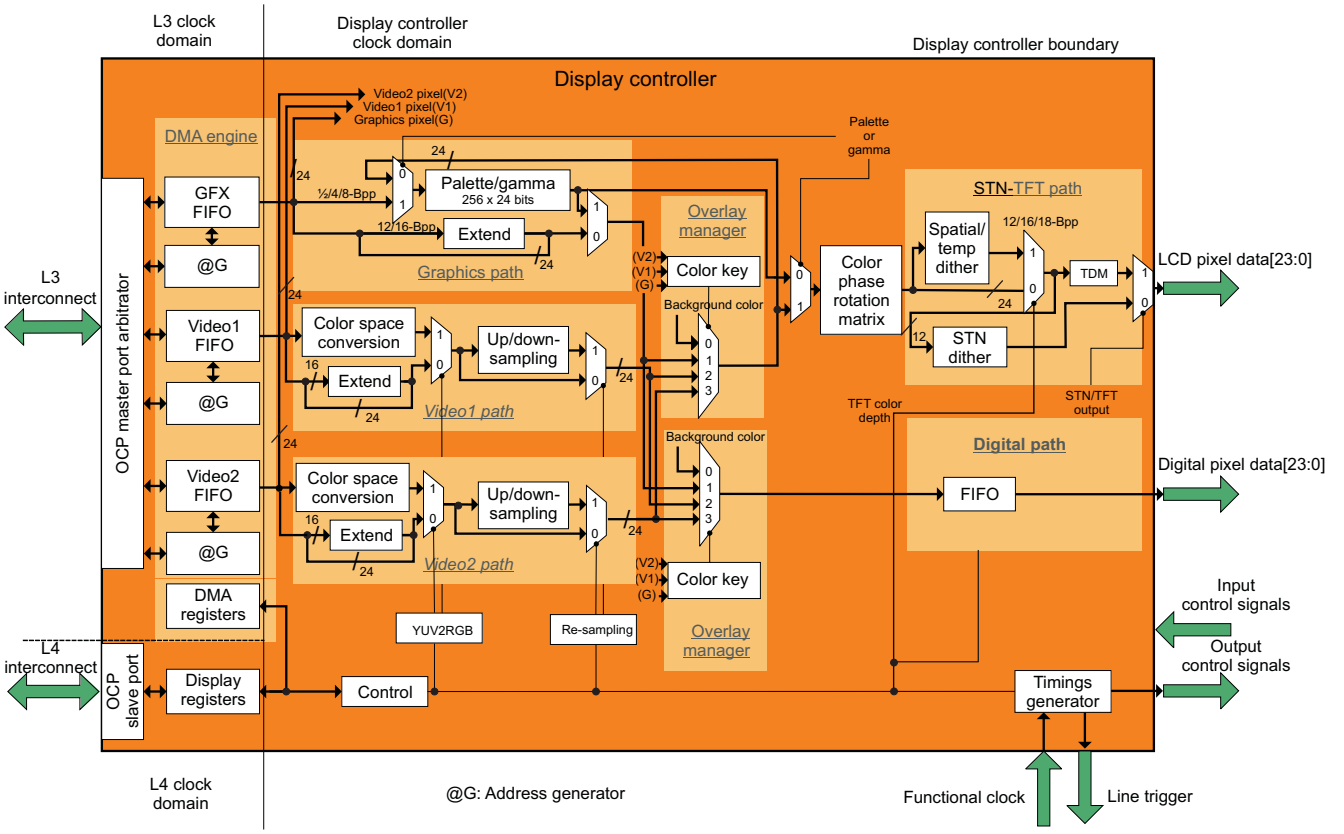
Figure 12-69. Display Subsystem Full Schematic



12.4.2 Display Controller Functionalities

The display controller can read and display the encoded pixel data stored in memory (see Figure 12-70).

Figure 12-70. Display Controller Architecture Overview



Several processes can be configured to manage the graphics pipeline (palette, gamma table correction) and video pipeline (color space conversion, upsampling, downsampling, overlay, and transparency features).

The internal timing generator logic generates the LCD input signals. The external timing generator generates the appropriated signals to drive the digital output. The data from the two overlay managers are sent on the two concurrent 24-bit buses outside the display controller module. The memory accessed by the display controller is either the SDRAM memory or the SRAM memory.

12.4.2.1 Display Modes

12.4.2.1.1 LCD Output

The display subsystem supports two types of display technologies (both monochrome and color modes):

- Passive matrix displays
- Active matrix displays

The passive matrix display mode supports 3375 possible colors, allowing 16, 256, or 3375 colors to be displayed in each frame, depending on the color depth. The monochrome LCD has 15 grayscale levels available.

In active matrix display mode, the configuration of colors depends on the color depth:

- 24 BPP supports 16,777,216 colors.
- 18 BPP supports 262,144 colors.
- 16 BPP supports 65,536 colors.
- 12 BPP supports 4096 colors.

12.4.2.1.2 Digital Output

The digital output is always a 24-bit RGB value based on an external pixel request.

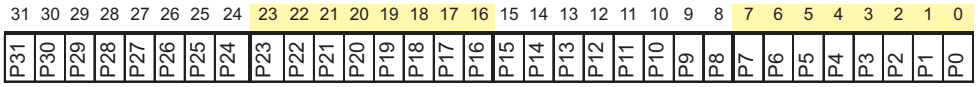
12.4.2.2 Graphics Pipeline

The graphics pipeline is connected to the graphics FIFO controller for the input port and to the two overlay managers (LCD and digital). It consists of one 256-entry palette and some programmable replication logic. The replication logic is used to convert the RGB pixels, excluding the RGB24 format, into RGB24 format based on user programming (replication of the most-significant bits [MSBs] for the RGB24 LSBs or use of 0s). The first unit connected to the input port of the graphics pipeline is the replication logic used for RGB pixels, then the second unit is the palette for concerned pixels.

12.4.2.2.1 Graphics Memory Format

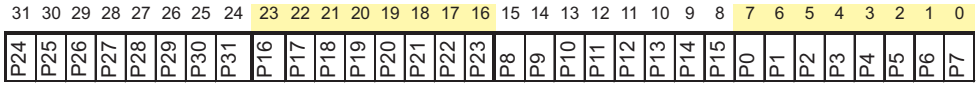
The supported formats for the graphics layer are CLUT bitmaps (1-, 2-, 4-, and 8-BPP) and true color bitmaps in RGB formats (12-, 16-, and 24-BPP [packet and nonpacket RGB24]) and in ARGB or RGBA formats (ARGB 16-, and 32-BPP, and RGBA 32-BPP) as follows:

- BITMAP 1-BPP data memory organization (CLUT) (little endian)



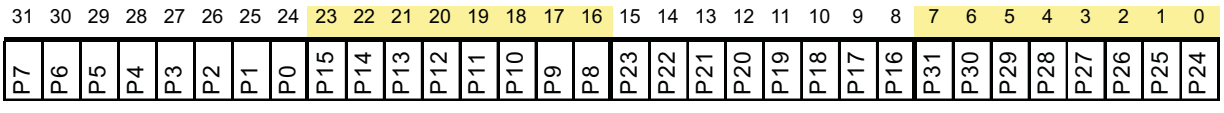
dss-T039

- BITMAP 1-BPP data memory organization (CLUT) (little endian + nibble mode)



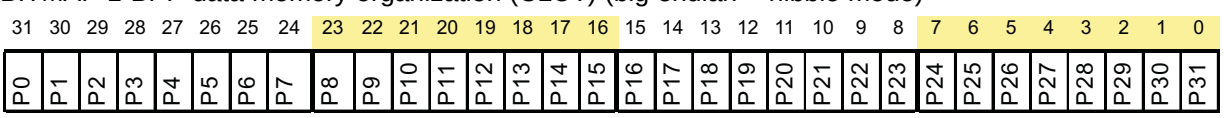
dss-T040

- BITMAP 1-BPP data memory organization (CLUT) (big endian)



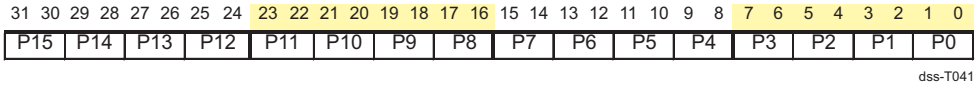
dss-T060

- BITMAP 1-BPP data memory organization (CLUT) (big endian + nibble mode)

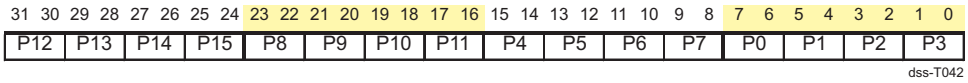


dss-T061

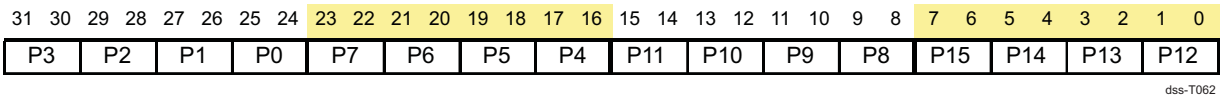
- BITMAP 2-BPP data memory organization (CLUT) (little endian)



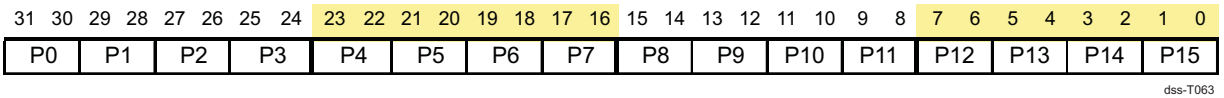
- BITMAP 2-BPP data memory organization (CLUT) (little endian + nibble mode)



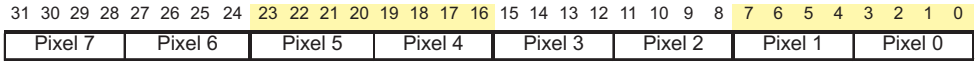
- BITMAP 2-BPP data memory organization (CLUT) (big endian)



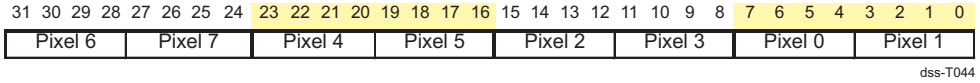
- BITMAP 2-BPP data memory organization (CLUT) (big endian + nibble mode)



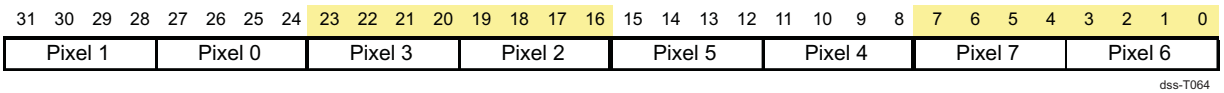
- BITMAP 4-BPP data memory organization (CLUT) (little endian)



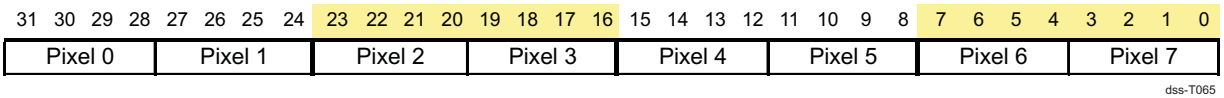
- BITMAP 4-BPP data memory organization (CLUT) (little endian + nibble mode)



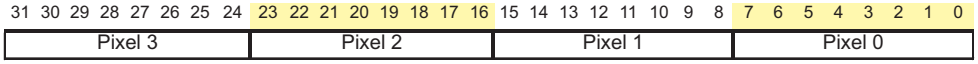
- BITMAP 4-BPP data memory organization (CLUT) (big endian)



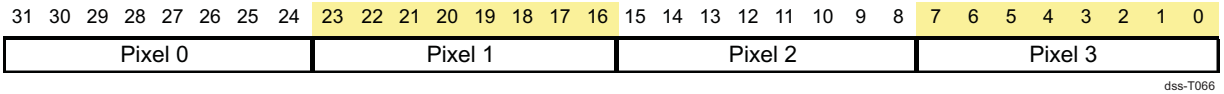
- BITMAP 4-BPP data memory organization (CLUT) (big endian + nibble mode)



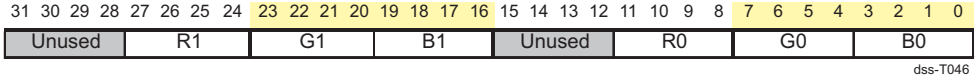
- BITMAP 8-BPP data memory organization (CLUT) (little endian)



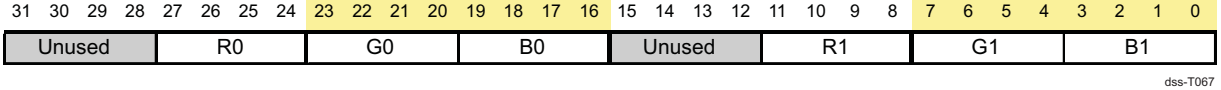
- BITMAP 8-BPP data memory organization (CLUT) (big endian)



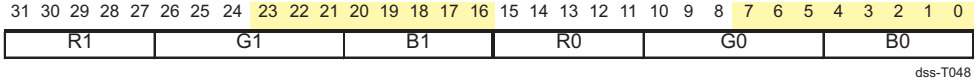
- RGB 12-BPP data memory organization (little endian)



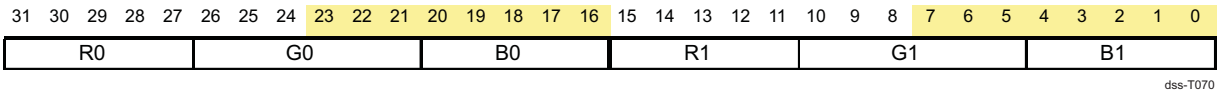
- RGB 12-BPP data memory organization (big endian)



- RGB 16-BPP data memory organization (little endian)

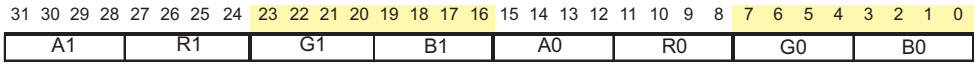


- RGB 16-BPP data memory organization (big endian)



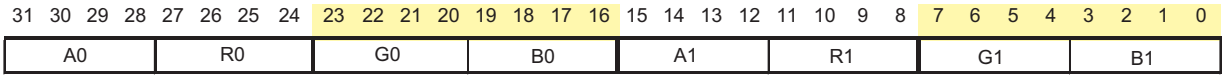
dss-T070

- ARGB 16-BPP data memory organization (little endian)



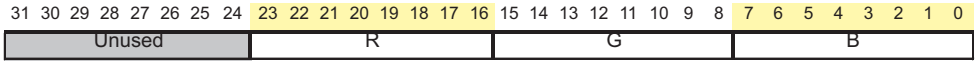
dss-T049

- ARGB 16-BPP data memory organization (big endian)



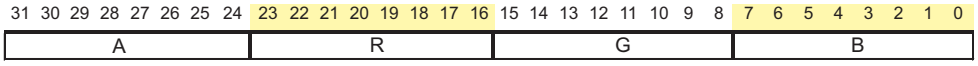
dss-T069

- RGB 24-BPP data memory organization (little or big endian)



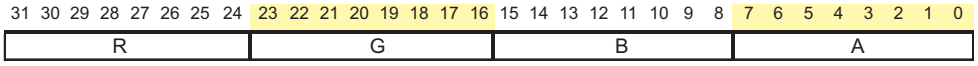
dss-T050

- ARGB 32-BPP data memory organization (little or big endian)



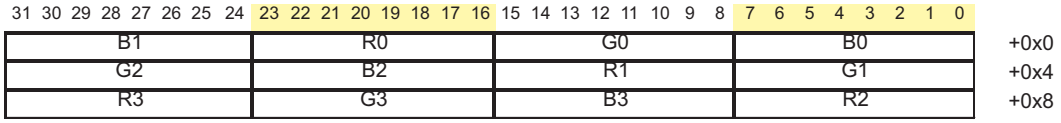
dss-T051

- RGBA 32-BPP data memory organization (little or big endian)



dss-T052

- RGB 24-BPP packet data memory organization (little or big endian)



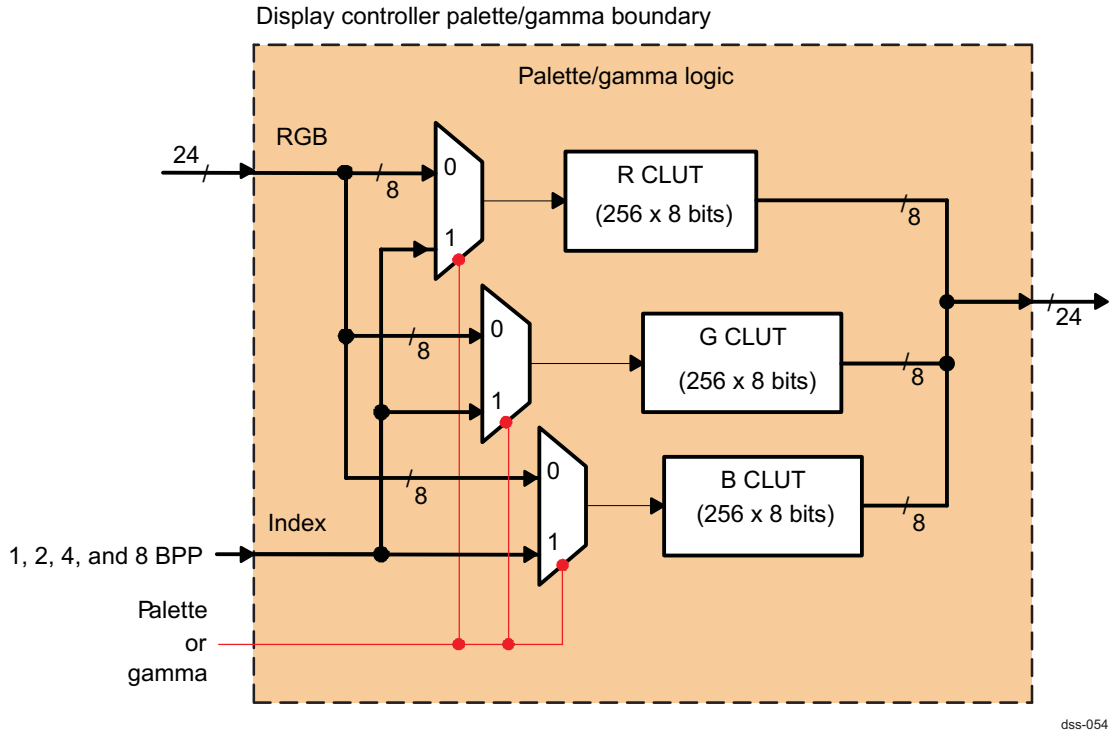
dss-T053

12.4.2.2.2 Color Look-Up Table/Gamma Table

The graphics path supports the palette/gamma table. Figure 12-71 shows the internal architecture of the color look-up/gamma table.

The palette is split into three memories of 256-bit x 8-bit entries. For bitmap (CLUT) indexes, the same value (1-, 2-, 4-, or 8-BPP) indexes the three memories. For gamma curve correction, each R, G, and B component indexes the corresponding memory to combine the three gamma curve values into a 24-bit value. The table can be reloaded every frame, once or never (at the beginning of the frame before fetching the pixels for the graphics and/or video windows).

Figure 12-71. Palette/Gamma Correction Architecture



12.4.2.2.2.1 Color Look-Up Table

The palette mode uses the encoded pixel values from the input graphics FIFO as pointers to index the 24-bit-wide palette: 1-BPP pixels address 2 palette entries, 2-BPP pixels address 4 palette entries, 4-BPP pixels address 16 palette entries, and 8-BPP pixels address 256 palette entries.

When a palette entry is selected by the encoded pixel value, the content of the entry is sent to the color/grayscale space/time base passive matrix dithering circuit, or to the color time base active matrix dithering circuit.

In color mode, the value within the palette is made up of three 8-bit fields, one for each color component (red, green, and blue). For color operation, an individual frame is limited to a selection of 256 colors (the number of palette entries).The format of one of the palette values in the memory is as follows:

- 24-BPP Data Memory Organization (Little Endian or Nibble)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused								R								G								B							

In monochrome mode, only one 8-bit value is present.

- 24-BPP Data Memory Organization (Llittle Endian or Nibble)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused								Unused								Unused								Gray							

After passing through the palette, 256 gray scales and 16,777,216 colors are numbers obtained. A redundancy introduced in the dithering logic step reduces these numbers when displaying. For passive matrix panels, the colors are limited to 15 gray scales and 3375 colors.

- Passive matrix technology

  The palette is bypassed in 12, 16, and 24 BPP. The palette is not used.
- Active matrix technology

The palette is bypassed in 12, 16, and 24 BPP, allowing up to 2^{24} = 16,777,216 colors to be displayed.

12.4.2.2.2 Gamma Table

In the gamma curve mode, the selected encoded pixel values based on the color keys from the video or graphics paths are sent to the gamma curve table. The mode is available only if the color look-up palette is not used for graphics. The output of the gamma curve processing is always sent to the LCD output. It is not available on digital output.

Each component of encoded pixel value is used as a pointer to index 1 out of 256 24-bit gamma curve entries in the table. Each 8-bit component is replaced with the 8-bit table value corresponding to an R, G, or B component. The format of one of the gamma curve values in the memory is as follows:

- 24-BPP Data Memory Organization (Little or Big Endian)

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Unused	Gamma-R	Gamma-G	Gamma-B

12.4.2.2.2.1 Replication Logic

The replication logic increases the color depth of the graphics and video encoded pixels (from true color RGB 12-, and 16-BPP to 24-BPP). The encoded value is shifted to the 24-bit alignment. The MSB bits are copied to the LSB missing ones. Then the graphics are merged with the video data based on the transparency color keys. When the replication logic is not selected, the encoded pixel values are shifted to the MSB boundary of the 24-bit format. The missing bit values are filled up with 0s.

This is an example for RGB16 extension:

- Original 16-BPP data:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

dss-T055

- If replication logic is ON:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	R4	R3	R2	G5	G4	G3	G2	G1	G0	G5	G4	B4	B3	B2	B1	B0	B4	B3	B2

dss-T056

- If replication logic is OFF:

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	0	0	0	G5	G4	G3	G2	G1	G0	0	0	B4	B3	B2	B1	B0	0	0	0

dss-T057

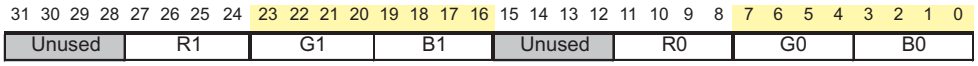
12.4.2.3 Video Pipeline

The video pipeline is connected to the video FIFO controller for the input port and to the two overlay managers (LCD and digital). It consists of the Re-Sampling unit, the Color Space Conversion Unit, and some programmable replication logic. The replication logic is used to convert the RGB pixels, excluding the RGB24 format, into RGB24 format based on user programming (replication of the MSBs for the RGB24 LSBs or use of 0s). The first unit connected to the input port of the video pipeline is the Re-Sampling Unit, then the replication logic used for RGB pixels, then the Color Space Conversion Unit for YUV422 pixels.

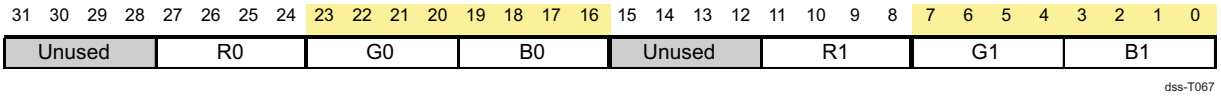
12.4.2.3.1 Video Memory Formats

The display subsystem supports the following formats for the video layer: YUV2, UYVY, RGB12, RGB16, RGB24 (non-packed and packed formats), ARGB16 (video channel 2 only), ARGB32 (video channel 2 only), and RGBA32 (video channel 2 only).

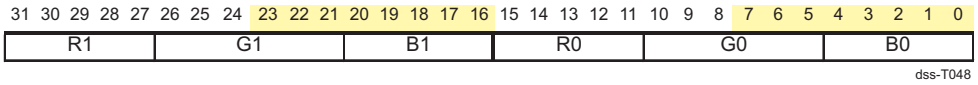
- RGB 12-BPP data memory organization (little endian)



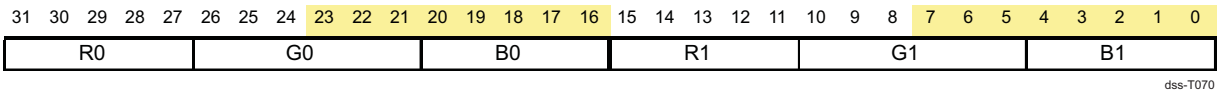
- RGB 12-BPP data memory organization (big endian)



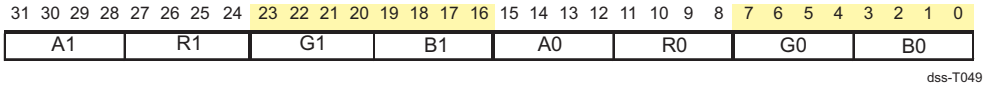
- RGB 16-BPP data memory organization (little endian)



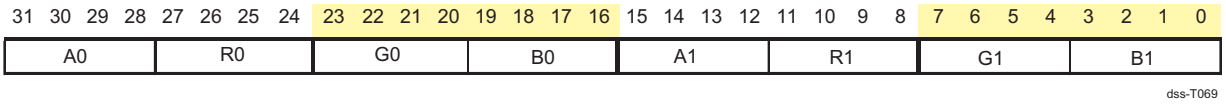
- RGB 16-BPP data memory organization (big endian)



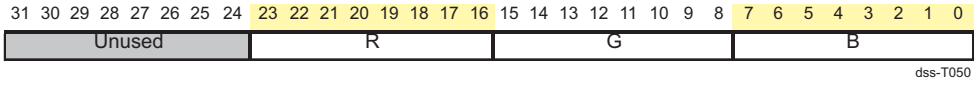
- ARGB 16-BPP data memory organization (little endian + video 2 channel only)



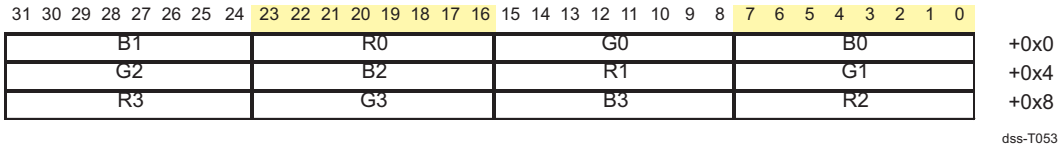
- ARGB 16-BPP data memory organization (big endian + video 2 channel only)



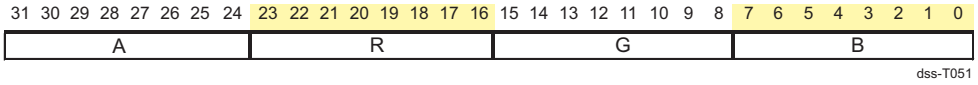
- RGB 24-BPP data memory organization (little or big endian)



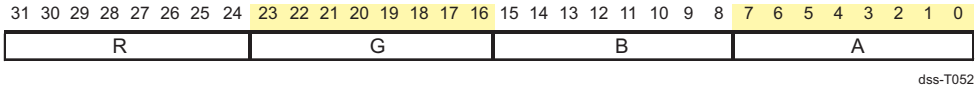
- RGB 24-BPP packet data memory organization (little or big endian)



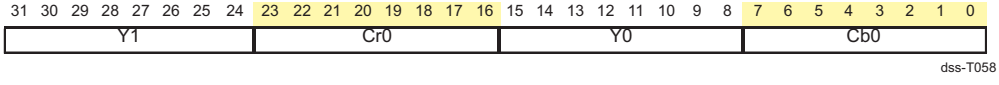
- ARGB 32-BPP data memory organization (little or big endian + video 2 channel only)



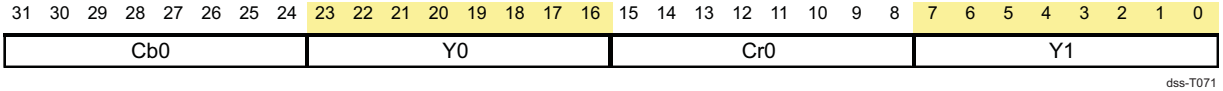
- RGBA 32-BPP data memory organization (little or big endian + video 2 channel only)



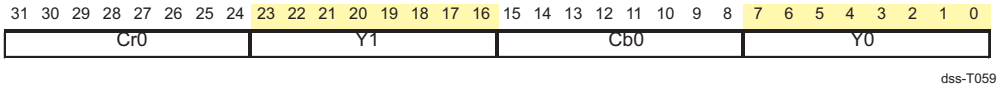
- UYVY 4:2:2 data memory organization (little endian)



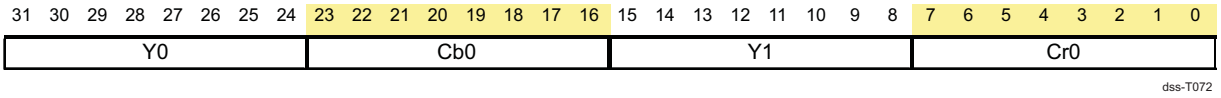
- UYVY 4:2:2 data memory organization (big endian)



- YUV2 4:2:2 data memory organization (little endian)



- YUV2 4:2:2 data memory organization (big endian)



12.4.2.3.2 Color Space Conversion

The color space conversion module converts the video-encoded pixel values from YCbCr 4:2:2 format into RGB24. Figure 12-72 and Figure 12-73 detail the YCbCr 4:2:2 conversion to YCbCr 4:4:4 depending on the rotation parameters.

Figure 12-72. YCbCr 4:2:2 to YCbCr 4:4:4 (0- or 180-Degree Rotation)

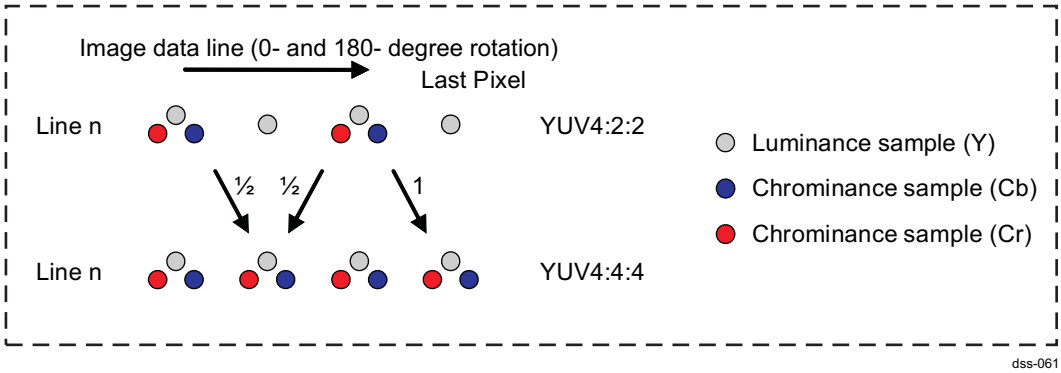
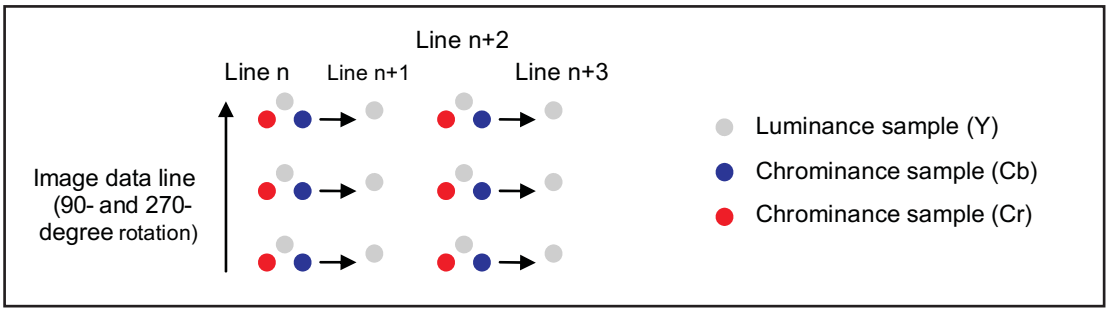


Figure 12-73. YCbCr 4:2:2 to YCbCr 4:4:4 (90- or 270-Degree Rotation)



The interpolation of the missing chrominance component is given by the equation in Figure 12-74.

Figure 12-74. Interpolation of the Missing Chrominance Component

$$Cb_n(YCbCr\ 444) = \frac{Cb_{n-1}(YCbCr\ 422) + Cb_{n+1}(YCbCr\ 422)}{2}\ (\text{n odd})$$

$$Cr_n(YCbCr\ 444) = \frac{Cr_{n-1}(YCbCr\ 422) + Cr_{n+1}(YCbCr\ 422)}{2}\ (\text{n odd})$$

dss-E062

First, to convert the YCbCr 4:2:2 encoded pixel values into YCbCr 4:4:4 format, the missing chrominance samples (Cb and Cr) are interpolated using the average values of the two closest values on the same line (1/2, 1/2) or are repeated from the second pixel in the same 32-bit container.

- In case of rotation 0-degree, for the last pixel, the chrominance samples are duplicated using the values from the previous pixel; otherwise, the chrominance samples are averaged using the two adjacent values.
- In case of 180-degree rotation, for the first pixel the chrominance samples missing are duplicated from

the adjacent pixel; otherwise, the chrominance samples are averaged using the two adjacent values.

- In case of rotation 90- and 270-degree, the missing chrominance components are duplicated from the adjacent pixel in the same 32-bit container.

  In case of 5-tap configuration for the vertical filtering, the missing chrominance samples are always duplicated using the second chrominance samples in the same 32-bit value.

  Then the pixels are converted from YCbCr color space into the RGB color space, because the output format of the color space conversion is RGB24 (8-bit value per component: Red, green, and blue). The following matrices show the 11-bit coefficients registers used to convert from YCbCr 4:4:4 into RGB24. Users set the coefficients according to the standard used to encode the pixel data in YCbCr color space.

  In case of resampling, the YUV422 format is converted into YUV444. The YUV422-to-YUV444 processing is bypassed in the color space conversion unit.

  If the active range for the luminance samples (Y) is [16235] and [16240] for the chrominance samples (Cb and Cr), the values of R, G, and B output components are clipped to the range [0:255]. The equation shown in Figure 12-75 gives the 11-bit coefficients of the YCbCr to RGB color space conversion.

Figure 12-75. YCbCr to RGB Registers (VIDFULLRANGE=0)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y - 16 \\ Cr - 128 \\ Cb - 128 \end{bmatrix}$$

dss-E063

If the active range for the luminance samples (Y) and chrominance samples (Cb and Cr) is [0:255], the values of R, G, and B output components are clipped to the range [0:255]. The equation shown in Figure 12-76 gives the 11-bit coefficients of the YCbCr-to-RGB color space conversion.

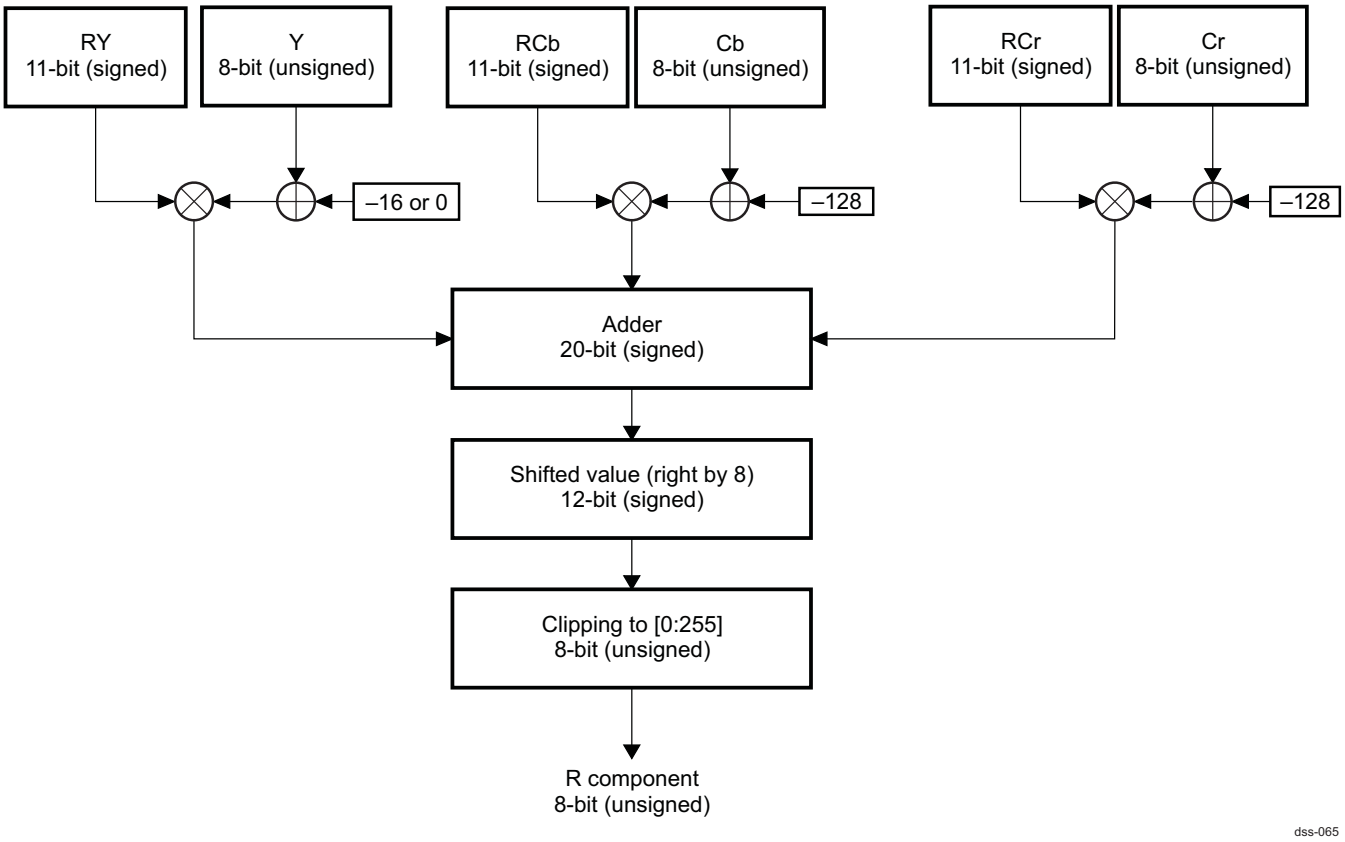
Figure 12-76. YCbCr to RGB Registers (VIDFULLRANGE=1)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y \\ Cr - 128 \\ Cb - 128 \end{bmatrix}$$

dss-E064

Figure 12-77 describes the computation for the calculation of the R component. The same computation applies for the G and B components:

Figure 12-77. Color Space Conversion Macro-Architecture



12.4.2.3.3 Hardware Cursor

The video layer can be used to display the hardware cursor. The encoded pixel data for the cursor image are in RGB12, RGB16 or RGB24 formats and the color space conversion block is bypassed. The transparency color key can be used when a non rectangle shape is used.

The alpha blending can be used to show a partial transparent cursor. When the alpha blender is enabled, the graphics layer is on top of the video layers. The cursor uses the graphics layer. The pixel alpha blending or the transparency color key can be used.

12.4.2.3.4 Up-/Down-Sampling

The video layer has a dedicated resizing block to upsample and downsample the video-encoded pixels. The supported input formats from memory are RGB24, RGB16, and YUV422

(RGB12 and all the alpha formats like ARGB and RGBA are not supported)

Users must set the right size and position of the original video before resizing for the upsampled/downsampled video to be inside the display screen boundaries.

The filtering applies on each component independently R, G, and B).

For the horizontal up-/downsampling, the equation is R component with five taps):

$$Rout(n) = \left(\sum_{i=-2}^{i=2} Ci(\Phi) \times Rin(n+i) \right) >> 7 \quad \text{dss-E066} \qquad (2)$$

For the vertical up-/downsampling, the equation is R component with three taps):

$$Rout(n) = \left(\sum_{i=-1}^{i=1} Ci(\Phi) \times Rin(n+i) \right) >> 7 \quad \text{dss-E067} \tag{3}$$

For the vertical up-/downsampling, the equation is R component with five taps):

$$Rout(n) = \left(\sum_{i=-2}^{i=2} Ci(\Phi) \times Rin(n+i) \right) >> 7 \quad \text{dss-E068}$$

Rout: R component output

$Ci(\Phi)$ dss-E069 : FIR filter coefficients

Rin: R component input (4)

The pixel (n + 1) is older than pixel (n). The line (n + 1) is older than line (n).

NOTE: The coefficients Ci() depend on the phase between input and output pixels.

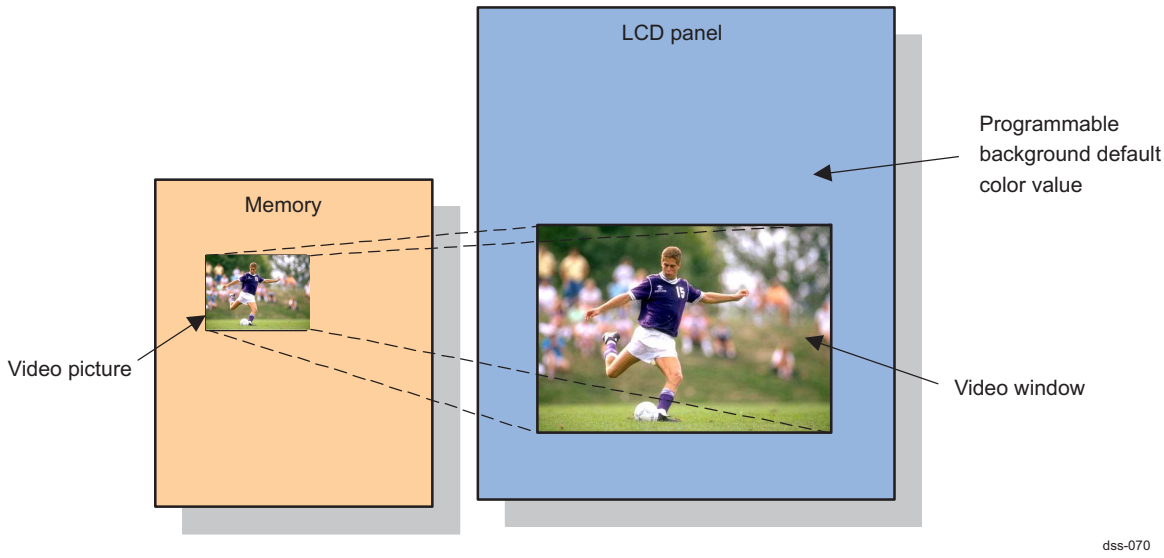
NOTE: If the 5-tap resizer is used for RGB16 and YUV422 picture formats, the width of the input picture must be a multiple of 2 pixels and more than 5 pixels.

DISPC_VIDn_ATTRIBUTES[21] VIDVERTICALTAPS == 1

DISPC_VIDn_PICTURE_SIZE[10:0] VIDORGSIZEX > 4 and even

Figure 12-78 shows an example of video upsampling.

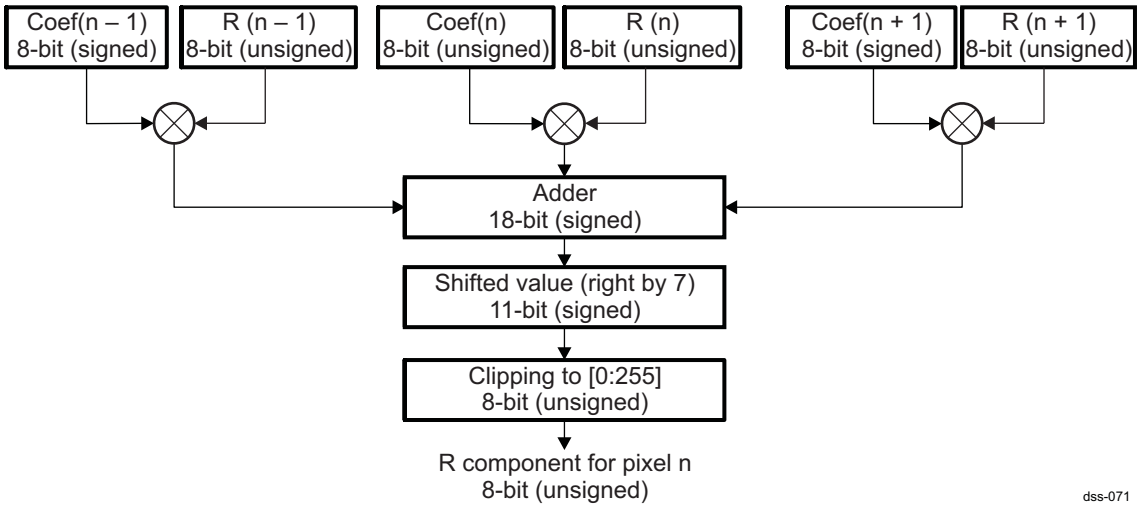
Figure 12-78. Video Upsampling



Filter Description

The up/downsampling filter is a poly-phase filter with five taps and eight phases for the horizontal filter and a programmable number of taps (three or five) and eight phases for vertical filter. The upsampling ratio is up to x8. The downsampling ratio using 3-tap configuration is/2. The downsampling ratio using 5-tap configuration is/4. The vertical filter is first applied to the encoded input pixel data; and then the horizontal filter is applied on the resulting pixel values to generate the output pixel values. Figure 12-79 shows the computation for the R component in the case of three coefficients (vertical filtering). The same computation applies to the G and B components.

Figure 12-79. Resampling Macro-Architecture (3-Coefficient Processing)



To determine if the minimum functional clock matches the down sampling ratio and the desired Pixel clock, the following formula must be used in conjunction with Table 12-26 and Table 12-27.

Ratio V when performing a vertical down-sampling only

$$h\_ratio = \frac{DISPC\_SIZE\_LCD.PPL}{DISPC\_VID\_SIZE.VidSizeX}$$

$$v\_ratio = \frac{DISPC\_VID\_PICTURE\_SIZE.VidOrgSizeY}{DISPC\_VID\_SIZE.VidSizeY}$$

$$Ratio = \frac{v\_ration}{2 \times h\_ratio} \qquad If \quad 1 < v\_ratio \le 2$$

$$Ratio = \max\left(\frac{v\_ration}{2 \times h\_ratio}, \frac{v\_ration - 2}{2 \times (h\_ratio - 1)}\right) \qquad If \quad 2 < v\_ratio \le 4$$

dss_swpu108-E135

NOTE: For frequency ratio calculation on the TV output, it is correct to replace DISPC_SIZE_LCD with DISPC_SIZE_DIG.

When the down-sampling ratio is below 0.5, it is not possible to use a video in full screen.

Ratio H when performing a horizontal down-sampling only

$$Ratio = \frac{DISPC\_VID\_PICTURE\_SIZE.VidOrgSizeX}{DISPC\_VID\_SIZE.VidSizeX}$$

dss_swpu108-E136

Ratio H+V when performing a horizontal and vertical down-sampling

Ratio = max (horizontal Ratio, vertical Ratio) as previously defined.

Table 12-26. Functional Clock Frequency Requirement in RGB16 & YUV422—Active Matrix Display

Minimum Functional Clock (MHz)		Horizontal Resampling				
		Off	Up	1:1 – 1:2	1:2 – 1:3	1:3 – 1:4
Vertical Resampling	Off	AxPCLK	AxPCLK	2xPCLK	3xPCLK	4xPCLK
	Up	AxPCLK	AxPCLK	2xPCLK	3xPCLK	4xPCLK
	3-tap 1:1 to 1:2	2xPCLK	2xPCLK	4xPCLK	6xPCLK	8xPCLK
	5-tap 1:1 to 1:4	RatioxPCLK	RatioxPCLK	RatioxPCLK	RatioxPCLK	RatioxPCLK

With A = 1 in case all the data and synchronization signals are asserted and deasserted on the rising edge of the PCLK; otherwise, A = 2.

Table 12-27. Functional Clock Frequency Requirement in RGB24—Active Matrix Display

Minimum Functional Clock (MHz)		Horizontal Resampling				
		Off	Up	1:1 – 1:2	1:2 – 1:3	1:3 – 1:4
Vertical Resampling	Off	AxPCLK	AxPCLK	2xPCLK	3xPCLK	4xPCLK
	Up	AxPCLK	AxPCLK	2xPCLK	3xPCLK	4xPCLK
	3-tap 1:1 to 1:2	2xPCLK	2xPCLK	4xPCLK	6xPCLK	8xPCLK
	5-tap 1:1 to 1:4	RatioxPCLK	RatioxPCLK	2xRatioxPCLK	2xRatioxPCLK	2xRatioxPCLK

With A = 1 in case all the data and synchronization signals are asserted and deasserted on the rising edge of the PCLK; otherwise, A = 2.

Use case example:

An input picture of 1024*768 is scaled to an output picture of size of 800*600 and displayed onto a LCD of resolution1280*768 at a PCLK of 74.25 MHz with a DSS functional clock of 133 MHz.

In this example, a H+V down-sampling is done on the input picture. Firstly the Ratio V and H are determined and the resulting maximum value is taken to calculate the functional clock frequency required.

Ratio V: h_ratio = 1.6 and v_ratio = 1.28 then Ratio = 0.4

Ratio H: Ratio = 1.28

Ratio H+V: Ratio = max (1.28, 0.4) = 1.28

In this use case, the horizontal and vertical down sampling range are 1:1–1:2. The 3-tap or 5-tap configuration can be taken into consideration. Therefore, from Table 12-26 and Table 12-27, If in RGB16-YUV422:

- 3-taps → DSS functional clock = 4 * PCLK = 297 MHz
- 5-taps → DSS functional clock = Ratio * PCLK = 95.36 MHz

If in RGB24,

- 3-taps → DSS functional clock = 4 * PCLK = 297 MHz
- 5-taps → DSS functional clock = 2 * Ratio * PCLK = 190.72 MHz

In this use case, the pixel format supported is RGB16-YUV422 in a 5-tap configuration.

12.4.2.4 Overlay Support

CAUTION

Enabling overlay optimization (setting the DSS.DISPC_CONTROL [12] OVERLAYOPTIMIZATION bit) if no overlay region effectively exists (the DSS.DISPC_VIDn_ATTRIBUTES [0] VIDENABLE bit is cleared, with n = 1, 2) leads to unpredictable behavior. The overlay optimization feature must be enabled only when an overlay area exists. Before enabling the overlay optimization, the DSS.DISPC_GFX_WINDOW_SKIP[31:0] GFXWINDOWSKIP bit field must be first set according to the video1 and graphics windows overlap.

The overlay mechanism consists of displaying more than one layer (graphics and video layers) using rules based on priority and transparency color keys.

When the pixel format is ARGB or RGBA, the color key match logic uses only the RGB value defined by ARGB or RGBA. The alpha blending factor is ignored.

The overlay managers are based on the same rules for priority and transparency color keys (see Figure 12-82).

Each data pipeline is assigned a single overlay related to a single display controller output.

The overlay manager is connected to all three outputs of the pipelines (graphics, video1 and video2). The output of the LCD overlay manger is connected to the Spatial/Temporal Dithering, and Passive Matrix units and back to the palette unit in the case of Gamma correction.

12.4.2.4.1 Priority Rule

The overlay manager can be configured in two distinct modes:

- Alpha mode (only source color key with the graphics layer)
- Normal mode (no alpha support)

The following rules apply in normal mode:

The video1 layer is always on top of the graphics layer. The video2 layer is always on top of the video1 and graphics. The display controller reads the data for each buffer from the system memory and, depending on the transparency color key values, displays either the pixels in the video layer, the pixels in the graphics layer, or the solid background color.

Each layer can have any size up to full-display screen. If there are no graphics or video-encoded pixels at a specific position, the programmable, solid background color appears (see Figure 12-84).

Figure 12-80. Overlay Manager in Normal Mode

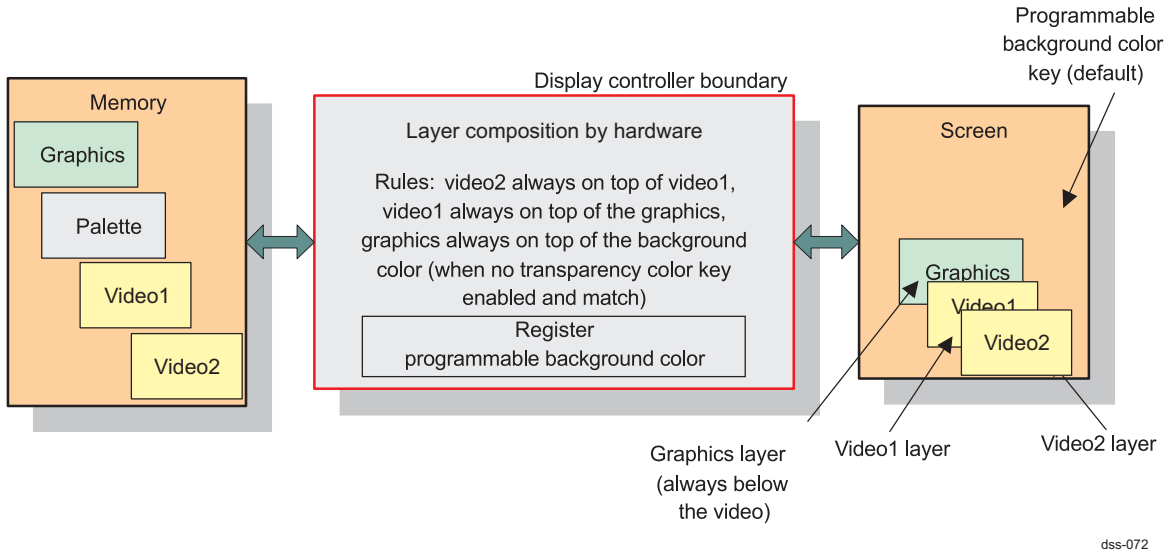
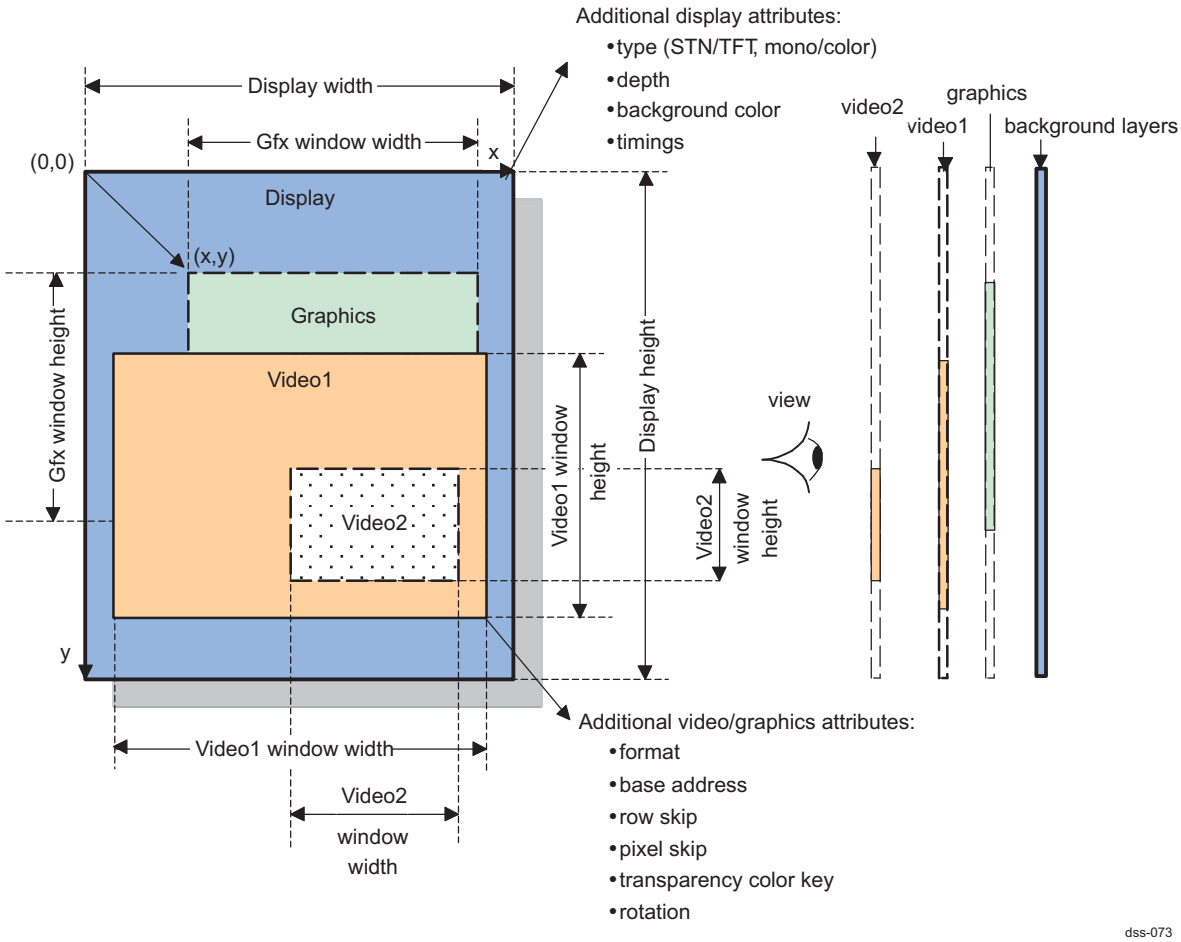


Figure 12-81. Display Attributes in Normal Mode



The following rules apply in alpha mode:

The video2 layer is always on top of the video1 layer. The graphics layer is always on top of the video1 and video2. The display controller reads the data for each buffer from the system memory and, depending on the transparency color key values, displays either the pixels in the video layer, the pixels in the graphics layer, or the solid background color.

Each layer can have any size up to full-display screen. If there are no graphics or video-encoded pixels at a specific position, the programmable, solid background color appears (see Figure 12-84).

Figure 12-82. Overlay Manager in Alpha Mode

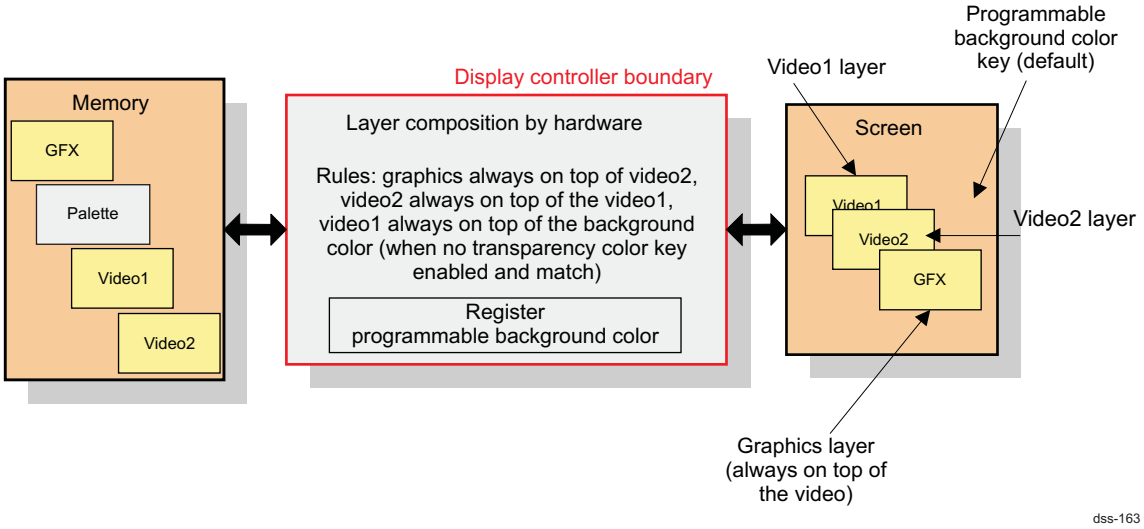


Figure 12-83. Display Attributes in Alpha Mode

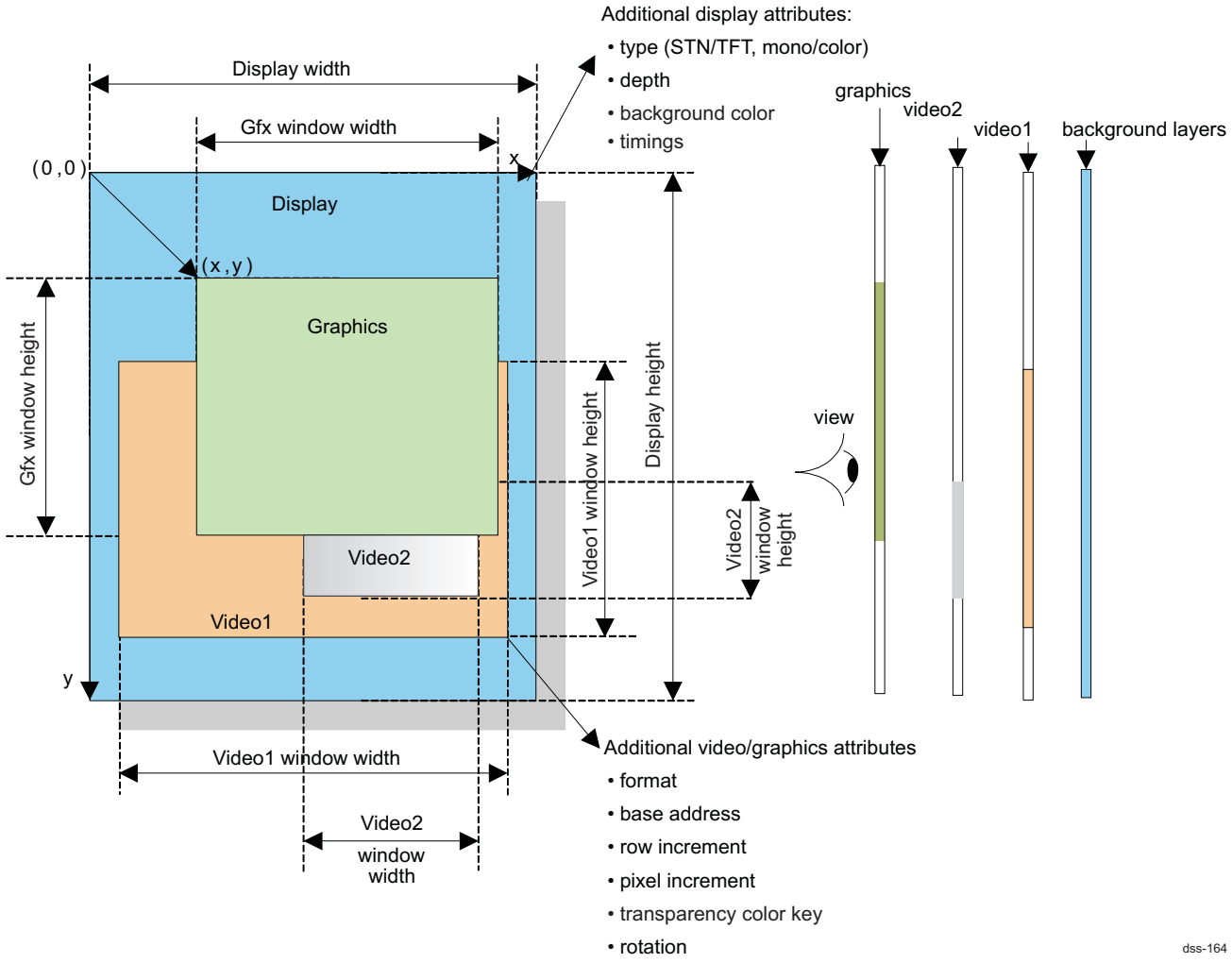
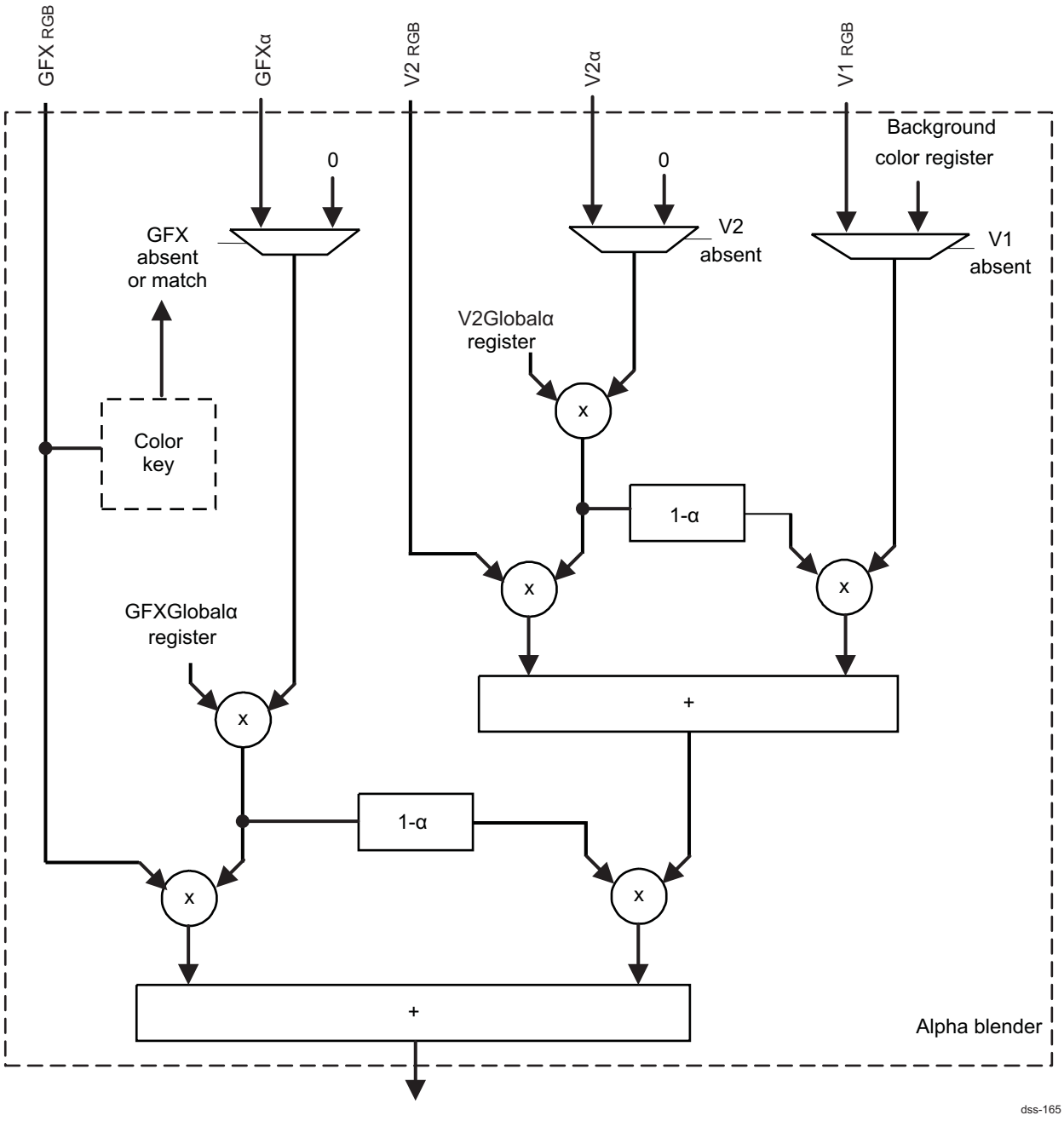


Figure 12-84 shows the alpha blending processing in detail.

Figure 12-84. Alpha Blending Macro Architecture



NOTE: "1-alpha" operator corresponds to the basic 1's complement operation.

The alpha blending value is defined by the pixel value (ARGB or RGBA formats). A global alpha blending value can be defined and used in combination with the pixel alpha blending value. If the pixel format contains no alpha blending value, the pixel alpha value is considered to be 0xFF.

In case of ARGB-444, the alpha blending is defined using a 4-bit value. It is converted into an 8-bit value by duplicating the 4-bit value. Table 12-28 details the alpha blending 4-bit values and the corresponding blending percentage.

Table 12-28. Alpha Blending 4-Bit Values

Alpha Blending 4-Bit Value (ARGB-444)	Alpha Blending 8-Bit Value (Converted Value)	% Blending
0x0	0x00	100% (transparent)
0x1	0x11	93.33%
0x2	0x22	86.6%
...	...	...
0xE	0xEE	6.6%
0xF	0xFF	0% (opaque)

12.4.2.4.2 Transparency Color Keys

12.4.2.4.2.1 Normal Mode

This section describes the features available in normal mode.

The two transparency color keys are the video source transparency color key and the graphics destination transparency color key. The encoded pixel color value is compared to the transparency color key. For CLUT bitmaps, the palette index is compared to the transparency color key and not to the palette value pointed out by the palette index.

NOTE: The video source transparency color key and graphics destination transparency color key cannot be active at the same time.

- Video source transparency color key value:

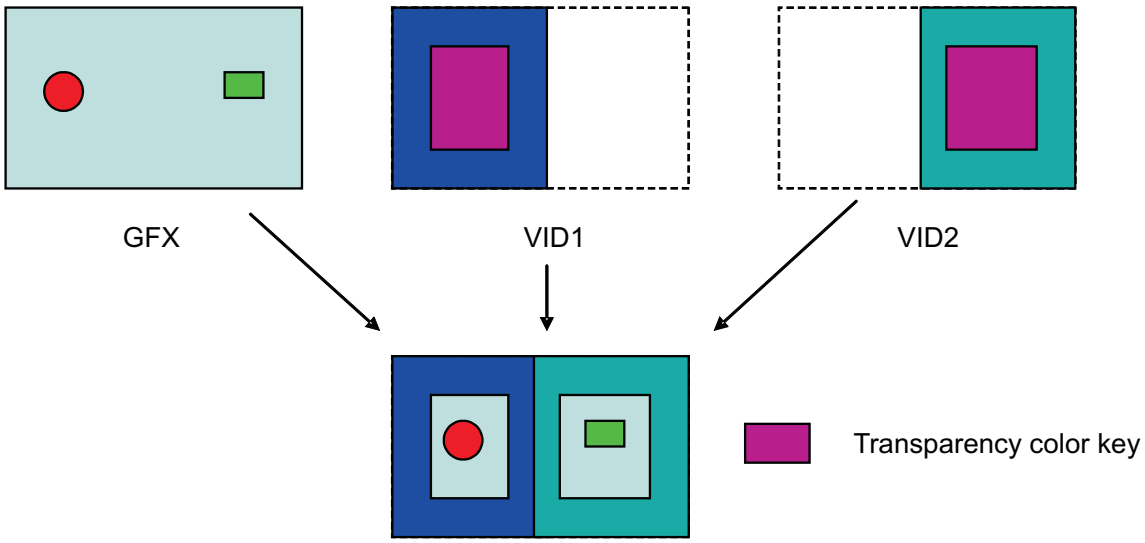
  The video source transparency color key value defines the encoded pixel data considered as the transparent pixel. The encoded pixel values with the source color key value are pixels not visible on the screen, and the underlayer encoded pixel values or solid background color are visible.

  The video source transparency color key can be used only if the color space conversion and the up-/down-scaling modules are disabled. The format of the data is RGB 16. (This feature handles the hardware cursor displayed by one of the video layers.)

  To enable the video source transparency color key, set to 0x1 the DSS.DISPC_CONFIG[11] TCKLCDSELECTION bit for LCD output or the DSS.DISPC_CONFIG[13] TCKDIGSELECTION bit for digital output. Program the DSS.DISPC_CONFIG[10] TCKLCDENABLE bit (LCD output) or the DSS.DISPC_CONFIG[12] TCKDIGENABLE bit (digital output) to enable or disable the transparency color key.

  An example is shown in Figure 12-85: The video source transparency is applied on video1 (VID1) and video2 (VID2) layers. The pixels with the transparency color key are not displayed; instead, underlying layers are shown.

Figure 12-85. Video Source Transparency Example



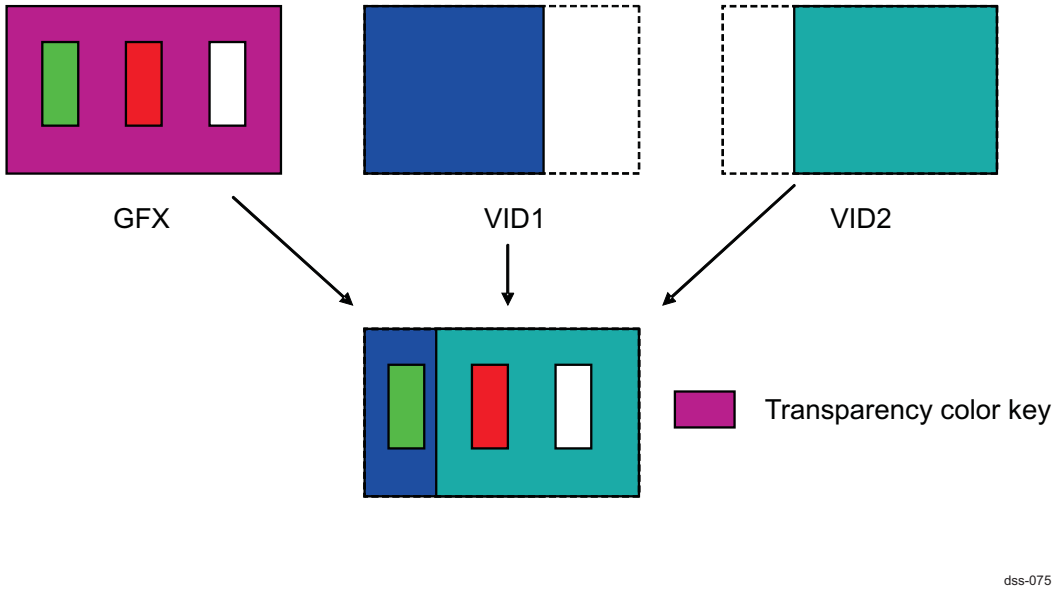
- Graphics destination transparency color key value:

  The graphics destination transparency color key value defines the encoded pixels in the video layers to be displayed. The encoded pixel values with the destination color key value are pixels not visible on the screen and the pixels different from the transparency color key are displayed over the video layers. The destination transparency color key is applicable only in the graphics region when graphics and video overlap; otherwise, the destination transparency color key is ignored.

  To enable the graphics destination transparency color key, set to 0x0 the DSS.DISPC_CONFIG[11] TCKLCDSELECTION bit for LCD output or the DSS.DISPC_CONFIG[13] TCKDIGSELECTION bit for digital output. Program the DSS.DISPC_CONFIG[10] TCKLCDENABLE bit (LCD output) or the DSS.DISPC_CONFIG[12] TCKDIGENABLE bit (digital output) to enable or disable the transparency color key.

  An example is shown in Figure 12-86: The destination transparency is applied on graphics (GFX) layer and the pixels without the transparency color key are displayed over the overlying layers.

Figure 12-86. Graphics Destination Transparency Example



12.4.2.4.2.2 Alpha Mode

This section describes the features available in alpha mode.

Only the graphics source transparency color key is available The encoded graphics pixel color value is compared to the transparency color key. The encoded pixel values with the source transparency key are not visible and the under-layer encoded pixel values or solid background color are visible. To enable the graphics source transparency color key, set to 0x0 the DSS.DISPC_CONFIG[11] TCKLCDSELECTION bit for LCD output or the DSS.DISPC_CONFIG[13] TCKDIGSELECTION bit for digital output. Program the DSS.DISPC_CONFIG[10] TCKLCDENABLE bit (LCD output) or the DSS.DISPC_CONFIG[12] TCKDIGENABLE bit (digital output) to enable or disable the transparency color key. In the case of CLUT bit maps, the palette index is compared to the transparency color key and not the palette value pointed out by the palette index.

12.4.2.4.3 Overlay Optimization (Only Available in Normal Mode)

The display controller can be configured to take advantage of the fact that the graphics pixels under video window 1 are not visible when the transparency color key is not used. The optimization can be selected to reduce the bandwidth used to fetch the pixels for graphics. The color key must be disabled. The graphics pixels under the video window 1 are not fetched from system memory. At least the video window 1 and the graphics window must be enabled. The following graphic formats are supported: RGB (RGB16 and RGB24 packed and unpacked), YUV422, and BITMAP 8. The formats BITMAP 1, 2, and 4 are not supported. The video format can be RGB (RGB16, RGB24 packed and unpacked, and YUV422 formats). The DMA engine does not fetch the unnecessary graphics pixels to avoid extra bandwidth use. Only visible pixels from graphics and video buffers in system memory are fetched and displayed by the display controller.

12.4.2.5 Active/Passive Matrix Display Data Path

For active matrix display data path, the following blocks are serial and each of them can be bypassed:

- Color phase rotation
- Spatial/temporal dithering
- Multiple cycle data format

For passive matrix display data path, the following blocks are serial and each of them can be bypassed:

- Color phase rotation

- Spatial/temporal dithering
- Passive matrix technology

12.4.2.5.1 Color Phase Rotation

The Color Phase Rotation (CPR) can be used to correct the LCD output colorimetry in case of non pure white backlight.

The color phase rotation can be selected for passive matrix and active matrix panel. The logic is integrated after the LCD overlay manager or the palette while using the gamma correction and before the spatial/temporal dithering. The color phase rotation can be selected to correct the nonpure white backlight of the LCD module by using a programmable matrix to convert the 24-bit RGB pixel value into a new 24-bit RGB pixel value. The matrix is programmed through a set of nine 10-bit signed coefficients. The output of the calculation is clipped to [0:255]. The color phase rotation is processed by the equation shown in Figure 12-87.

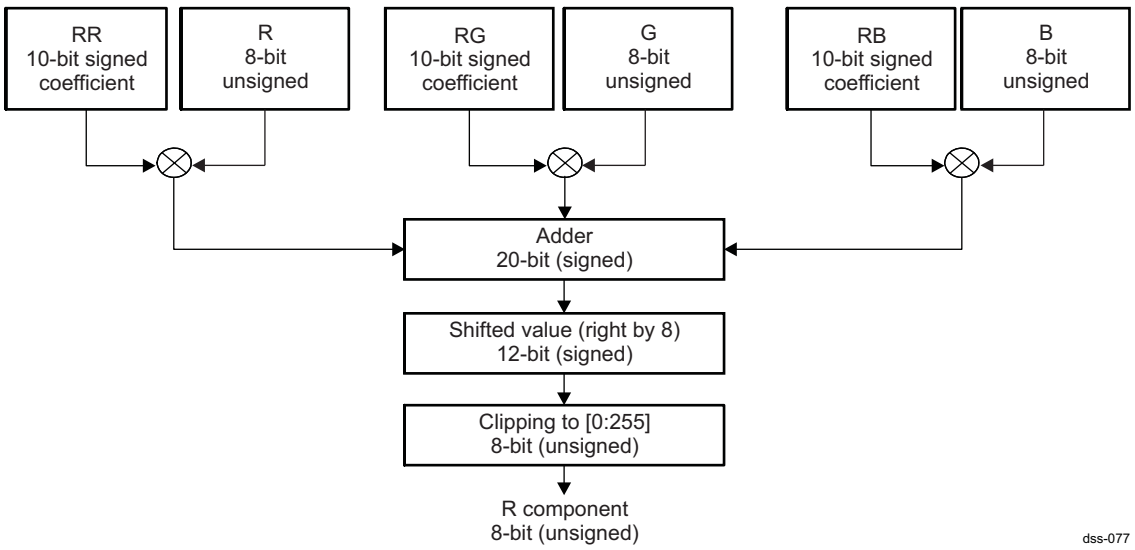
Figure 12-87. Color Phase Rotation Matrix

$$\begin{bmatrix} Rout \\ Gout \\ Bout \end{bmatrix} = \frac{1}{256} * \left(\begin{bmatrix} RR & RG & RB \\ GR & GG & GB \\ BR & BG & BB \end{bmatrix} * \begin{bmatrix} Rin \\ Gin \\ Bin \end{bmatrix} \right)$$

dss-E076

Figure 12-88 shows the color phase rotation macro-architecture.

Figure 12-88. Color Phase Rotation Macro Architecture



12.4.2.5.1.1 Spatial/Temporal Dithering

The spatial/temporal dithering logic can be selected for passive matrix and active matrix panel. The dithering logic is integrated after the color phase rotation and before the TDM and passive matrix units. The spatial/temporal dithering logic can be selected to enhance the quality of the passive matrix and active matrix outputs. The dithering logic can process the pixels over a single frame, two frames, or four frames. In the case of a single frame, only spatial processing is applied. In the case of multiple frames, spatial and temporal processing is applied to the pixels.

- Passive Matrix Technology: The passive matrix display dithering logic path is used. The spatial/temporal dithering logic can be selected. When selected, the pixels are preprocessed by the spatial/temporal dithering logic before the passive matrix display dithering logic. The output format of

the spatial/temporal dithering logic is RGB 12-bit (not configurable).

- Active Matrix Technology: The encoded pixel values are used by spatial/temporal dithering logic to display the data in a lower color depth on the LCD panel. The spatial/temporal dithering algorithm is based on the (x,y) pixel position, the value of removed bits and the frame number. The picture quality is improved when enabling the spatial/temporal dithering logic. When spatial/temporal dithering is not enabled, the three MSBs of the pixel color components are output on the interface data bus if the interface data bus is smaller than the pixel format size. If the interface data bus is wider than the pixel format size, by programming the pixel components replication active/inactive, the MSB is replicated to the LSB of the interface data bus or the LSB is filled with 0s.

12.4.2.5.2 Passive Matrix Display Dithering Logic

- Passive matrix technology

  After the graphics data are merged with the video data from the video layers depending on the transparency status, the result is sent to the color/grayscale space-/time-based dither generator. The monochrome data and each RGB color component are encoded on 4 bits, which are the 4 MSBs of the pixel-encoded component 8-bit value defined by the merge of the graphics data and the video data.

  These 4-bit values are used to select on the 16 intensity levels. The gray/color intensity is controlled by turning individual pixels on and off at varying period rates, making the average time the pixel is off longer than the average time the pixel is on, thus producing more intense grays/colors. The dithering generator also uses the intensity of adjacent pixels in the calculation to give the screen image a smooth appearance. The proprietary dither algorithm is optimized to provide a range of intensity values that matches the visual perception of color/gray graduations.

- Active matrix technology

  The passive matrix dithering logic is always bypassed in active displays.

NOTE: If the interface data bus is smaller than the pixel format size, dithering logic can be enabled. If the interface data bus is wider than the pixel format size, the dithering logic cannot be enabled and replication feature can be used.

12.4.2.5.3 Passive Matrix Display Output FIFO

- Passive matrix technology

  The display controller contains a 2-entry by 8-bit-wide output FIFO used to store pixel data before it is driven out to the LCD pins. Each time a modulated pixel value is output from the dither generator, it is placed into a serial shifter. The shifter can be configured to be 4 or 8 bits wide. Single-panel monochrome screens use either four or eight data lines; single-panel color screens use eight data pins.

- Active matrix technology

  The output FIFO is bypassed in active matrix mode.

12.4.2.5.4 Multiple Cycle Output Format

The pixels after the active matrix display processing are formatted on one or multiple cycles (from one to three cycles). The interface width can be 8-, 9-, 12-, or 16-bit. On three cycles, two pixels can concatenate and send to the panel. When the TDM is disabled, the display controller outputs the pixels using the conventional formats: Passive matrix display/active matrix display monochrome/color.

The following example shows an output configuration based on the interface width (8-bit) and the pixel format output (24-bit) (also see Table 12-29):

- The DSS.DISPC_CONTROL[24:23] TDMCYCLEFORMAT bit field is set to 0x2 (three cycles for one pixel).
- The DSS.DISPC_DATA_CYCLEk (k=0) register is set to 0x00000008 (8 bits from pixel 1 for the first cycle).
- The DSS.DISPC_DATA_CYCLEk (k=1) register is set to 0x00000008 (8 bits from pixel 1 for the second cycle).

- The DSS.DISPC_DATA_CYCLEk (k=2) register is set to 0x00000008 (8 bits from pixel 1 for the third cycle).

Table 12-29. 8-Bit Interface Configuration/24-Bit Mode

	24-Bit Mode		
	1st Cycle	2nd Cycle	3rd Cycle
Data[7]	R0[7]	G0[7]	B0[7]
Data[6]	R0[6]	G0[6]	B0[6]
Data[5]	R0[5]	G0[5]	B0[5]
Data[4]	R0[4]	G0[4]	B0[4]
Data[3]	R0[3]	G0[3]	B0[3]
Data[2]	R0[2]	G0[2]	B0[2]
Data[1]	R0[1]	G0[1]	B0[1]
Data[0]	R0[0]	G0[0]	B0[0]

12.4.2.6 Video Line Buffer

The line buffer size is 1024 x 24-bit. There are six line buffers (1024 x 24-bit) that can be merged into three lines (2048 x 24-bit). Table 12-30 lists the maximum width depending on the TAP configuration and the pixel format.

Table 12-30. Maximum Width Allowed

Vertical Tap	Pixel Format	Maximum Width (Pixels)
3	RGB16	2048
	RGB24	
	YUV422	
5	RGB16	1024
	RGB24	
	YUV422	

12.4.2.7 Synchronized Buffer Update

A synchronization mismatch between the frame buffer and the display refreshes, named tearing effect, can lead to images that appear to be stretched on the screen. To avoid this, a synchronization mechanism is needed between the display controller and the process that updates the buffer. An interrupt is generated when the display reaches a predefined line number. This PROGRAMMEDLINENUMBER interrupt is a level signal and stays active during the programmed line of the display.

12.4.2.8 Rotation

In case of SDRAM buffer, the display controller accesses the encoded pixels in burst, always considering the consecutive data in memory. The rotation engine (VRFB) in the SDRAM scheduler (SDRC) is in charge of translating the addresses from virtual to physical SDRAM addresses (see , *Memory Subsystem*).

NOTE: It is highly recommended to use the VRFB Rotation engine when possible and not the display subsystem DMA engine for rotating frame buffer to have a good performance in the L3 interconnect and SDRAM memory efficiency.

The rotation using the SMS-VRFB rotation engine is supported for BITMAP8, RGB12 (16-bit container) and ARGB16, RGB16, ARGB16, RGB24 (using 32-bit container) and ARGB32 and RGBA32 and YUV422 (YUV2 and YUYV). The formats not supported are BITMAP1, BITMAP2, BITMAP4 and RGB24 (using 24-bit container).

12.4.2.9 Multiple Buffer Support

Users update the base address of the buffer when the update of the working buffer has finished and is ready to be displayed. The register that contains the base address of the buffer is a shadow register that is read by the hardware at the next Vertical Front Porch (VFP).

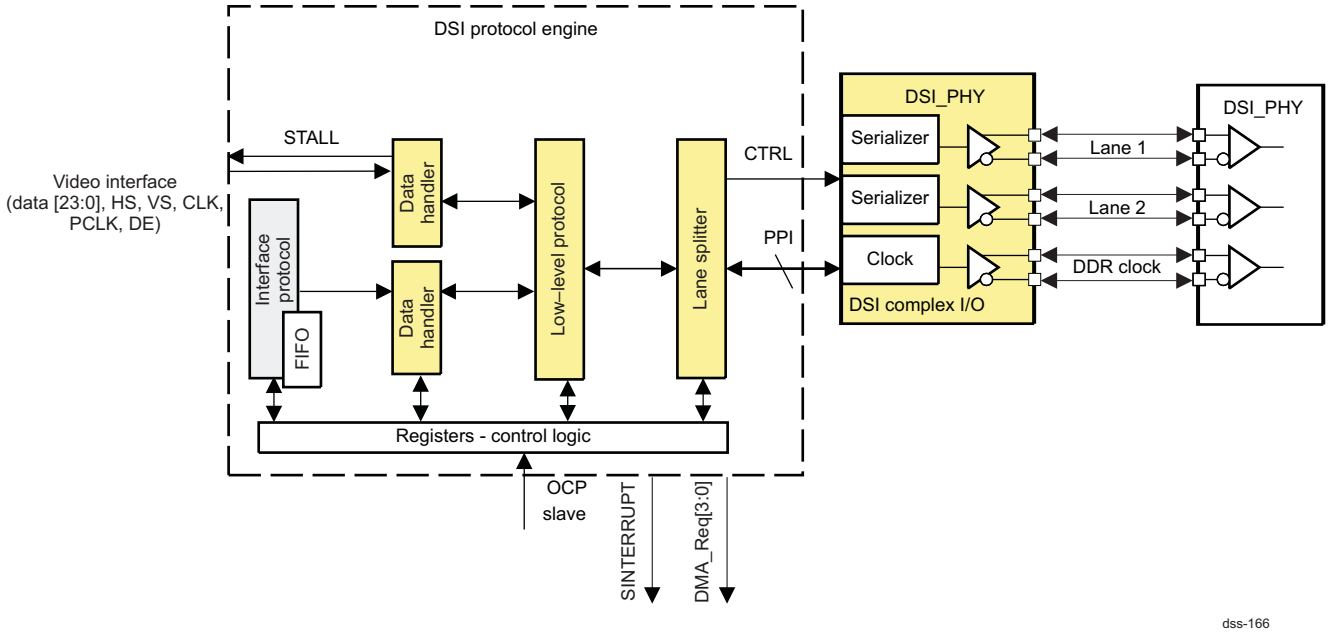
12.4.3 DSI Protocol Engine Functionalities

The DSI protocol engine integrates DSI interface to the display through the DSI DSI_PHY module, L4 interconnect interface and video interface from the display controller. The DSI DSI_PHY or complex I/O module is detailed in Section 12.4.5. The DSI Transmitter (Protocol Engine + PHY) port can be connected to multiple displays using a single DSI host port. The DSI protocol engine controls the DSI PLL Control module detailed in Section 12.4.4. The DSI Transmitter port can be used in video mode or/and command mode.

12.4.3.1 DSI Protocol Architecture

The DSI protocol engine receives data from the video port and/or the L4 interconnect slave port, encapsulates them with the VC ID, generates the ECC and check-sum, and splits the data into byte stream to the DSI_PHY to be sent using the low-speed (LS) or high-speed (HS) protocol. The DSI protocol engine receives data and acknowledge from the display using the same DSI link in case of bidirectional display. Multiple data streams can be interleaved to support multiple panels connected to the same host DSI port. Figure 12-89 details the DSI protocol engine architecture.

Figure 12-89. DSI Protocol Engine

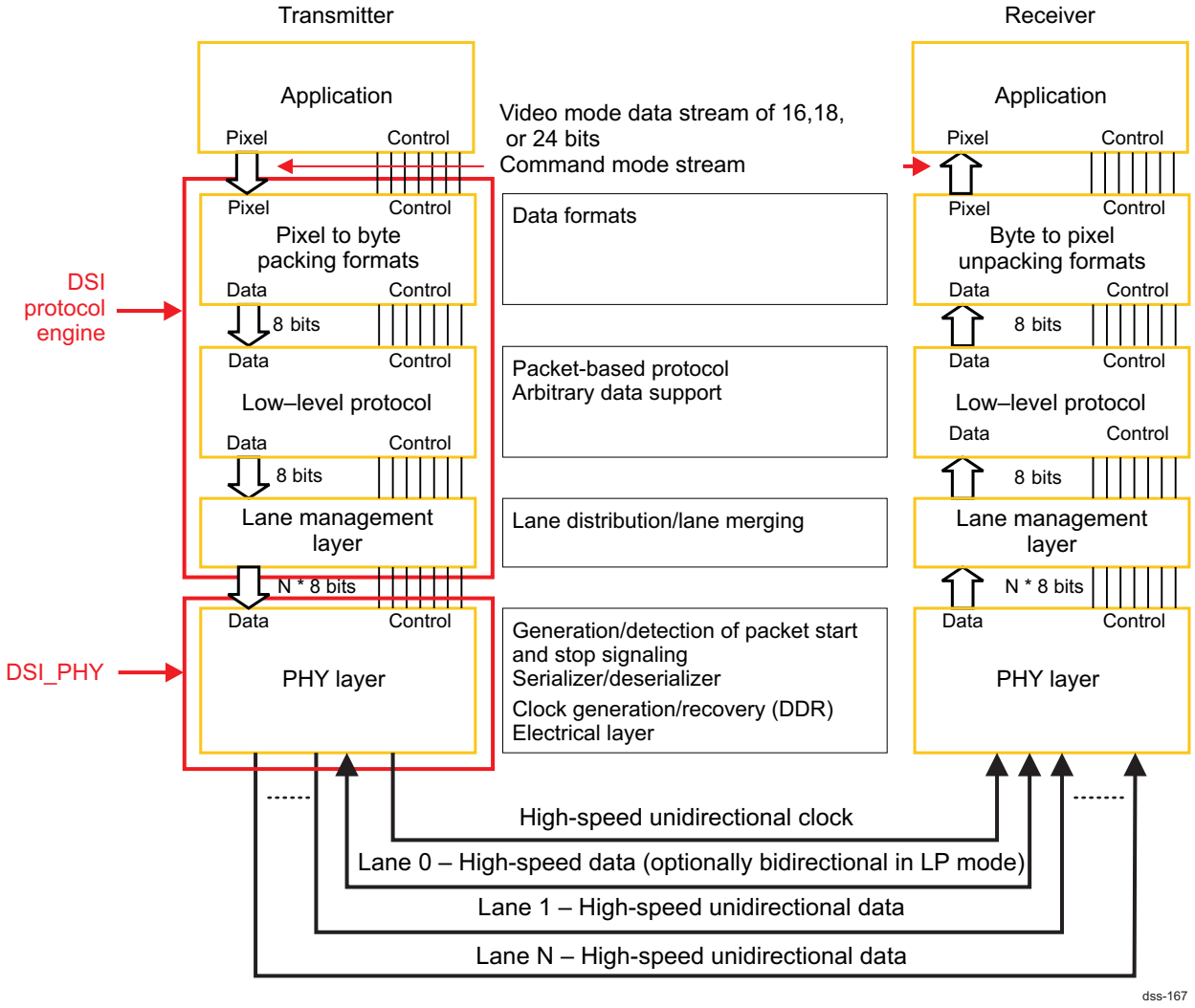


NOTE: The order of the PHY pairs (clock and data lanes) is informative. Each PHY pair can be Clock or Data. The DSI complex I/O receives the configuration for pin order and the differential +/- in a pair from the settings in DSS.DSI_COMPLEXIO_CFG1 register.

The DSI serial interface is a bidirectional differential serial interface with data/clock for the physical layer (configured in unidirectional link in case the display module is only unidirectional). The maximum DSI data transfer capacity is 800 Mbps per channel. The speed of the link can be software configured only when the DSI_PHY is in stop state or in ULPS.

Figure 12-90 shows the DSI transmitter/receiver high-level data flow.

Figure 12-90. DSI Transmitter/Receiver Data Flow



12.4.3.2 Clock Requirements

The serial clock generated by the DSI host and sent to the display can be a continuous clock. The clock lane supports clock transmission even there is no data to send for displays that require continuous clock. It is software programmed through the DSS.DSI_CLK_CTRL[13] DDR_CLK_ALWAYS_ON bit: This bit can be programmed only when the interface is disabled (that is, DSS.DSI_CTRL[0] IF_EN bit set to 0).

The peripheral can use two different kinds of clocks. The first one is the DDR clock provided on the clock lane. The second clock is some transitions on the data lane 1 even if there is no valid data to send using low power mode.

The LP clock (TxClkEsc) frequency provided to the DSI complex I/O is in the range of 67% to 150% of the peripheral Low-Power (LP) clock frequency. It is generated internally by the DSI protocol engine module using the DSI functional clock. The DSI functional clock is divided by 1, 2, 3, up to 8191 using the value programmed in the DSS.DSI_CLK_CTRL[12:0] LP_CLK_DIVISOR bit field. The LP clock generated from DSI functional clock should be in the range of 20 MHz down to 32 KHz. The duty cycle should be 50/50 (tolerance of 45/55 for maximum value). LP clock frequency visible on the pads (DP xor DN) is half the frequency of TxClkEsc.

The DSS.DSI_CLK_CTRL[20] LP_CLK_ENABLE bit is used to enable or disable the clock. When disabled, the value of DSS.DSI_CLK_CTRL[12:0] LP_CLK_DIVISOR bit field is ignored and does not have to be programmed by software users.

12.4.3.2.1 Timing Parameters for an LP to HS Transaction

Figure 12-91 shows the timing requirement when switching the data and clock lane state from LP to HS. Table 12-31 lists the LP to HS timing parameters.

Figure 12-91. LP to HS Timing

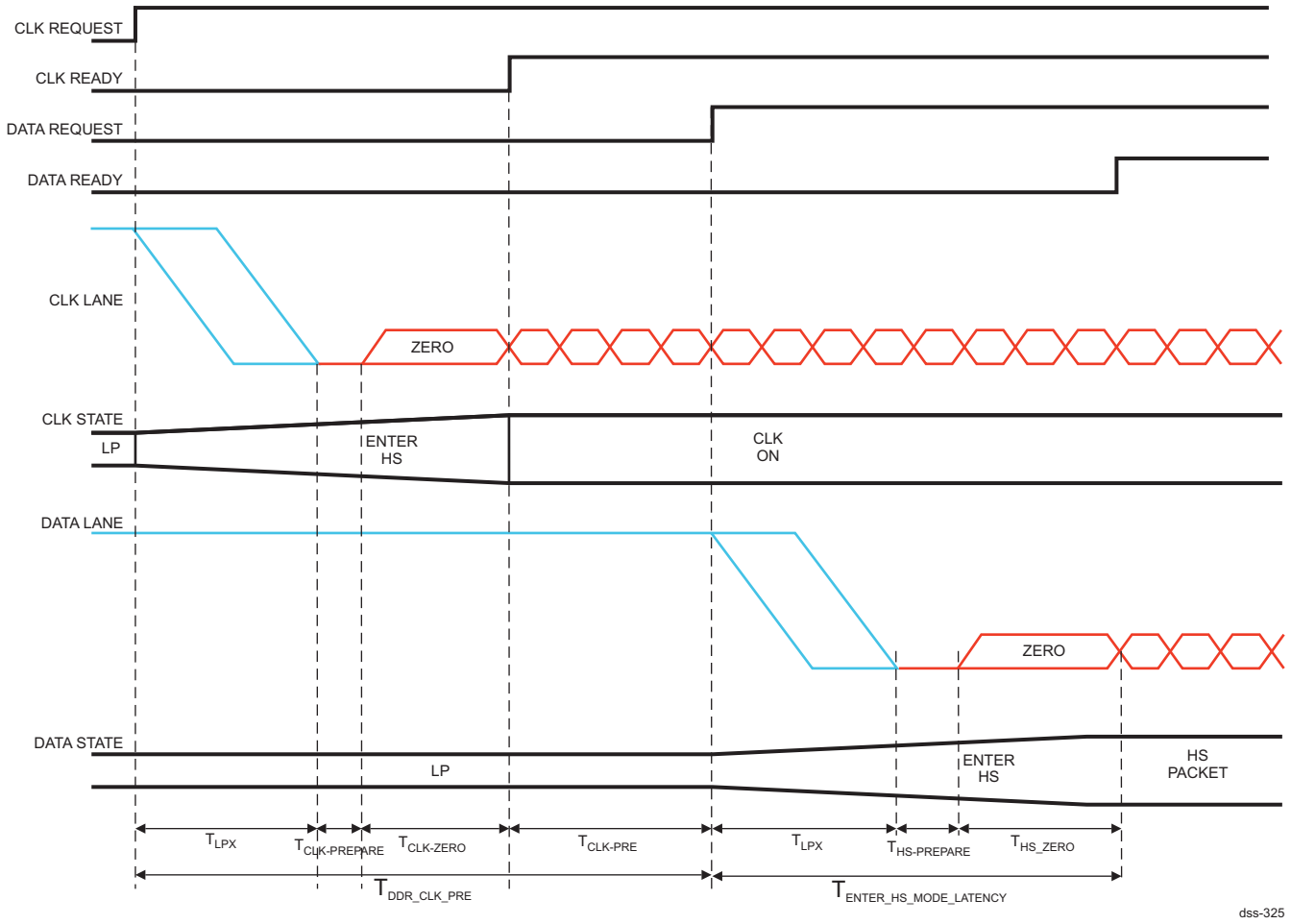


Table 12-31. LP to HS Timing Parameters

Timing	Description	Register
T_{LPX}	Length of any low-power state period. The value set in DSI_PHY_CFG1[20:16] TLPX_HALF bit field is half of the T_{LPX}.	DSI_PHY_CFG1[20:16] TLPX_HALF
$T_{CLK-PREPARE}$	Time to drive the CLK lane to LP-00 state, to prepare for HS clock transmission	DSI_PHY_CFG2[7:0] TCLK_PREPARE
$T_{CLK-ZERO}$	Time to drive the CLK lane to HS-0 state before starting the clock	DSI_PHY_CFG1[7:0]TCLK_ZERO
$T_{CLK-PRE}$	Time that the HS clock must be driven before any associated data lane begins the transition from LP to HS mode $T_{CLK-PRE}$ = DDR_CLK_PRE - T_{LPX} - $T_{CLK-PREPARE}$ - $T_{CLK-ZERO}$	See the MIPI DSI_PHY specification.
$T_{HS-PREPARE}$	Time to drive the data lane to LP-00 state, to prepare for HS packet transmission	DSI_PHY_CFG0[31:24] THS_PREPARE
$T_{HS-ZERO}$	Time to drive the data lane to HS-0 state before the synchronous sequence. $T_{HS-ZERO}$= DSI_PHY_CFG0[23:16] THS_PREPARE_THS_ZERO - $T_{HS-PREPARE}$	DSI_PHY_CFG0[23:16] THS_PREPARE_THS_ZERO

Table 12-31. LP to HS Timing Parameters (continued)

Timing	Description	Register
$T_{DDR\_CLK\_PRE}$	Time between the CLK lane request assertion and the data request assertion to switch the data lanes to HS	DSI_CLK_TIMING[15:8] DDR_CLK_PRE
$T_{ENTER\_HS\_MODE\_LATENCY}$ (1)	Time to enter into HS mode. It is critical that $T_{ENTER\_HS\_MODE\_LATENCY}$ = 1 + DIVROUNDUP (2 * TLPX_HALF, 4) + DIVROUNDUP (THS_PREPARE ,4) + DIVROUNDUP (THS_PREPARE_THS_ZERO - THS_PREPARE + 3, 4) (2)	DSI_VM_TIMING7[31:16] ENTER_HS_MODE_LATENCY

(1) The formula for ENTER_HS_MODE_LATENCY timing is relevant only in video mode. It does not need to be programmed in command mode.
(2) The formula DIVROUNDUP(value, div) is equivalent to ROUNDUP(value/div).

12.4.3.2.2 Timing Parameters for an HS to LP Transaction

Figure 12-92 shows the timing requirement when switching the state of the data and clock lanes from HS to LP. Figure 12-92 lists the HS to LP timing parameters.

Figure 12-92. HS to LP Timing

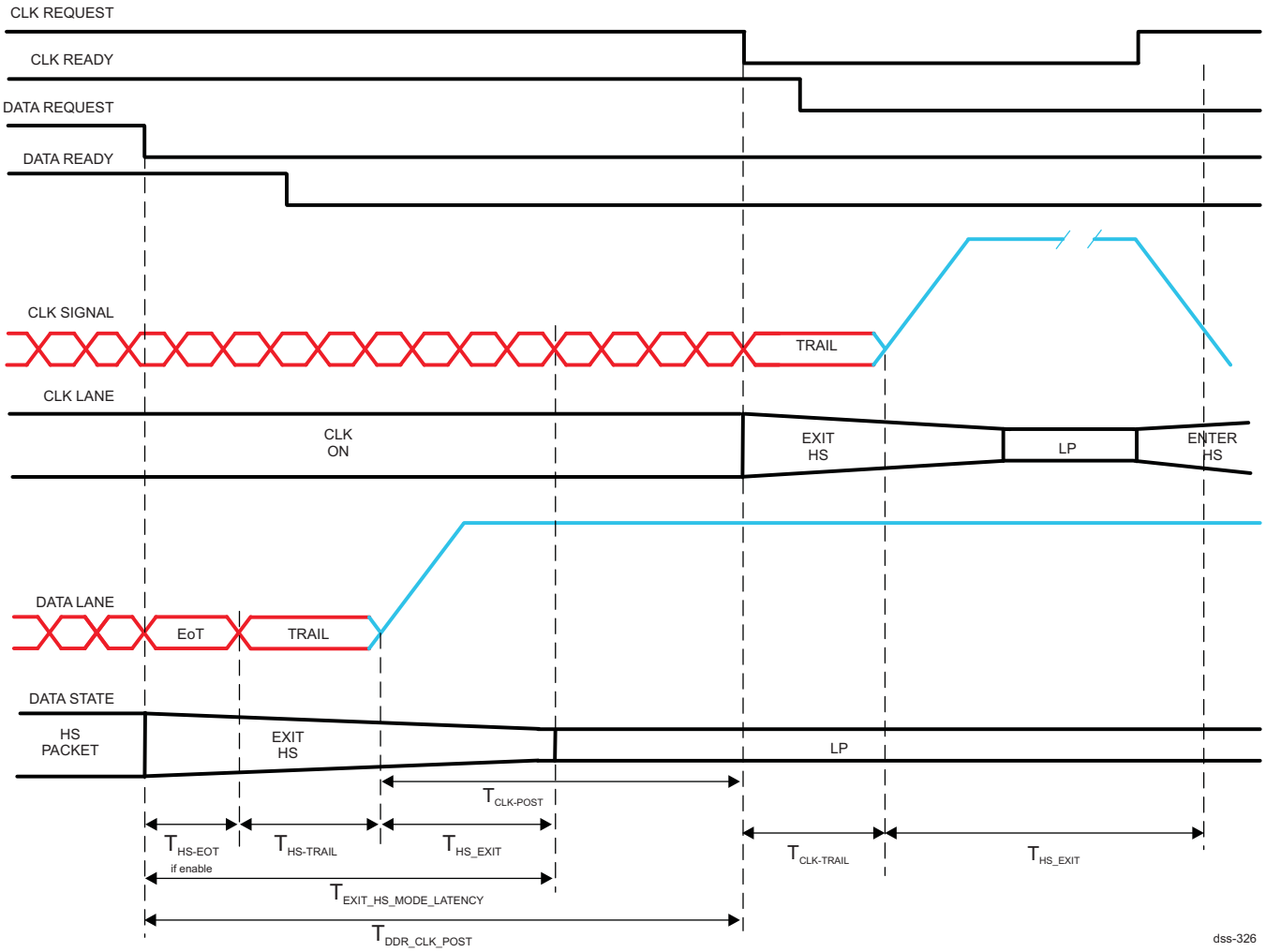


Table 12-32. HS to LP Timing Parameters

Timing	Description	Register
T_{HS-EOT}	If EoT is enabled, a delay is added to EXIT_HS_MODE_LATENCY to send the EoT packet. The EoT period depends on the number of data lanes, and is expressed with the following formula: THS-EOT = DIVROUNDUP (4, NB_DATA_LANES). Thus: 1 data lane = 4 DDR clocks 2 data lanes = 2 DDR clocks	
$T_{HS-TRAIL}$	Time to drive flipped differential state after last payload data bit of a HS transmission burst	DSI_PHY_CFG0[15:8] THS_TRAIL
$T_{HS-EXIT}$	Time to drive data lane to LP-11 state, after HS burst	DSI_PHY_CFG0[7:0] THS_EXIT
$T_{CLK-POST}$	Time that the transmitter must continue sending HS clock after the last associated data lane has transitioned to LP mode	See the MIPI DSI-PHY specification.
$T_{CLK-TRAIL}$	Time to drive HS differential state after last payload clock bit of a HS transmission burst	DSI_PHY_CFG1[15:8]TCLK_TRAIL
$T_{DDR\_CLK\_POST}$	Time between the data lane request deassertion and the CLK request deassertion to switch the data lanes into LP mode. The DDR_CLK_POST value must follow the rule: $T_{DDR\_CLK\_POST} \geq T_{HS-TRAIL} + T_{HS-EOT} + T_{CLK-POST}$	DSI_CLK_TIMING[7:0] DDR_CLK_POST
$T_{EXIT\_HS\_MODE\_LATENCY}$ (1)	Time to exit in HS mode. It is critical that $T_{EXIT\_HS\_MODE\_LATENCY}$ = DIVROUNDUP(($T_{HS-TRAIL}$ + $T_{HS-EXIT}$),4) + 1 + T_{HS-EOT} (2)	DSI_VM_TIMING7[15:0] EXIT_HS_MODE_LATENCY

(1) The formula for EXIT_HS_MODE_LATENCY timing is relevant only in video mode. It does not need to be programmed in command mode.

(2) The formula DIVROUNDUP(value, div) is equivalent to ROUNDUP(value/div).

12.4.3.2.3 Extra LP Transitions

Some DSI receivers require extra clock cycles in LP mode to process the data. The DSI protocol engine can be programmed to send automatically one NULL long packet. It applies only when no more data are ready to be sent from the internal FIFO to the peripheral on the last low speed transfer. The same value is used for all the VCs sending packets in low speed mode.

The size of the payload is defined by the DSS.DSI_CLK_CTRL[17:16] LP_CLK_NULL_PACKET_SIZE bit field. The header value depends on the VC ID and the size of the payload as detailed in Table 12-33 and Table 12-34.

Table 12-33. Extra NULL Packet Header

Virtual Channel ID	Payload size (DSI_CLK_CTRL[17:16] LP_CLK_NULL_PACKET_SIZE)	Header (1st Byte)	Header (2nd Byte): WC LSB	Header (3rd Byte): WC MSB	Header (ECC)
0x0	0	0x9	0x0	0x0	0x9
	1		0x1		0x13
	2		0x2		0x2F
	3		0x3		0x35
0x1	0	0x49	0x0		0x1F
	1		0x1		0x05
	2		0x2		0x39
	3		0x3		0x23
0x2	0	0x89	0x0		0x10
	1		0x1		0x0A
	2		0x2		0x36
	3		0x3		0x2C
	0		0x0		0x06

Table 12-33. Extra NULL Packet Header (continued)

Virtual Channel ID	Payload size (DSI_CLK_CTRL[17:16] LP_CLK_NULL_PACKET_SIZE)	Header (1st Byte)	Header (2nd Byte): WC LSB	Header (3rd Byte): WC MSB	Header (ECC)
0x3	1	0xC9	0x1		0x1C
	2		0x2		0x20
	3		0x3		0x3A

Table 12-34. Extra NULL Packet Payload

Payload size (DSI_CLK_CTRL[17:16] LP_CLK_NULL_PACKET_SIZE)	Payload (1st byte)	Payload (2nd byte)	Payload (3rd byte)	Payload (CRC) LSB	Payload (CRC) MSB
0	NA	NA	NA	0xFF	0xFF
1	0	NA	NA	0x87	0x0F
2	0	0	NA	0xB8	0xF0
3	0	0	0	0x33	0x39

NOTE:

- In Table 12-33 and Table 12-34, both ECC and checksum are enabled.
- NA means not available.

12.4.3.3 DSI Transfer Modes

There are two transfer modes supported by the DSI module:

- Video mode (VM): Pixels are received from the video port, there are some real time constraints (pixels must be sent at the pixel frequency required by the display module) for sending the data to the display;
- Command mode (CM): Pixels can be received from the video port or from the L4 interconnect, there are no real time constraints except that TE should be avoided by starting the transfer at the right time during scan of the display and should be fast enough.

12.4.3.3.1 Video Mode

The video mode refers to the MIPI DPI 1.0 standard. The sync events and pixels should be sent according to the display mode timings. Data are received from the video port. The display controller is in charge of fetching the data from the system memory and providing the data to the DSI protocol engine using the video port. The short packets used for the sync event are using precalculated 32-bit values. The long packets are constructed using the header defined in DSS.DSI_VCn_LONG_PACKET_HEADER registers.

12.4.3.3.2 Command Mode

The command mode refers to the MIPI DCS standard. The commands, parameters and pixels are sent to the display module with limited real time constraints (as defined in Section 12.4.3.3.1). The pixels can be provided on the video port by the display controller or on the L4 interconnect port.

NOTE: In DSI command mode, the display controller must be configured in stall mode by setting the DSS.DISPC_CONTROL[11] STALLMODE bit to 1.

The DSS.DSI_VCn_LONG_PACKET_HEADER registers are used for the header of long packets, the DSS.DSI_VCn_SHORT_PACKET_HEADER registers are used for the short packets.

The error correction code (ECC) can be provided while writing the ECC value directly into the DSS.DSI_VCn_LONG_PACKET_HEADER and DSS.DSI_VCn_SHORT_PACKET_HEADER registers. The DSS.DSI_VCn_CTRL[8] ECC_TX_EN bit indicates if the ECC value should be calculated or if the value written in the register should be used instead for command and video modes. In case of synchronization short packets for video mode, since the hardware generates the short packets without using DSS.DSI_VCn_SHORT_PACKET_HEADER registers, if the DSS.DSI_VCn_CTRL[8] ECC_TX_EN bit is set to 1, the ECC is calculated otherwise the value zero is used. The feature is used to generate incorrect ECC for debug purpose and to ease the check for the link and peripheral error detection and correction.

For the payload, the DSS.DSI_VCn_LONG_PACKET_PAYLOAD registers are used. Each 32-bit PAYLOAD data is written into the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register from the MPU subsystem or system DMA. It is buffered to be able to send packets with higher rate than the L4 interconnect frequency can provide. The word count defined in the DSS.DSI_VCn_LONG_PACKET_HEADER registers is used to determine the number of bytes to be sent using the DSS.DSI_VCn_LONG_PACKET_PAYLOAD registers. The write into the DSS.DSI_VCn_LONG_PACKET_HEADER registers is required before accessing the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register. The hardware should be able to extract the length of the payload and be able to discard extra data sent using the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register. The hardware takes into account the write into the DSS.DSI_VCn_LONG_PACKET_HEADER register only if the VC is enabled otherwise the write is ignored by hardware.

In the case of pixels received on the video port, only the DSS.DSI_VCn_LONG_PACKET_HEADER register is used. The video port pixels are used for the payload. When the pixel data is coming from the display controller video port, the DSI protocol can add a DCS command byte between the packet header and pixel data by setting the DSS.DSI_CTRL[24] DCS_CMD_ENABLE bit to 0x1. The value will be either 0x2c (write_memory_start) by setting the DSS.DSI_CTRL[25] DCS_CMD_CODE bit to 0x1, or 0x3c (write_memory_continue) by setting the DSS.DSI_CTRL[25] DCS_CMD_CODE bit to 0x0.

When transmitting RGB 16-BPP data, the DSS.DSI_CTRL[26] RGB565_ORDER bit must be set to 0x1 to maintain the pixel byte order as in video mode.

A 2-line ping-pong buffer is implemented to allow the DSI protocol engine to store incoming pixels from the display controller through the video port while sending the DSI formatted frame to the DSI_PHY. The ping-pong buffer is supported in command mode, provided the size of the packet defined in the header register is less than the size of each line buffer (768 *32 bits). If the size of the packet is greater than the size of the line buffer, the ping-pong mechanism cannot be used (both lines are used as a single line).

The ping-pong buffer status can be checked by the DSI_VCn_CTRL[14] PP_BUSY bit.

- When PP_BUSY equals 1, the ping-pong buffer is active and the line buffers are not ready to receive data; therefore, the user cannot update a new header.
- When PP_BUSY equals 0, at least one line buffer is empty; therefore, the user can update a new header. PP-BUSY is then set to 0x1. If both line buffers are empty, the user can write two headers, one following the other. PP_BUSY remains at 0x0 after the first header is written, and is set to 0x1 after second header is written.

An IRQ is available to allow software to update header on events. The IRQ is enabled by setting the DSI_VCn_IRQENABLE[8] PP_BUSY_CHANGE_IRQ bit to 0x1, and its status is accessible on the DSI_VCn_IRQSTATUS[8] PP_BUSY_CHANGE_IRQ bit.

12.4.3.3.3 Video + Command Modes

The two modes can be interlaced to send two DSI streams to two types of panels: Video or command types. The number of concurrent video stream is limited to a single one. The number of concurrent command mode streams is limited to 4 when there is no video stream and 3 otherwise. In case there is one DSI stream using video mode, the command mode pixels should be provided on the L4 interconnect only.

12.4.3.3.4 Burst Modes

- Frequency-burst mode The frequency-burst mode is used to reduce the high-speed (HS) period by increasing the clock frequency on the DSI link. It allows in some case, the power consumption

reduction of the link. The non-HS period used typically to drive the main panel can be used to send data to the secondary panel or to allow feedback (acknowledge) from the primary and secondary panels. The DSI protocol engine needs to buffer a full line before sending the HS packets for the line. A double buffering mechanism is required to be able to send a line while the following one is being received on the video port.

- Transparent-burst mode The transparent-burst mode is used by increasing the pixel clock frequency generated by the display controller with in addition an increase of the horizontal blanking period.

12.4.3.3.5 Interleaving Mode

Video mode can output command mode packets, which are provided to DSI through the L4 interconnect, during the blanking periods of the video stream sequence on the PPI link. These command mode packets can be programmed as high-speed packets or low-power packets.

During a video stream sequence on the PPI link, four types of gap exist:

- BLLP gap: Blanking period during VSA, VBP, and VFP lines
- HSA gap: Blanking period during VACT lines; always between HS and HE short packet
- HBP gap: Blanking period during VACT lines; always between HS/HE short packet and data pixel long packet
- HFP gap: Blanking period during VACT lines; always between data pixel long packet and the end of the current VACT line

To perform interleaving in a particular gap, video mode must be set to go into low-power state during the blanking gap. Each type of gap has separate configurable register bits that determine whether a blanking long packet will be sent or the link will go into low-power state during the gap on the PPI link. If low-power state is set during a gap, the DSI module performs interleaving during that period.

Two set of registers are available for:

- High-speed interleaving (when high-speed command mode packets must be sent during a video stream on the PPI link)
- Low-power interleaving (when low-power command mode packets must be sent during a video stream on the PPI link)

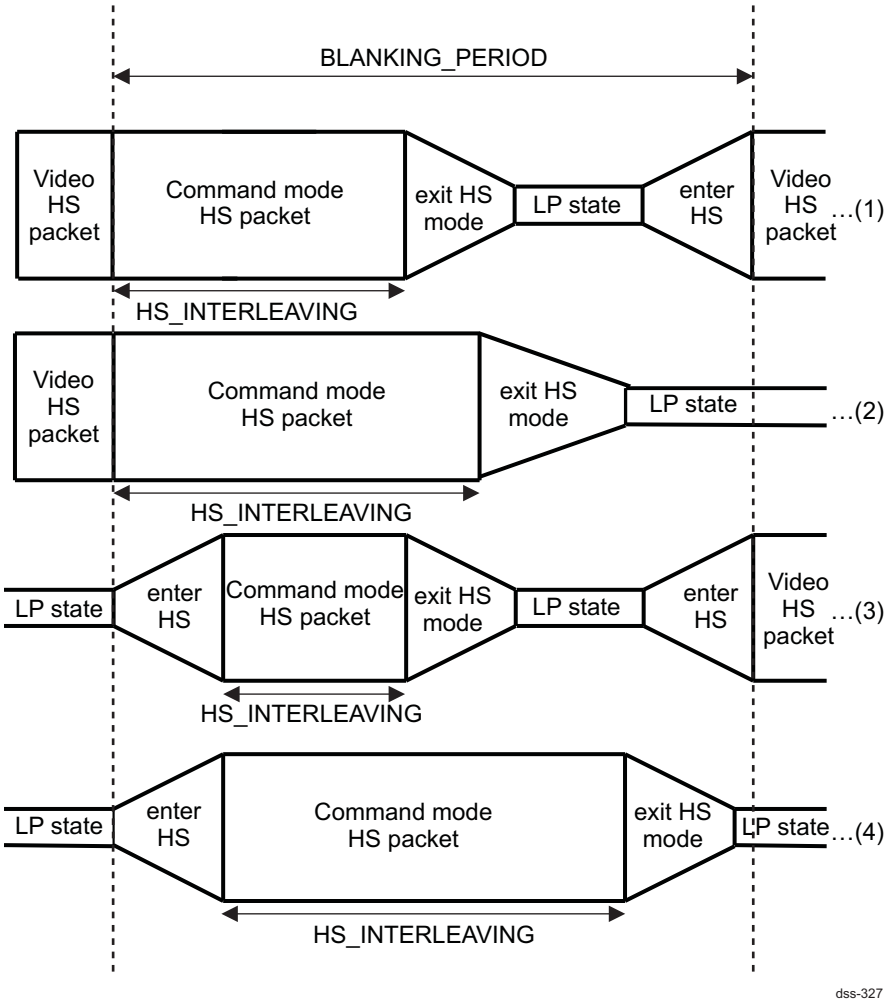
12.4.3.3.5.1 HS Command Mode Interleaving Programming Model

Figure 12-93 shows the various HS mode scenarios in interleaving mode during a blanking gap. For each type of blanking gap, a dedicated bit field determines the number of TxByteClkHS clock cycles used for interleaving in HS command mode packets.

- The BL_HS_INTERLEAVING[31:16] DSI_VM_TIMING6 bit field defines the number of TxByteClkHS clock cycles used to interleave HS command mode packets during a BLLP gap.
- The HBP_HS_INTERLEAVING[7:0] DSI_VM_TIMING4 bit field defines the number of TxByteClkHS clock cycles used to interleave HS command mode packets during an HBP gap.
- The HFP_HS_INTERLEAVING[15:8] DSI_VM_TIMING4 bit field defines the number of TxByteClkHS clock cycles used to interleave HS command mode packets during an HFP gap.
- The HSA_HS_INTERLEAVING[23:16] DSI_VM_TIMING4 bit field defines the number of TxByteClkHS clock cycles used to interleave HS command mode packets during an HSA gap.

These programmable values must be programmed to satisfy the timings for the clock and data lane to enter and exit HS mode latency. According to the scenario, different equations must be considered when calculating the register values.

Figure 12-93. HS Command Mode Interleaving



NOTE: For calculations and equations, the following abbreviations are used: EXIT_CLK_HS_MODE represents the exit HS mode latency for the clock lane. There is no dedicated register for this value but the programmer must know this value for further calculations.
EXIT_CLK_HS_MODE = $T_{CLK\text{-}TRAIL}$ + $TH_{S\text{-}EXIT}$

For the following equations, BLANKING_PERIOD represents the BLLP, HSA, HBP, or HFP blanking periods. The HS_INTERLEAVING period represents the maximal period HS command mode packets. Its value is set in the BL_HS_INTERLEAVING, HSA_HS_INTERLEAVING, HBP_HS_INTERLEAVING, or HFP_HS_INTERLEAVING registers, depending on the blanking type.

In each scenario, two calculations are present, depending on the value of ddr_clk_always_on.

- ddr_clk_always_on = 1: Clock lane is always active.
- ddr_clk_always_on = 0: Clock lane is activated only when there are HS packets to be sent on the PPI link.
- Scenario 1: The gap for interleaving starts and ends with a regular video stream HS packet.
  - ddr_clk_always_on = 1

    HS_INTERLEAVING = BLANKING_PERIOD – (EXIT_HS_MODE_LATENCY + max{ ENTER_HS_MODE_LATENCY, 2} + 1)
  - ddr_clk_always_on = 0

    HS_INTER1 = BLANKING_PERIOD – (EXIT_HS_MODE_LATENCY + max{

ENTER_HS_MODE_LATENCY, 2} + 1)

HS_INTER2 = BLANKING_PERIOD – (DDR_CLK_POST + EXIT_CLK_HS_MODE + DDR_CLK_PRE+ ENTER_HS_MODE_LATENCY + 1)

HS_INTERLEAVING = min{ HS_INTER1, HS_INTER2}

- Scenario 2: The gap for interleaving starts with a regular video stream HS packet and ends in LP state.
  - ddr_clk_always_on = 1

    HS_INTERLEAVING = BLANKING_PERIOD – (EXIT_HS_MODE_LATENCY + 3)
  - ddr_clk_always_on = 0

    HS_INTER1 = BLANKING_PERIOD – (EXIT_HS_MODE_LATENCY + 3)

    HS_INTER1 = BLANKING_PERIOD – (DDR_CLK_POST + EXIT_CLK_HS_MODE + 3)

    HS_INTERLEAVING = min{ HS_INTER1, HS_INTER2}
- Scenario 3: The gap for interleaving starts with the LP state and ends with a regular video stream HS packet.
  - ddr_clk_always_on = 1

    HS_INTERLEAVING = BLANKING_PERIOD – (ENTER_HS_MODE_LATENCY + EXIT_HS_MODE_LATENCY + max{ ENTER_HS_MODE_LATENCY, 2} + 1)
  - ddr_clk_always_on = 0

    HS_INTER1 = BLANKING_PERIOD – (DDR_CLK_PRE + ENTER_HS_MODE_LATENCY + EXIT_HS_MODE_LATENCY + max{ ENTER_HS_MODE_LATENCY, 2} + 1)

    HS_INTER2 = BLANKING_PERIOD – (DDR_CLK_PRE + ENTER_HS_MODE_LATENCY + DDR_CLK_POST + EXIT_CLK_HS_MODE + DDR_CLK_PRE+ ENTER_HS_MODE_LATENCY + 1)

    HS_INTERLEAVING = min{ HS_INTER1, HS_INTER2})
- Scenario 4: The gap for interleaving starts with the LP state and ends with a regular video stream HS packet.
  - ddr_clk_always_on = 1

    HS_INTERLEAVING = BLANKING_PERIOD – (ENTER_HS_MODE_LATENCY + EXIT_HS_MODE_LATENCY + 3)
  - ddr_clk_always_on = 0

    HS_INTER1 = BLANKING_PERIOD – (DDR_CLK_PRE+ ENTER_HS_MODE_LATENCY + EXIT_HS_MODE_LATENCY + 3)

    HS_INTER2 = BLANKING_PERIOD – (DDR_CLK_PRE+ ENTER_HS_MODE_LATENCY + DDR_CLK_POST + EXIT_CLK_HS_MODE + 1)

    HS_INTERLEAVING = min{ HS_INTER1, HS_INTER2}

12.4.3.3.5.2 LP Command Mode Interleaving Programming Model

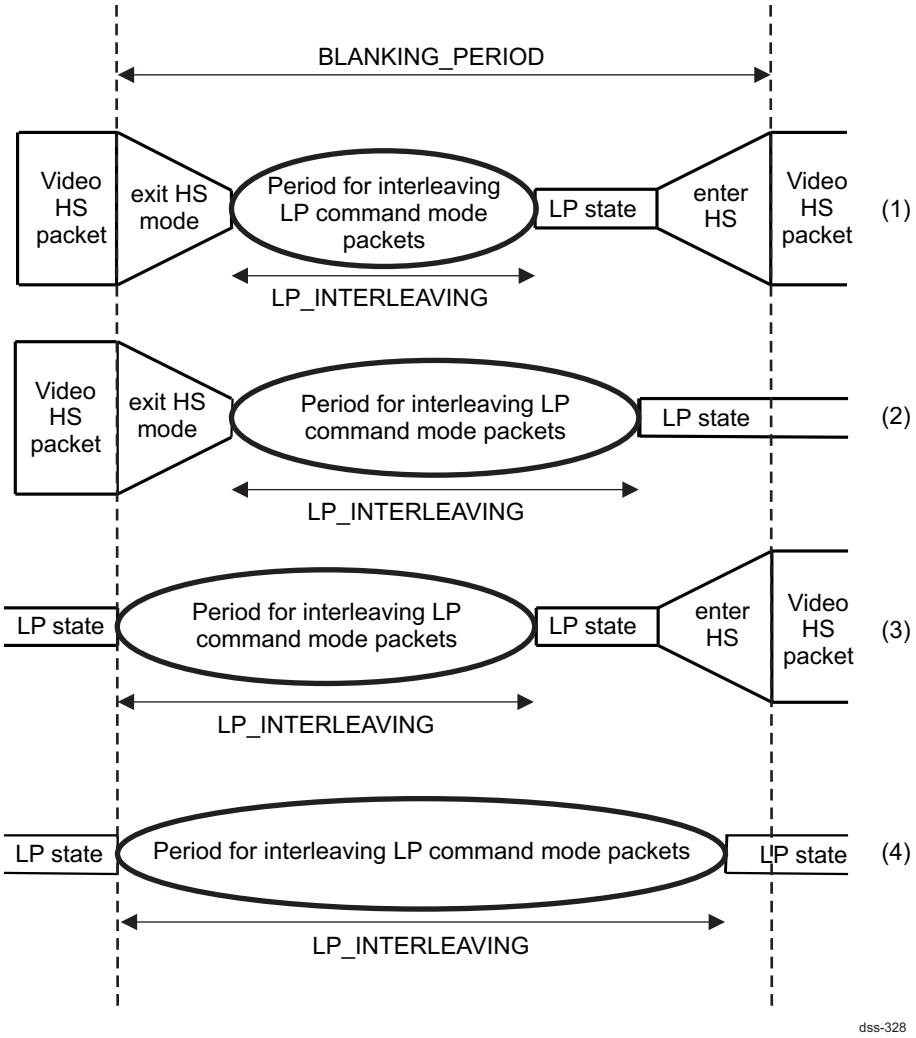
Figure 12-94 shows the various LP mode scenarios in interleaving mode during a blanking gap. For each type of blanking gap, a dedicated bit field determines the number of bytes of LP command mode packets that can be sent during a blanking period.

- BL_LP_INTERLEAVING bit field DSI_VM_TIMING6[15:0] defines the number of bytes of LP command mode packets that can be sent during a BLLP gap.
- HBP_LP_INTERLEAVING bit field DSI_VM_TIMING5[7:0] defines the number of bytes of LP command mode packets that can be sent during an HBP gap.
- HFP_LP_INTERLEAVING bit field DSI_VM_TIMING5[15:8] defines the number of bytes of LP command mode packets that can be sent during an HFP gap.
- HSA_LP_INTERLEAVING bit field DSI_VM_TIMING5[23:16] defines the number of bytes of LP command mode packets that can be sent during an HSA gap.

These programmable values must be programmed to satisfy the timings for clock and data lane enter and exit LP mode latency. Clock lane timings do not affect LP command mode interleaving, because the clock lane can be controlled separately, compared with the data lane high-speed and low-power mutually exclusive control. Clock lanes can be in high-speed mode while the data lanes are in high-speed data transfer mode, low-power data transfer mode, or in low-power state.

According to this scenario, different equations must be considered for calculating register values.

Figure 12-94. LP Command Mode Interleaving



For the following equations, BLANKING_PERIOD represents the BLLP, HSA, HBP, or HFP blanking periods. The LP_INTERLEAVING period represents the maximal period in LP command mode packets. Its value is set in the BL_LP_INTERLEAVING, HSA_LP_INTERLEAVING, HBP_LP_INTERLEAVING, or HFP_LP_INTERLEAVING registers, depending on the blanking type.

ALLOWED_HSBYTE_CLOCKS_FOR_LP represents the number of TxByteClkHS clock cycles during which LP interleaving can appear.

To calculate the LP_INTERLEAVING value:

1. Calculate how many TxByteClkHS clock cycles can be reserved for LP interleaving during the appropriate blanking video mode gap.
2. Calculate the LP_INTERLEAVING value according to the results of Step 1.

Step 1:

- Scenario 1: The gap for interleaving starts and ends with a regular video stream HS packet.

ALLOWED_HSBYTE_CLOCKS_FOR_LP = BLANKING_PERIOD – (EXIT_HS_MODE_LATENCY + max{ ENTER_HS_MODE_LATENCY, 2} + 1)

- Scenario 2: The gap for interleaving starts with a regular video stream HS packet and ends in LP state.

  ALLOWED_HSBYTE_CLOCKS_FOR_LP = BLANKING_PERIOD – (EXIT_HS_MODE_LATENCY + 1)
- Scenario 3: The gap for interleaving starts with the LP state and ends with a regular video stream HS packet.

  ALLOWED_HSBYTE_CLOCKS_FOR_LP = BLANKING_PERIOD – (max{ ENTER_HS_MODE_LATENCY, 2} + 1)
- Scenario 4: The gap for interleaving starts with the LP state and ends with a regular video stream HS packet.

  ALLOWED_HSBYTE_CLOCKS_FOR_LP = BLANKING_PERIOD – 1

After finishing Step 1, the time period available for LP interleaving is known:

$T_{lp\_available}$ = ALLOWED_HSBYTE_CLOCKS_FOR_LP *T TxByteClkHS

Step 2:

The resulting value must be programmed in the appropriate video mode register for LP interleaving.

$$LP\_INTERLEAVING < \left\lfloor \frac{\frac{Tlp\_available - 8*Thsbyte\_clk - 5*Tdsif\_clk}{Ttxclkesc}}{16} - 26 \right\rfloor$$

dss-E124

TxByteClkHS: Period of HS byte clock of DSI_PHY module

Tdsif_clk: Period of DSI functional clock

Ttxclkesc: Period of LP transmit escape clock

12.4.3.4 Power Management

The DSI protocol engine implements an handshake protocol on its L4 interconnect port with the PRCM. The DSI protocol engine provides a clock gating signal CIO_CLK_ICG to gate the L3 interface clock (L3_ICLK) provided by the PRCM to the DSI complex I/O. It allows reduction of the power consumption of the DSI complex I/O while the DSI link is not in used. To gate the L3_ICLK clock at DSI complex I/O level, set the DSS.DSI_CLK_CTRL[14] CIO_CLK_ICG bit to 1.

12.4.3.5 Serial Configuration Port (SCP) Interface

The SCP interface is used to transfer register values from the DSI protocol engine to the DSI PLL Control module and to the DSI complex I/O. It spends several cycles to serialize the data to be sent. Software users should take into account the delay in processing the transfer of the data from/to the slave port to/from the module.

12.4.3.5.1 Shadowing Register

The two first SCP registers for the DSI complex I/O address map should be implemented as shadow registers. The shadowing mechanism is enabled/disabled using the DSS.DSI_COMPLEXIO_CFG1[31] SHADOWING bit:

- When setting the DSS.DSI_COMPLEXIO_CFG1[31] SHADOWING bit to 1, the transfer of the values from the two first L4 interconnect port registers into the two first registers of the DSI complex I/O (DSS.DSI_PHY_CFG0 and DSS.DSI_PHY_CFG1) is done only when the DISPC_UPDATE_SYNC signal from the display controller is active and the DSS.DSI_COMPLEXIO_CFG1[30] GOBIT is set to 1. If there is no pending update for the two registers, when the DISPC_UPDATE_SYNC signal is asserted, the DSS.DSI_COMPLEXIO_CFG1[30] GObit is reset by hardware and there is no SCP transfer.
  - If there is only one register to update, only the corresponding new value is transferred. The second register in the DSI complex I/O is not updated. When the transfer is completed, the

DSS.DSI_COMPLEXIO_CFG1[30] GOBIT is reset by hardware.

– If the two registers need to be updated, the order of the transfer is first the register with lower address and then the second one. When the transfers are completed, the DSS.DSI_COMPLEXIO_CFG1[30] GOBIT is reset by hardware.

When there is an on-going transfer (read or write) to any SCP register, the transfer should complete prior to start the update of shadowing registers.

- When unsetting the DSS.DSI_COMPLEXIO_CFG1[31] SHADOWING bit to 0, if the transfer into the two first DSI complex I/O registers has already started, it should be finished

NOTE: When reading the shadow registers, the local value stored in the DSI protocol engine is returned if the update is pending; otherwise, the values stored in the DSI complex I/O are returned.

12.4.3.5.2 Busy Signal

The signal SCPBusy indicates that there is still some activity using the SCPClk provided by the PRCM. The SCPClk clock is the DSS_L3_ICLK clock.

12.4.3.6 Power Control

The DSI protocol engine can control and send power commands for both DSI complex I/O and DSI PLL controller modules.

12.4.3.6.1 Complex I/O Power Control Commands

12.4.3.6.1.1 Complex I/O Power Control Commands

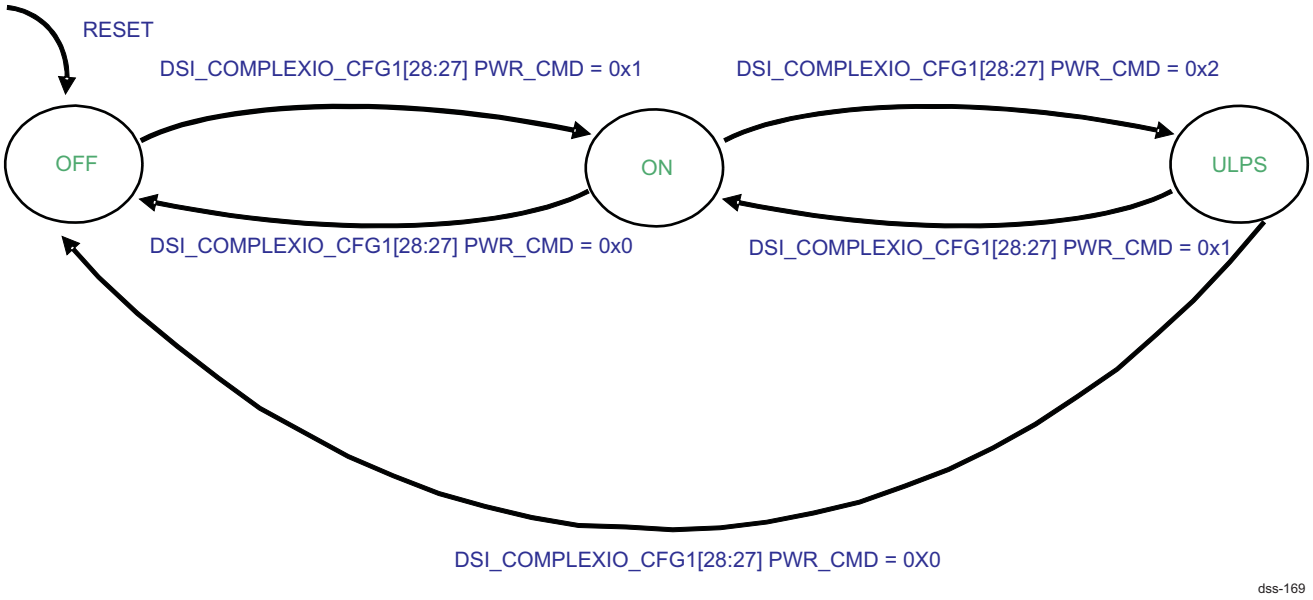
The DSI complex I/O can be set into three modes:

- OFF: In this power state, the complete DSI_PHY circuit is powered down. The internal LDO is OFF.
- ON: In this power state, the complete DSI_PHY circuit is powered on and functional.
- ULPS: In this power state, the ULPS exit detection circuit power switch is ON for the lanes which are in receive ULPS mode. For the lanes which are in transmit ULPS mode, the circuitry for weak pull-down is ON. The ultralow-power state should only be used when all the three lanes are in ULPS (transmit or receive).

12.4.3.6.1.2 Complex I/O Power FSM

Figure 12-95 describes the power control FSM to control the power state of the complex I/O.

Figure 12-95. Complex I/O Power FSM



The PwrCmdOff, PwrCmdUlp and PwrCmdOn commands control the state transition of the DSI complex I/O. Software users should set the DSS.DSI_COMPLEXIO_CFG1[28:27] PWR_CMD bit field to ask for a state change. The allowed transitions are: OFF -> ON and ON -> ULP and ULP -> OFF. The DSS.DSI_COMPLEXIO_CFG1[26:25] PWR_STATUS bit field gives a status on the current state of the DSI complex I/O.

CAUTION

- In automatic mode, the software should ensure that the DSI complex I/O in the ON mode (that is, ON command already sent) before sending requests to the complex I/O.
- In a command request to change to a state which is the current one (acknowledge has been received), the command is ignored (nothing is sent to the DSI complex I/O).
- To change state to ULP state, users should ensure that all the three ULPSActiveNot signals are low. The ULPSActiveNot_ALL0_IRQ interrupt can be used by software users to determine the state of the ULPSActiveNot signals. The change from ULP to ON state is required before starting the ULP status exit sequence (refer to Section 12.4.3.7.1 for details).

12.4.3.6.2 DSI PLL Power Control Commands

The DSI PLL controller module can be set into four modes:

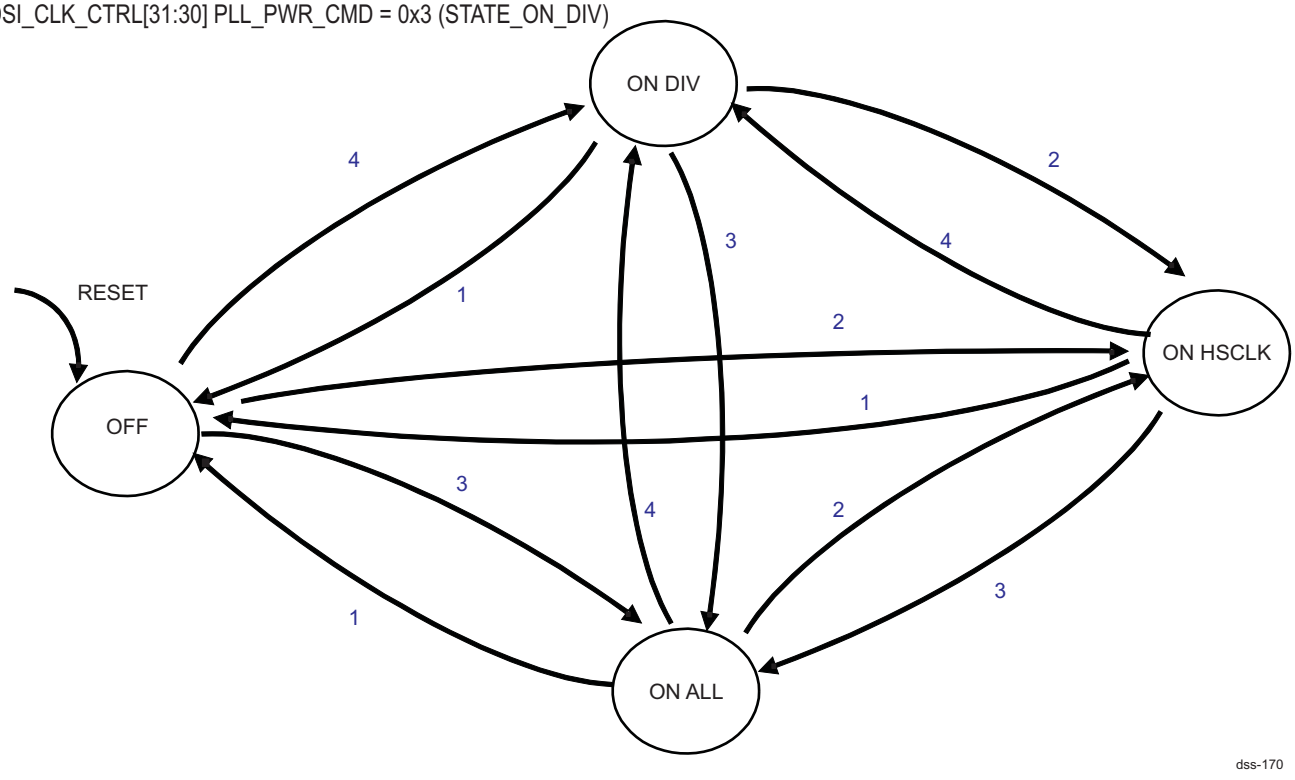
- OFF: The DSI PLL and HSDIVIDER are OFF.
- ON ALL: Both DSI PLL and HSDIVIDER are ON. The HS_CLK clock is provided to the DSI complex I/O and the second clock output is provided to the HSDIVIDER.
- ON HSCLK: The DSI PLL is ON. The HSDIVIDER is OFF. The HS_CLK clock is provided to the DSI complex I/O but the second clock output is not provided to the HSDIVIDER.
- ON DIV: Both DSI PLL and HSDIVIDER are ON. The HS_CLK clock is not provided to the DSI complex I/O but the second clock output is provided to the HSDIVIDER.

12.4.3.6.2.1 DSI-PLL Power FSM

Figure 12-96 shows the DSI PLL power FSM.

Figure 12-96. DSI PLL Power FSM

1 = DSI_CLK_CTRL[31:30] PLL_PWR_CMD = 0x0 (STATE_OFF)
2 = DSI_CLK_CTRL[31:30] PLL_PWR_CMD = 0x1 (STATE_ON_HSCLK)
3 = DSI_CLK_CTRL[31:30] PLL_PWR_CMD = 0x2 (STATE_ON_ALL)
4 = DSI_CLK_CTRL[31:30] PLL_PWR_CMD = 0x3 (STATE_ON_DIV)



The commands PLLPwrCmdOff, PLLPwrCmdOnAll, PLLPwrCmdOnDIV and PLLPwrCmdOnHSClk controls the state transition of the DSI PLL control module. Software users should set the DSS.DSI_CLK_CTRL[31:30] PLL_PWR_CMD bit field to ask for a state change. The DSS.DSI_CLK_CTRL[29:28] PLL_PWR_STATUS bit field gives a status on the current state of the DSI PLL controller.

NOTE: In a command requests to change to a state which is the current one (acknowledge has been received), the command is ignored (nothing is sent to the DSI PLL Control module).

All the DSI PLL power is controlled by the DSI protocol engine except the LDO power of the PLL and HSDIVIDER that can be controlled by the DSI PLL controller module. Indeed, the HSDIVIDER and PLL SYSRESET signals can be forced by the DSI PLL controller module by setting DSS.DSI_PLL_CONTROL[4] DSI_HSDIV_SYSRESET and DSS.DSI_PLL_CONTROL[3] DSI_PLL_SYSRESET bits, respectively. By setting these bits to 1, the SYSRESET signal is forced active (module is forced to reset state). When these bits are set to 0 (reset value), the SYSRESET signals are controlled by the DSI PLL power FSM.

12.4.3.6.2.1.1 DSI-PLL HS Clock Signals

The DSIStopClk signal is provided to the DSI PLL control module. It indicates when the DSI Protocol engine does not need to use the high-speed transfer mode (HS mode) and PLL HS output (HS_CLK clock) can be stopped. The following conditions must also be met when DISPC_UPDATE_SYNC may be generated by the display controller, as that may also result in the PLL HS output being stopped.

When the interface is disabled (that is, DSS.DSI_CTRL[0] IF_EN bit set to 0), the signal DSIStopClk is asserted.

The assertion of the DSIStopClk depends on the following conditions:

- Clock lane TxRequestHS is deasserted (the DDR clock on the clock lane is not required anymore). The get TxRequestHS deassertion, all of the following conditions are required:
  - The DSS.DSI_CLK_CTRL[13] DDR_CLK_ALWAYS_ON bit must be reset to 0 and no HS data transfer should be on going or already scheduled
  - No VC active in video mode. No VC using the video mode is enabled; if the VC is enabled, the mode is command mode only (that is, DSS.DSI_VCn_CTRL[0] VC_EN bit set to 1 and DSS.DSI_VCn_CTRL[4] MODE bit set to 0)
  - No command mode requiring high-speed transfer (one or more VCs using command mode can be active)
  - Or DSS.DSI_CTRL[0] IF_EN bit reset to 0 (if all previous conditions are not required)

The deassertion of the DSIStopClk depends on one of the following conditions (the DSI interface is enabled by setting the DSS.DSI_CTRL[0] IF_EN bit to 1):

- Clock lane TxRequestHS must be asserted (the DDR clock on the clock lane is required anymore).
- One video mode VC active
- At least one VC in command mode requiring high-speed transfer
- The DSS.DSI_CLK_CTRL[13] DDR_CLK_ALWAYS_ON bit is set to 1 by software users (the DSS.DSI_CTRL[0] IF_EN bit should be reset to 0 for updating the DDR_CLK_ALWAYS_ON bit value)

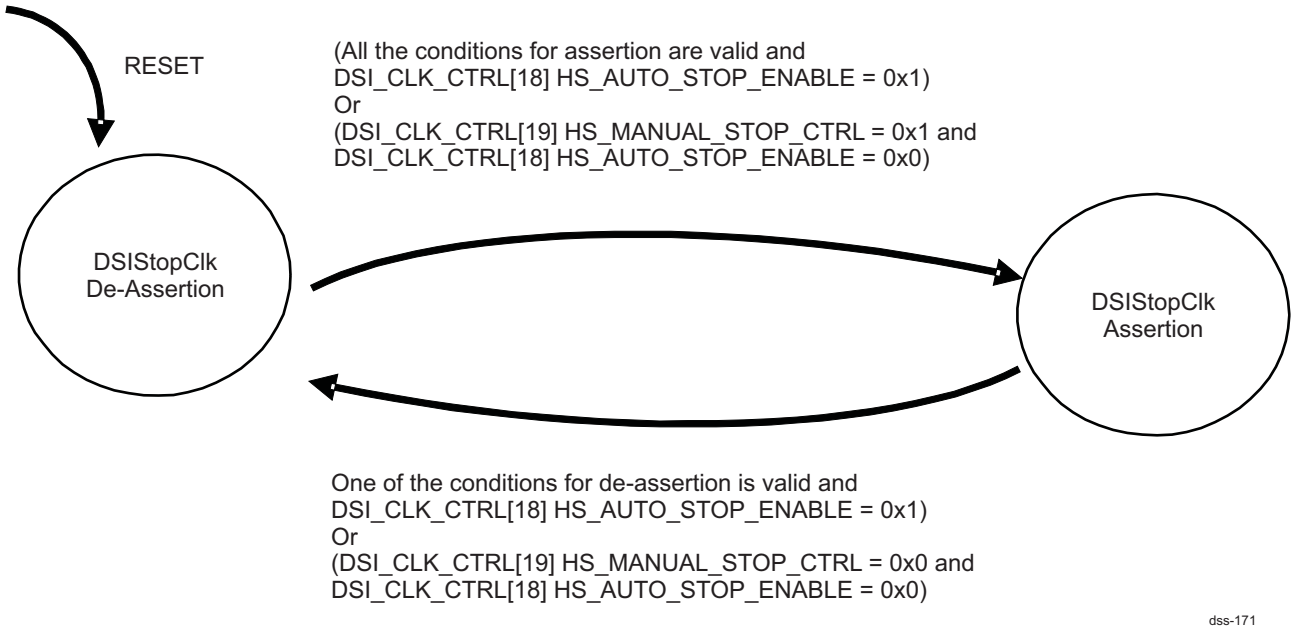
The automatic assertion/deassertion is enabled by using the DSS.DSI_CLK_CTRL[18] HS_AUTO_STOP_ENABLE bit.

The manual mode can be used by setting/resetting the DSS.DSI_CLK_CTRL[19] HS_MANUAL_STOP_CTRL bit to assert/deassert the DSIStopClk signal.

12.4.3.6.2.1.2 DSI-PLL HS Clock FSM

Figure 12-97 shows the DSI PLL HS clock FSM.

Figure 12-97. DSI PLL HS Clock FSM



When DSIStopClk is used there is a latency through other modules (DSI PLL controller and DSI_PHY) before TxByteClkHS is stopped. This latency must be accounted for to prevent any issue when DSIStopClk is deasserted soon after being asserted. This is done using a hardware timer programmed using the DSS.DSI_STOPCLK_TIMING[7:0] DSI_STOPCLK_LATENCY bit field. This timer is programmed in number of periods of the DSI Protocol functional clock (DSI_FCLK). At reset value, the timer is programmed with 0x80 (128) value.

CAUTION

The programmed value in the DSS.DSI_STOPCLK_TIMING[7:0] DSI_STOPCLK_LATENCY bit field must be greater than ((3 x L3_ICLK period) + (5 x CLKIN4DDR period))/(DSI_FCLK period).

12.4.3.7 Timers

NOTE: Among the timers described in this section, only the HS TX, LP RX and turnRequests timers generates interrupts immediately when the timer value is null. For ForceTxStopMode timer, it ends counting instantly and ForceTxStopMode is not asserted.

12.4.3.7.1 Twakeup Timer

The T_{WakeUp} timer is not implemented in the DSI protocol engine. The software must use a general-purpose (GP) timer to handle this. This timer is used for existing ULP status mode for the active lanes (clock and/or data lanes). The sequence to exit ULP state is:

1. Change the state of TxULPSExit for each lane to ACTIVE.
2. Wait for the interrupt indicating that all lanes with TxULPSExit active have acknowledged by asserting ULPSActiveNot. This is done by reading the DSS.DSI_COMPLEXIO_IRQSTATUS ULPSACTIVENOT_ALLi_IRQ bit fields (i = 0, 1).
3. Start the application wake-up timer (GP timer).
4. Wait for the time-out.
5. Change the TxUlpsClk signals to INACTIVE state for the clock lane and/or TxRequestEsc INACTIVE

state for the data lane(s).

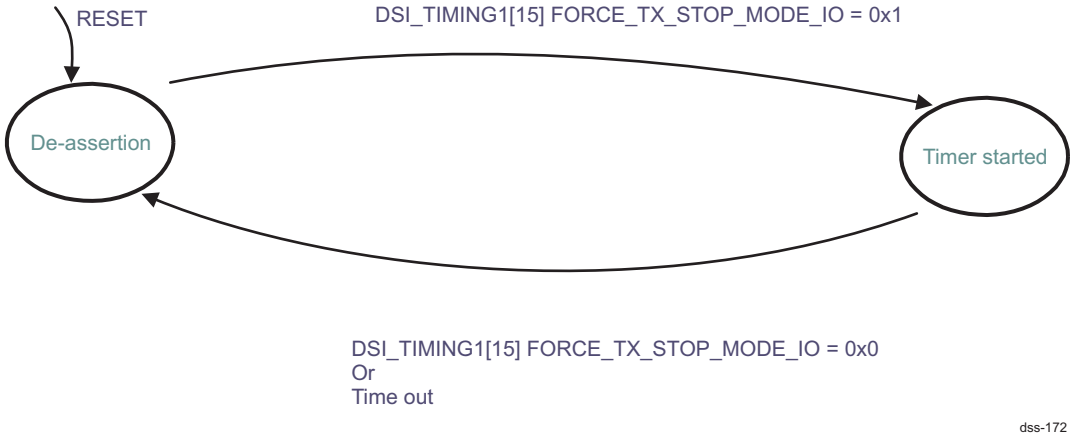
NOTE: The minimum time for the wake-up period is 1 ms.

To enter ULPS mode for clock lane, TxUlpsClk state should be change to active state. To enter ULPS mode for data lane, TxRequestEsc state should be changed to active state (TxUlpsEsc as well if it is not in active state already).

12.4.3.7.2 ForceTxStopMode FSM

The signal ForceTxStopMode is used at initialization time (DSI complex I/O). Figure 12-98 describes the ForceTxStopMode FSM to assert/deassert ForceTxStopMode signal.

Figure 12-98. ForceTxStopMode FSM



The DSI protocol engine asserts ForceTxStopMode by setting the DSS.DSI_TIMING1[15] FORCE_TX_STOP_MODE_IO bit to 1. Asserting the FORCE_TX_STOP_MODE_IO bit allows to initialize the lanes. The lanes are in the Stop State when ForceTxStopMode signal is high.

No data can be sent before the ForceTxStopMode signal is deasserted. The deassertion time is defined by the STOP_STATE_COUNTER_IO, Stop_State_x4_IO, Stop_State_x16_IO field DSI_TIMING1[15:0]. The FORCE_TX_STOP_MODE_IO bit is reset by hardware when the time is reached.

This bit can be reset by software.

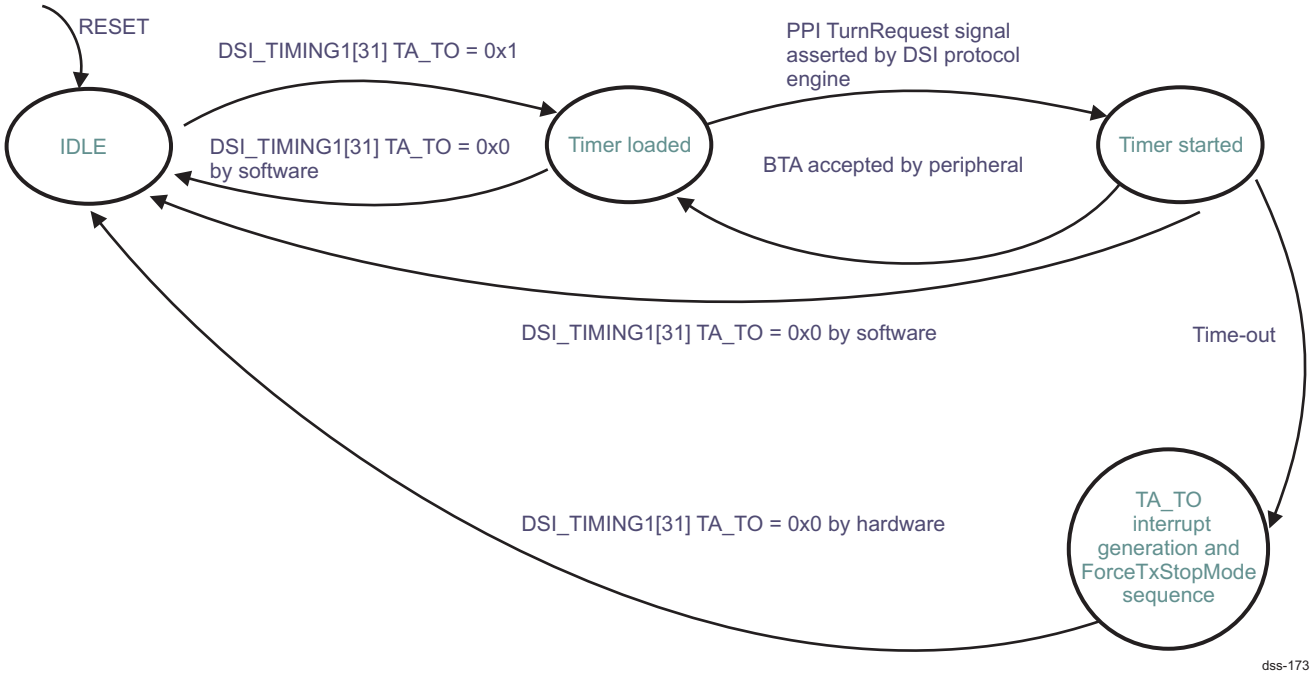
The calculation of the number of DSI_FCLK cycles assertion period is defined by:

Total period in DSI_FCLK cycles = DSI_TIMING1[12:0] STOP_STATE_COUNTER_IO x ((DSI_TIMING1[14] STOP_STATE_X16_IO x 15) + 1) x ((DSI_TIMING1[13] STOP_STATE_X4_IO x 3) + 1)

12.4.3.7.3 TurnRequest FSM

The signal TurnRequest is used to request turnaround. It is only valid for the data lane #1 since the other data lanes can not be used in the reverse direction to receive data from the DSI receiver. Figure 12-99 describes the TurnRequest FSM to assert/deassert TurnRequest signal.

Figure 12-99. TurnRequest FSM



The DSI protocol engine asserts TurnRequest signal during one TxClkEsc cycle when the turn-around is enabled through the DSS.DSI_VCn_CTRL[6] BTA_EN bit (for more information, see Section 12.4.3.8, *Bus Turnaround*. The DSS.DSI_TIMING1[31] TA_TO bit is set/reset by software to respectively enable/disable the timer for turnaround procedure failure. It can be reset by software or automatically by hardware when the time out occurs.

The timer is loaded with the value in number of DSI_FCLK cycles:

DSI_TIMING1[28:16] TA_TO_COUNTER x ((DSI_TIMING1[30] TA_TO_X16 x 15) + 1) x ((DSI_TIMING1[29] TA_TO_X8 x 7) + 1).

When the TA_TO_IRQ interrupt is generated (turn-around timer expired and procedure failed), the hardware automatically asserts ForceTXStopMode in order for the DSI_PHY to drive LP-11 stop state. The ForceTXStopMode timer is used to define the minimum duration of LP-11 state. The Stop State can be longer if there is no activity.

The hardware resets the ForceTXStopMode bit, followed by an internal logic reset except all register values and TX FIFO content, then resets the DSS.DSI_CTRL[0] IF_EN bit. The software should take action to recover by resetting the peripheral, for example, if it is not responding. It should wait for DSS.DSI_TIMING1[15] FORCE_TX_STOP_MODE_IO and DSS.DSI_CTRL[0] IF_EN bits to be reset to 0 before starting the recovery sequence.

12.4.3.7.4 Peripheral Reset Timer

The peripheral reset timer is not implemented in the DSI protocol engine module. Such as the Twakeup timer, a general-purpose timer (GPTimer)should be used in case of reset of the peripheral to determine when the peripheral is ready again for operation.

12.4.3.7.5 HS TX Timer

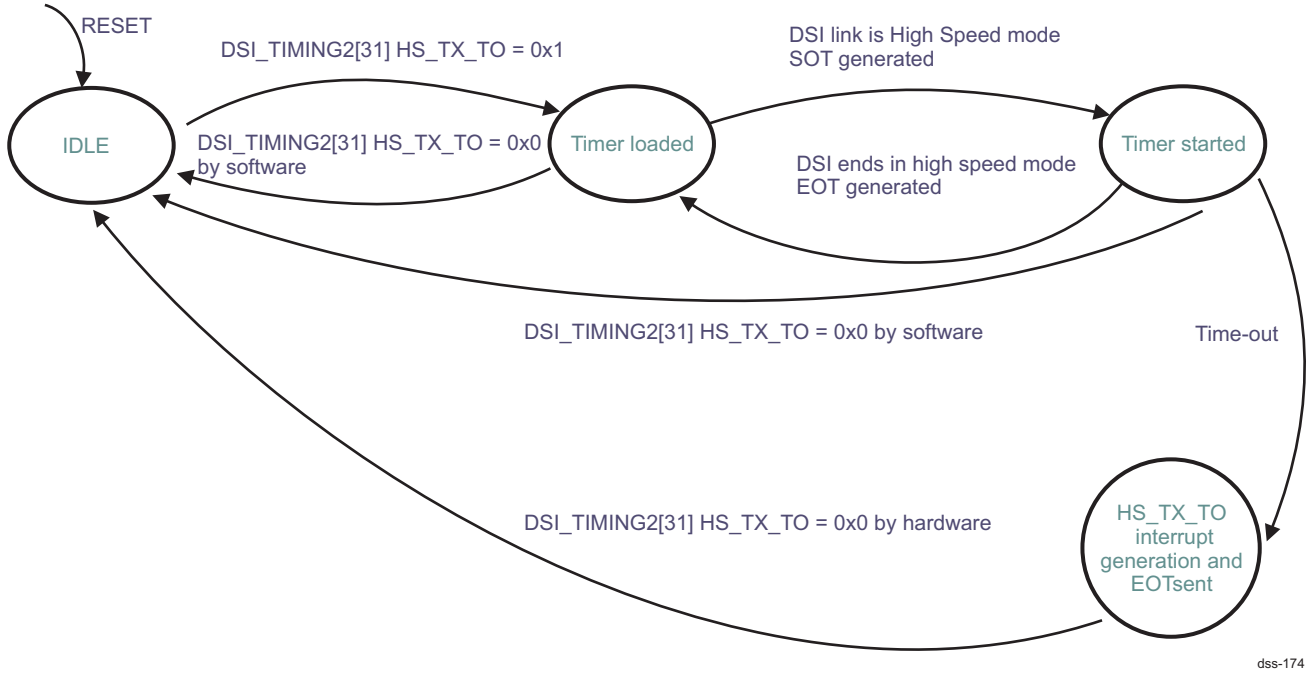
The HS TX timer is used to detect when the host has been in TX mode for too long. When time-out occurs, the EOT is forced. The timer is reloaded when a start of high speed transmission occurs. It is enabled/disabled by software through the DSS.DSI_TIMING2[31] HS_TX_TO bit. The interrupt HS_TX_TO_IRQ is generated when the timer expires. The DSS.DSI_IRQSTATUS[14] HS_TX_TO_IRQ bit is set to 1 when the HS TX time-out occurs.

The maximum time to be supported is 20 ms. It can be used to determine that at least once a frame in video mode, the HS mode is stopped to enter ULPS. Since the refresh rate can be up to 50 frames per second in video mode, the maximum time in HS is 20 ms.

The timer is loaded with the value in number of TxByteClkHS:

DSI TIMING2[28:16] HS_TX_TO_COUNTER x ((DSI TIMING2[30] HS_TX_TO_X16 x 15) +1) x ((DSI TIMING2[29] HS_TX_TO_X8 x 7) +1)

Figure 12-100. High-Speed TX Timer FSM



When the time-out occurs, the hardware should sent EOT request in order for the DSI complex I/O to drive LP-11 stop state. This is followed by the generation of the interrupt. The hardware will perform an internal logic reset including the TX FIFO content, but excluding the register values and then resets the DSS.DSI_CTRL[0] IF_EN bit.

The software should wait for the DSS.DSI_CTRL[0] IF_EN bit to be reset to 0 before taking any recovery action by resetting for example the peripheral if it is not responding.

12.4.3.7.6 LP RX Timer

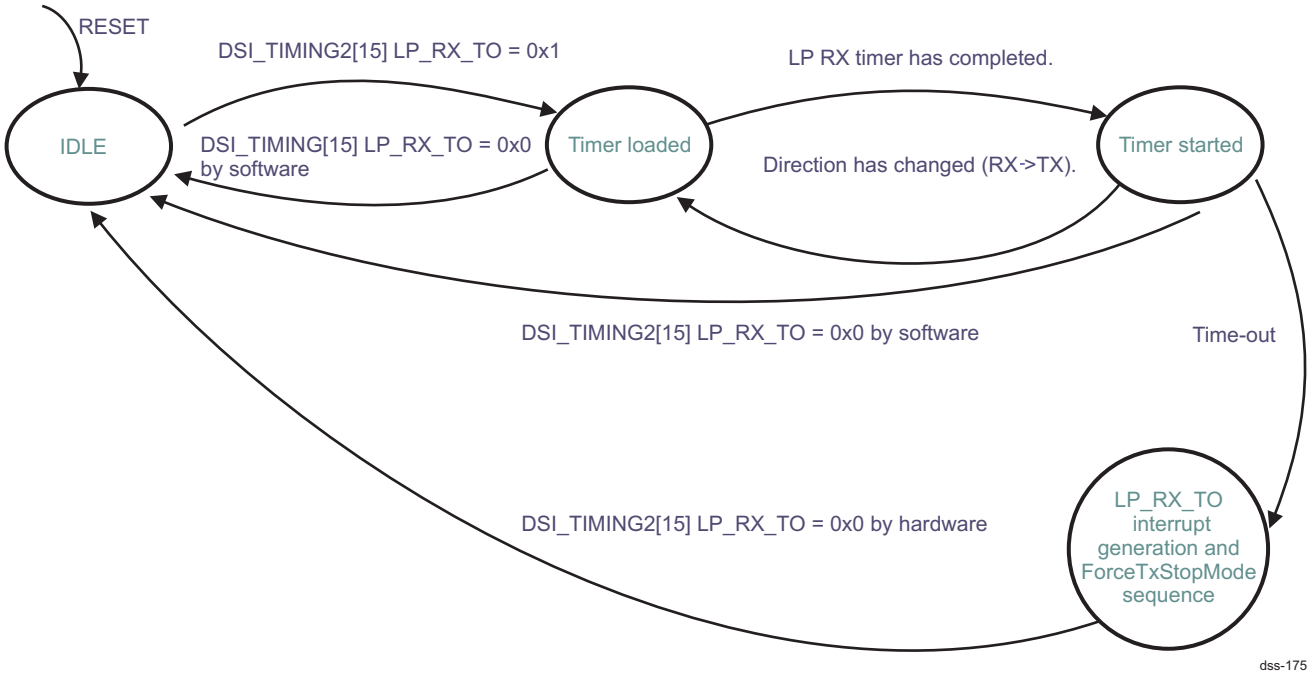
When the host is in Low power Receive mode after a bus turn-around, the LP RX timer is loaded. When the timer expires, the host requests the DSI complex I/O to drive LP-11. The interrupt LP_RX_TO_IRQ is generated when the timer expires. The DSS.DSI_IRQSTATUS[15] LP_RX_TO_IRQ bit is set to 1 when the LP RX time-out occurs.

The DSS.DSI_TIMING2[15] LP_RX_TO bit is set/reset by the software to respectively enable/disable the timer.

The timer is loaded with the value in number of DSI_FCLK cycles:

DSI_TIMING2[12:0] LP_RX_TO_COUNTER x ((DSI_TIMING2[14] LP_RX_TO_X16 x 15) + 1) x ((DSI_TIMING2[13] LP_RX_TO_X4 x 3) + 1)

Figure 12-101. Low-Power RX Timer FSM



When the interrupt is generated, the hardware should automatically reset the DSS.DSI_TIMING2[15] LP_RX_TO bit and then assert ForceTXStopMode in order for the DSI complex I/O to drive LP-11 stop state. The ForceTXStopMode timer is used to define the minimum duration of LP-11 state. The Stop State can be longer if there is no activity.

The hardware resets the ForceTXStopMode bit, followed by an internal logic reset except all register values and TX FIFO content, then resets the DSS.DSI_CTRL[0] IF_EN bit. The software should take action to recover by resetting the peripheral, for example, if it is not responding. It should wait for the DSS.DSI_TIMING1[15] FORCE_TX_STOP_MODE_IO and DSS.DSI_CTRL[0] IF_EN bits to be reset before starting the recovery sequence. The TX FIFO is not flushed (the FIFO is flushed only when DSS.DSI_VCn_CTRL[0] VC_EN is set to 1).

12.4.3.8 Bus Turnaround

The bus turn-around (BTA) is not automatically sent by default after each packet sent to the display(s). It is programmable independently for each VC ID. The VC can be enabled when DSS.DSI_VCn_CTRL[6] BTA_EN bit is set to 1 by software. The software should ensure that, when the BTA is sent to the peripheral, there is enough time allocated for the response and the BTA from the peripheral to host. For more information about possible DSI PHY timing adjustments during the turn-around procedure, see Section 12.5.6.4.3, *Turn-Around Request in Transmit Mode*, and Section 12.5.6.4.4, *Turn-Around Request in Receive Mode*. When setting the DSS.DSI_VCn_CTRL[6] BTA_EN bit to 1, one BTA is sent manually to the peripheral. This manual mode can be used for packets in command or video mode.

Acknowledgment from the peripheral for successful BTA is indicated by asserting the BTA_IRQ interrupt, if it is enabled in the DSS.DSI_VCn_IRQENABLE[5] BTA_IRQ_EN bit. To monitor the BTA interrupt, the user must read the DSS.DSI_VCn_IRQSTATUS[5] BTA_IRQ status bit.

CAUTION

The BTA should not be sent when the RX FIFO is not empty. Users should take care of emptying the RX FIFO before sending BTA to the peripheral. It is to ensure that when receiving new data from peripheral, all the allocated spaces for all the VCs are empty.

In automatic mode, the BTA is sent automatically at the end of short or long packets when respectively the DSS.DSI_VCn_CTRL[2] BTA_SHORT_EN or the DSS.DSI_VCn_CTRL[3] BTA_LONG_EN bits are set to 1.

NOTE: If the DSS.DSI_VCn_CTRL[2] BTA_SHORT_EN bit is enabled, users can still set the DSS.DSI_VCn_CTRL[6] BTA_EN bit. Only one BTA is sent to the peripheral and the DSS.DSI_VCn_CTRL[6] BTA_EN bit is reset by hardware.

If the DSS.DSI_VCn_CTRL[3] BTA_LONG_EN bit is enabled, users can still set the DSS.DSI_VCn_CTRL[6] BTA_EN bit. Only one BTA is sent to the peripheral and the DSS.DSI_VCn_CTRL[6] BTA_EN bit is reset by hardware.

If the DSS.DSI_VCn_CTRL[2] BTA_SHORT_EN and DSS.DSI_VCn_CTRL[3] BTA_LONG_EN bits are both enabled, users can still set the DSS.DSI_VCn_CTRL[6] BTA_EN bit to send a BTA. Only one BTA is sent and the DSS.DSI_VCn_CTRL[6] BTA_EN bit is reset by hardware.

As explained previously, two modes can be used for each VC ID:

- Automatic: After each packet, a bus turn-around is sent. To determine the size of the long packet, the protocol engine on the host side should read the word count defined in the header (in DSS.DSI_VCn_LONG_PACKET_HEADER register) and use it to determine the last data to be sent on the DSI link. For short packets, the size is always 4 bytes. Then the bus turn-around is sent to the peripheral. The word count is also used to determine how many bytes should be transferred from the 32-bit writes access to the payload register (DSS.DSI_VCn_LONG_PACKET_PAYLOAD register).
- Manual: In case of data transfer using the L4 interconnect port, while all data have been provided to the DSI protocol engine, users can select bus turn-around for the last packet provided to the L4 interconnect port only by setting the bus turn-around enable bit (DSS.DSI_VCn_CTRL[6] BTA_EN) or for last packets and following ones by setting the automatic mode; in case of data transfer using the video port, the bus turnaround enable bit (DSS.DSI_VCn_CTRL[6] BTA_EN) can be selected at any time during the transfer of the packet. In case of video mode packets (data and synchronization events) users can not determine when the BTA is sent relatively the video mode packets, so it is highly recommended to use manual BTA mode only for packets generated in command mode but it is possible to use BTA when for a VC in video mode. In case of data provided on the video port, an interrupt for end of packet transfer (PACKET_SENT_IRQ) is provided to indicate users when the packet has been completely sent by the DSI complex I/O. The PACKET_SENT_IRQ can be monitored in DSI_VCn_IRQSTATUS[2] PACKET_SENT_IRQ status bit. Users can request BTA even if the space allocated in the TX FIFO for the corresponding VC is empty. It can be sent later on even if there was no packet sent before BTA request. The DSS.DSI_VCn_CTRL[6] BTA_EN bit should be reset by hardware if the BTA request has been sent even if the automatic mode for this specific type of packets is enabled.

The bus turnaround is supported for video mode packets and for command mode packets. It is not possible to send BTA during the blanking periods of the video mode when HS blanking packets should be sent, that is, when one of the following bits is set to 1:

- DSS.DSI_CTRL[20] BLANKING_MODE
- DSS.DSI_CTRL[21] HFP_BLANKING_MODE
- DSS.DSI_CTRL[22] HBP_BLANKING_MODE
- DSS.DSI_CTRL[23] HSA_BLANKING_MODE

Therefore, in video mode, the BTA request is delayed until there is a blanking period without HS blanking packets.

TA Timer

When TurnRequest signal is asserted (always only for data lane #1), the TA_TO timer is started. If the direction signal is no changed according to the turn-around request, the TA_TO interrupt is generated. When the Direction signal is in output mode, any data on the input data bus should be ignored since the DSI is in transmission mode (data and triggers should be ignored). Refer toSection 12.4.3.7.3 for more details on the TA_TO timer.

12.4.3.9 PHY Triggers

The DSI protocol engine uses three triggers, which are supported only for data lane 1:

- Reset from host to display
- Tearing effect (TE) from display to host
- Acknowledge from display to host

CAUTION

Each trigger is associated with a dedicated user-configurable receive or transmit pattern, loaded in the DSI_PHY_CFG3 or DSI_PHY_CFG4 bit fields. The default (reset) values of the bit fields are aligned with the MIPI DSI_PHY specification v0.9. If the user must change any of these values, the following must be considered:

- If any of the bit fields is written with a non-default value, the other bit fields in the same register must also be configured with different values. This is to ensure that two different trigger bit fields are not programmed with the same pattern.
- If two or more bit fields are written with equal values, this may lead to unpredictable behavior of the DSI PHY module.

12.4.3.9.1 Reset

The DSI protocol engine can use one of the triggers of the DSI_PHY to send a reset to the display. The reset trigger pattern is configurable through the DSI_PHY_CFG3[31:24] TXTRIGGERESC3 bit field. To send the reset pattern to the peripheral, the DSS.DSI_CTRL[5] TRIGGER_RESET bit must be set to 1. When the software requires the trigger reset pattern to be sent, the DSI protocol engine resets its own logic but not the registers. The software can select between two reset modes:

- Immediate reset: All pending requests in TX FIFO not already taken into account for transfer scheduling, the RX FIFO requests, and the data from video port are ignored. Only the current transfer on DSI link and already scheduled ones are transmitted. All the other transfers are discarded.
- Synchronized reset: The mode is only valid if there is VC using the video mode and if it is active. The principle is to wait for the current video frame to be transferred on the link. Any data on VP after the current frame are ignored.

To select the reset mode, software users must program the DSS.DSI_CTRL[14] TRIGGER_RESET_MODE.

CAUTION

For both reset modes, the hardware should flush the FIFOs, synchronization buffers, and line buffers before resetting the DSS.DSI_CTRL[0] IF_EN bit.

12.4.3.9.2 Tearing Effect

The TE on the display is avoided by having synchronization information from the display. It is used only in command mode. In case of video mode, it is not functional. Users are responsible for selecting the command mode for the VC using the TE feature.

The software must set and send the appropriate sequence to receive the TE trigger pattern from the peripheral. The value of the expected TE trigger pattern can be configured through the DSI_PHY_CFG4[23:16] RXTRIGGERESC2 bit field. When the TE trigger pattern is received, the DSI protocol engine generates the TE_TRIGGER_IRQ interrupt with TE event if the interrupt is enabled. To enable the interrupt, set to 1 the DSS.DSI_IRQENABLE[16] TE_TRIGGER_IRQ_EN bit. The DSS.DSI_IRQSTATUS[16] TE_TRIGGER_IRQ status bit indicates if the interrupt event has been generated.

One or multiple VCs can be synchronized using the same TE trigger. The DSS.DSI_VCn_TE[30] TE_EN bit should be set to indicate that the hardware should use the following TE trigger to start the transfer of the data from the related VC. This bit is reset when all the data have been sent to the peripheral. The DSS.DSI_VCn_TE[31] TE_START bit should be used when the automatic mode enabled by setting the DSS.DSI_VCn_TE[30] TE_EN bit is not used. It allows users to start the transfer manually based on application events or based on the TE trigger interrupt (TE_TRIGGER_IRQ).

The number of bytes to be transferred is defined by using the DSS.DSI_VCn_TE[15:0] TE_SIZE bit field. The TE_SIZE bit field is decremented for each payload byte (it does not include Check-sum) sent on the DSI link. The register content should not be modified by software during a transfer. The DSS.DSI_VCn_TE[15:0] TE_SIZE bit field should be set first to indicate that the following accesses to DSS.DSI_VCn_LONG_PACKET_HEADER register should be used for TE transfer.

The data can be provided from two sources (selection by setting the DSS.DSI_VCn_CTRL[1] SOURCE bit):

- L4 interconnect port using DMA request: The DMA request DSI_DMA_REQi (where i =0 to 3) to should be asserted only when TE trigger is received or when the DSS.DSI_VCn_TE[31] TE_START bit is set by user and should not be asserted anymore when all the bytes defined in DSS.DSI_VCn_TE[15:0] TE_SIZE bit field have been sent on the DSI link. The VC is associated with a DMA request (from DSI_DMA_REQ0 to DSI_DMA_REQ3) by programming the number in the DSS.DSI_VCn_CTRL[23:21] DMA_TX_REQ_NB bit field. The DSS.DSI_VCn_LONG_PACKET_PAYLOAD register is used to provide the number of bytes defined by the DSS.DSI_VCn_TE[15:0] TE_SIZE bit field (the check-sum value is not provided in the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register). The size of the header is not taken into account in the number of bytes to transfer. The DSS.DSI_VCn_SHORT_PACKET_HEADER register is not used.
- Video port: The DMA request is not asserted. The data are captured in the line buffer using the STALL mechanism. In case there is no line buffer instantiated (that is, DSS.DSI_CTRL[13:12] LINE_BUFFER bit field set to 0), it is not possible to use the video port to provide data. The line buffer should be filled up according to the word count defined in the DSS.DSI_VCn_LONG_PACKET_HEADER register header. The value should be written before the TE trigger event is received or before the DSS.DSI_VCn_TE[31] TE_START bit is set to 1 by software. In case the total number of bytes defined by the DSS.DSI_VCn_TE[15:0] TE_SIZE bit field is not a multiple of the word count defined in the DSS.DSI_VCn_LONG_PACKET_HEADER register, all the packets have the same size defined by the WC of the header except the last transfer. The size of the last transfer is defined by the remaining bytes to send. Since the DSS.DSI_VCn_TE[15:0] TE_SIZE bit field is modified after each packet transfer, the size of the last packet is equal to the value of DSS.DSI_VCn_TE[15:0] TE_SIZE bit field just before the last transfer (the header and the payload check-sum sizes are not included in DSS.DSI_VCn_TE[15:0] TE_SIZE bit field).

When the transfer is completed, the value of the DSS.DSI_VCn_TE[15:0] TE_SIZE bit field is equal to 0. The software must ensure that the pending data in the TX FIFO for the corresponding VC using TE are related to TE transfer. Any data in the TX FIFO that should be sent before reception of TE trigger should be sent before TE. This is done by not enabling TE trigger until all data for the corresponding VC have been sent to the peripheral. The software can check that the space allocated for the VC in the TX FIFO is empty by reading the DSS.DSI_VCn_CTRL[5] TX_FIFO_NOT_EMPTY status bit.

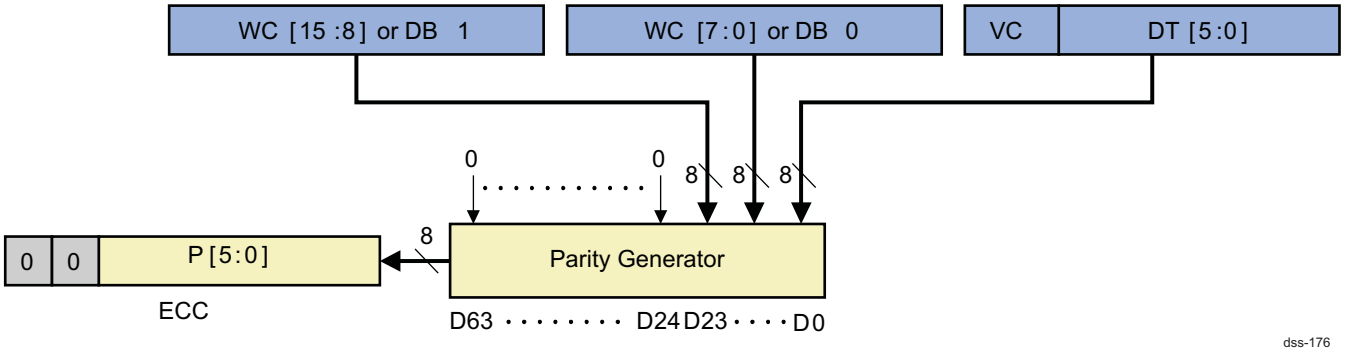
12.4.3.9.3 Acknowledge

The corresponding Acknowledge interrupt (ACK_TRIGGER_IRQ) is generated upon reception of the acknowledge trigger. The value of the expected acknowledge trigger pattern can be configured through the DSI_PHY_CFG4[15:8] RXTRIGGERESC1 bit field. To enable the acknowledge interrupt, set the DSS.DSI_IRQENABLE[17] ACK_TRIGGER_IRQ_EN bit to 1. When the interrupt is generated, the DSS.DSI_IRQSTATUS[17] ACK_TRIGGER_IRQ status bit is set to 1.

12.4.3.10 ECC Generation

Note that the DSI protocol uses a four-byte packet header. Since ECC generation requires a fixed word length of 64-bits, the packet headers should be padded with additional bits to form a full eight-byte value for ECC generation and checking. The packet header less the ECC byte should occupy bits D[23:0] and the pad bits should occupy bits D[63:24]. All padding bits should be zero for the purpose of generating the ECC byte. ECC can be generated using a parallel approach as illustrated in Figure 12-102.

Figure 12-102. 64-Bit ECC Generation on TX Side



The ECC generation/check can be enabled and disabled by software. It is defined by a common bit for all the VCs:

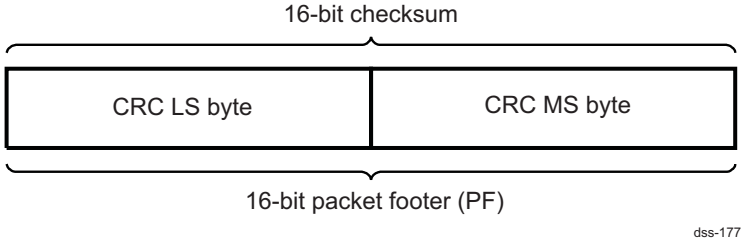
- The DSS.DSI_CTRL[2] ECC_RX_EN bit enables/disables the ECC generation in the receive direction.
- The DSS.DSI_VCn_CTRL[8] ECC_TX_EN bit enables/disables the ECC generation in the transmit direction

12.4.3.11 Checksum Generation for Long Packet Payloads

Long packets are comprised of a packet header protected by an ECC byte and a payload of 0 to 2^{16} - 1 bytes. To detect the errors during the transmission of long packets, a checksum is calculated over the payload portion of the data packet. Note that, for the special case of a zero-length payload, the 2-byte checksum is set to 0xFFFF. The checksum can only indicate the presence of one or more errors in the payload. Unlike ECC, the checksum does not enable error correction. For this reason, checksum calculation is not useful for some unidirectional DSI implementations since the peripheral has no way for reporting errors to the host processor. Checksum generation and transmission is mandatory for host processors sending long packets to peripherals. It is optional for peripherals transmitting long packets to the host processor. However, the format of long packets is fixed; the peripherals that do not support checksum generation should transmit two bytes having value 0x0000 in place of the checksum bytes when sending long packets to the host processor. The host processor should disable checksum checking for received long packets from peripherals that do not support checksum generation.

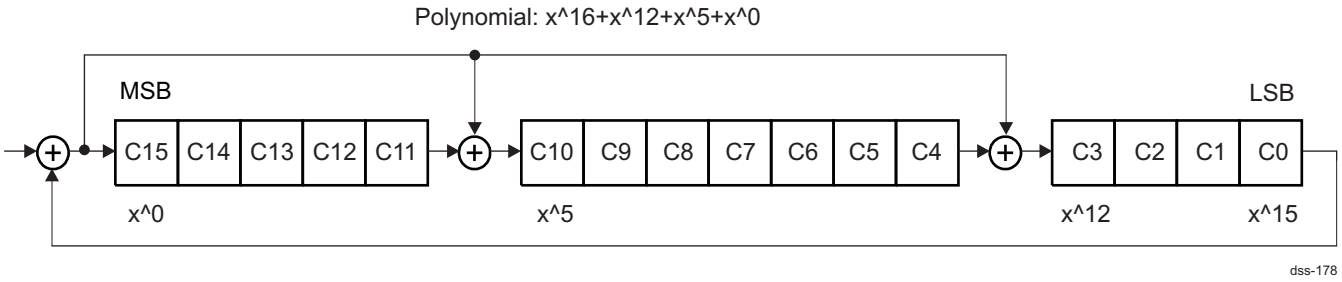
The checksum should be realized as a 16-bit CRC with a generator polynomial of x^16+x^12+x^5+x^0. The LS byte is sent first, followed by the MS byte. Note that within the byte, the LS bit is sent first.

Figure 12-103. Checksum Transmission



The CRC implementation is presented in Figure 12-104. The CRC shift register is initialized to 0xFFFF before packet data enters. Packet data not including the packet header then enters as a bitwise data stream from the left, LS bit first. Each bit is fed through the CRC shift register before it is passed to the output for transmission to the peripheral. After all bytes in the packet payload have passed through the CRC shift register, the shift register contains the checksum. C15 contains the checksums MSB and C0 the LSB of the 16-bit checksum. The checksum is then appended to the data stream and sent to the receiver.

Figure 12-104. 16 Bit CRC Generation Using a Shift Register



The check-sum generation/check can be enabled and disabled by software. It is defined by a common bit for all the VCs:

- The DSS.DSI_CTRL[1] CS_RX_EN bit enables/disables the check-sum generation in the receive direction.
- The DSS.DSI_VCn_CTRL[7] CS_TX_EN bit enables/disables the check-sum generation in the transmit direction

12.4.3.12 End of Transfer Packet

To allow the DSI protocol (rather than the DSI_PHY) at the display to detect the HS End Of Transfer (EOT), an EOT packet type is added. It is a fixed short packet (4 bytes) that is added at every HS-to-LP transition. This function is enabled by the DSI_CTRL[19] EOT_ENABLE bit.

The EOT packet has a fixed format:

- Data Type = DI [5:0] = 0b001000
- Virtual Channel = DI [7:6] = 0b00
- Payload Data [15:0] = 0x0F0F
- ECC [7:0] = 0x01

When more than one data lane is used, the bytes in the EOT packet are distributed across multiple lanes. EOT packet generation is supported only for the end of HS transmissions. No EOT packet is added at the end of LP transmissions. For LP reception, any EOT packet received is simply passed through the same as any other packet, but no internal decode or use is made of the EOT information.

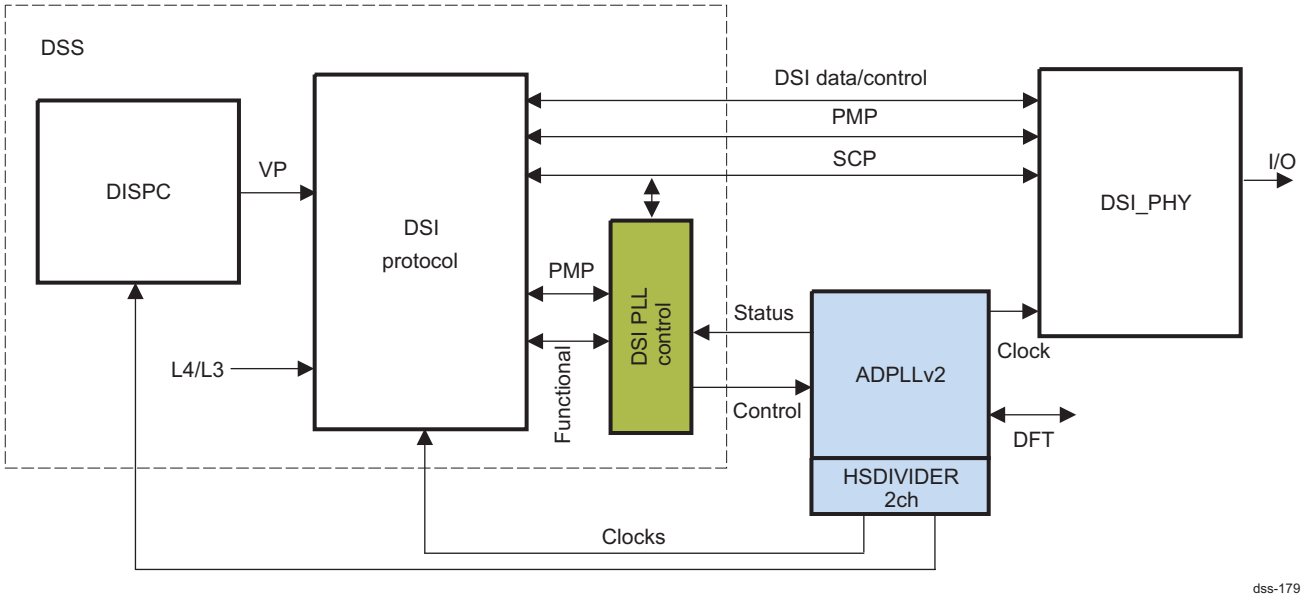
12.4.4 DSI PLL Controller Functionalities

12.4.4.1 DSI PLL Controller Overview

The DSI PLL controller module forms part of the display sub-system. Nevertheless, it uses the SCP (Serial Configuration Port) and PMP (Power Management Port) ports as the primary interfaces to the DSI protocol engine. The SCP interface is used to set the configuration of the DPLL and HSDIVIDER modules, primarily the various counter values. The PMP port is used to control the power state of the DPLL and HSDIVIDER modules. Figure 12-105 provides an overview of the DSI PLL controller module inside the display subsystem.

The DSI PLL is also used to generate the 74.25-MHz frequency used for HDTV applications.

Figure 12-105. DSI PLL Controller Overview



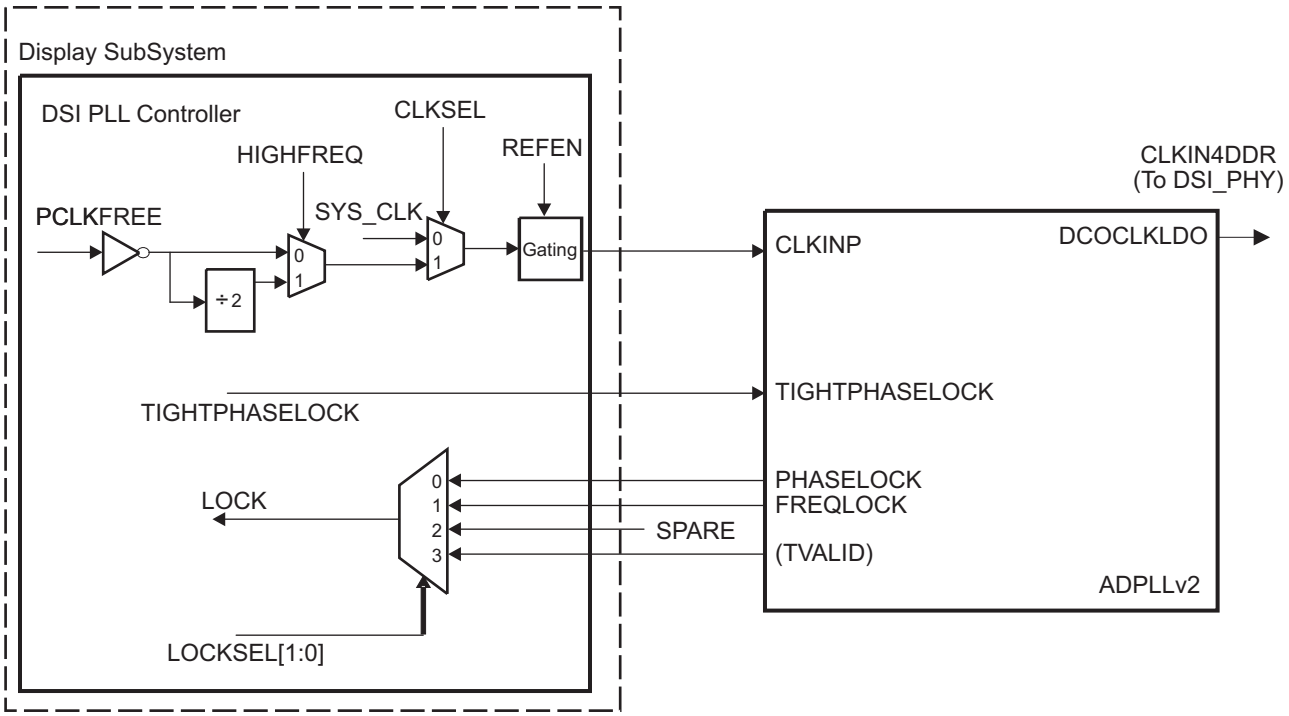
NOTE: The DSI PLL controller module does not have an interface to L4 interconnect. The programmable features are managed by registers mapped into the DSI protocol engine.

12.4.4.2 DSI PLL Controller Architecture

The DSI PLL is an ADPLLv2 module. The pixel clock (PCLK) frequency range is 2 to 65 MHz. This may be divided by 2. This is performed by setting the DSS.DSI_PLL_CONFIGURATION2[12] DSI_PLL_HIGHFREQ bit to 1.

Figure 12-106 shows the internal DSI PLL reference diagram.

Figure 12-106. DSI PLL Reference Diagram



The DSI PLL clock output corresponds to the CLKIN4DDR clock of the DSI complex I/O module.

The DSI PLL reference clock is the DSI_PLL_REFCLK clock. Depending on the setting in DSS.DSI_PLL_CONFIGURATION2[11] DSI_PLL_CLKSEL bit, the reference clock can be either the DSS2_ALWON_FCLK provided by the PRCM or the PCLKFREE provided by the DISPC module.

12.4.4.3 DSI PLL Operations

The DSI PLL configuration signals operate according to Table 12-35. Table 12-35 indicates the operation when the PLL is not locked.

Table 12-35. DSI PLL Operation Modes When Not Locked

DSI PLL Operation Mode	Stop mode Low power[1]	Stop mode Fast Relock[1]	Idle bypass
Mode Description	**Output clocks stopped Lowest power standby**	**Output clocks stopped Fastest start-up time**	**Selects when PLL and HSDIVIDER bypass clocks are used**
DSS.DSI_PLL_CONFIGURATION2[0] DSI_PLL_IDLE	0	0	1
DSS.DSI_PLL_CONFIGURATION2[6] DSI_PLL_LOWCURRSTBY	1	0	1
DSS.DSI_PLL_CONFIGURATION1[0] DSI_PLL_STOPMODE	1	1	X

(1) Recommended

When locked, the PLL output frequency is: [(2xREGM)/(REGN + 1)] x [CLKin(MHz)/(HIGHFREQ + 1)] where:

- M multiplier is programmed in DSS.DSI_PLL_CONFIGURATION1[18:8] DSI_PLL_REGM bit field.
- N divider is programmed in DSS.DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN bit field.
- HIGHFREQ divider by 2 is enabled by setting the DSS.DSI_PLL_CONFIGURATION2[12]

DSI_PLL_HIGHFREQ bit to 1.

12.4.4.4 DSI PLL Controller Shadowing Mechanism

The configuration registers are accessed through the DSI protocol engine register space using SCP port. This includes all the configuration signals and the returning status signals.

CAUTION

All writes must be 32-bit operations as the SCP always transfers 32 bits. Any 16-bit or 8-bit operations may lead to unpredictable errors.

A shadow mechanism is implemented for appropriate register values so that configurations may optionally be updated in synchronism with the display controller (DISPC) and DSI protocol engine. The front porch time from the DISPC indicates the time when making the update of the value. All the required updated values must have been written before this signal is asserted. Refer to Section 12.4.3.5.1 for more details.

12.4.4.5 Error Handling

The PLL lock and recalibration signals may be monitored to detect loss of lock or requirement to recalibrate (due to large temperature change since the last lock request):

- The DSS.DSI_PLL_STATUS[1] DSI_PLL_LOCK status bit gives the DSI PLL lock state.
- The DSS.DSI_PLL_STATUS[2] DSI_PLL_RECAL status bit informs if the PLL must be uncalibrated

These signals can also generate interrupts at DSI protocol engine level:

- The PLL_LOCK_IRQ interrupt indicates that the DSI PLL control module has sent a lock request to the DSI PLL. To monitor this event, read the DSS.DSI_IRQSTATUS[7] PLL_LOCK_IRQ bit. Set this bit to 1 to clear the status bit.
- The PLL_UNLOCK_IRQ interrupt indicates that the DSI PLL control module has sent an unlock request to the DSI PLL. To monitor this event, read the DSS.DSI_IRQSTATUS[8] PLL_UNLOCK_IRQ bit. Set this bit to 1 to clear the status bit.
- The PLL_RECAL_IRQ interrupt indicates that the DSI PLL control module has sent a recalibration request to the DSI PLL. To monitor this event, read the DSS.DSI_IRQSTATUS[9] PLL_RECAL_IRQ bit. Set this bit to 1 to clear the status bit.

12.4.5 DSI Complex I/O Functionalities

12.4.5.1 DSI Complex I/O Overview

DSI_PHY is a complex I/O with 3 unidirectional (HS) Lane Modules. This includes 2 data lane modules and 1 clock lane module. Each lane module has 2 data pads (DX, DY). These data pads are connected with a complementary lane module on the DSI receiver device using point to point interconnect.

Lane modules support high-speed burst mode. Forward direction and reverse direction escape modes are also supported. Escape modes maybe used for Low Power Data Transmission, among other things.

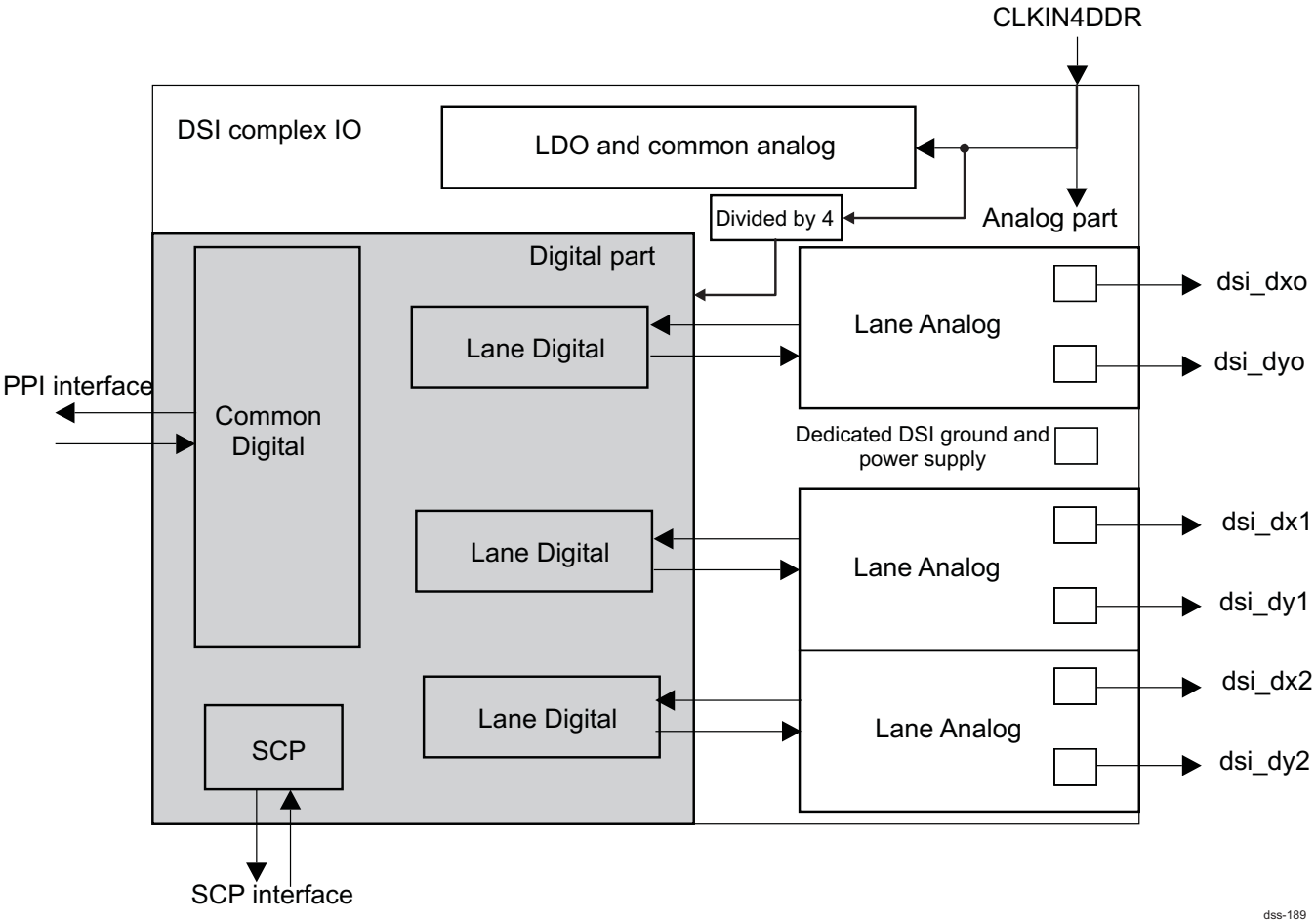
The maximum data rate supported in high-speed Mode is 800Mbps per data lane. The lane module function and position is configurable, that is, any lane module can be chosen as clock lane module, and DX/DY data pad for each lane module can be configured as either DP or DN pins defined by DSI_PHY spec.

DSI_PHY interacts with the higher layers of the DSI link through the PHY-Protocol Interface (PPI). DSI_PHY does not include a PLL; a high frequency clock input is expected in HS mode (CLKIN4DDR). DSI_PHY supports also Serial Configuration Protocol (SCP) to set various configuration and control registers. The DSI_PHY supports GPIO operation on each of the six data pins.

12.4.5.2 DSI Complex I/O Architecture

Figure 12-107 shows the top-level block diagram of DSI_PHY. It has 3 lane module functions, and some common analog and digital functions. Each lane module has an analog and a digital part. The digital lane circuitry implements the protocol interface of MIPI DSI_PHY. It interacts with the DSI protocol engine.

Figure 12-107. DSI Complex I/O Architecture



The lane module analog circuit converts the digital levels to the LP and HS level signals for the link. It also converts the analog state of the link into digital levels.

For high-speed signals, the lane module analog also does the last stage untiming of HS data to obtain optimum quadrature with the HS clock.

DSI_PHY has an internal LDO to generate 1.2V supply. This LDO requires an off-chip decoupling capacitor and hence a dedicated pin. DSI_PHY expects a high frequency clock input in HS mode.

The clock input is required to have four times the frequency of the DDR clock, that is, twice the HS data rate. GPIO functions are supported on each dsi_dxi and dsi_dyi.

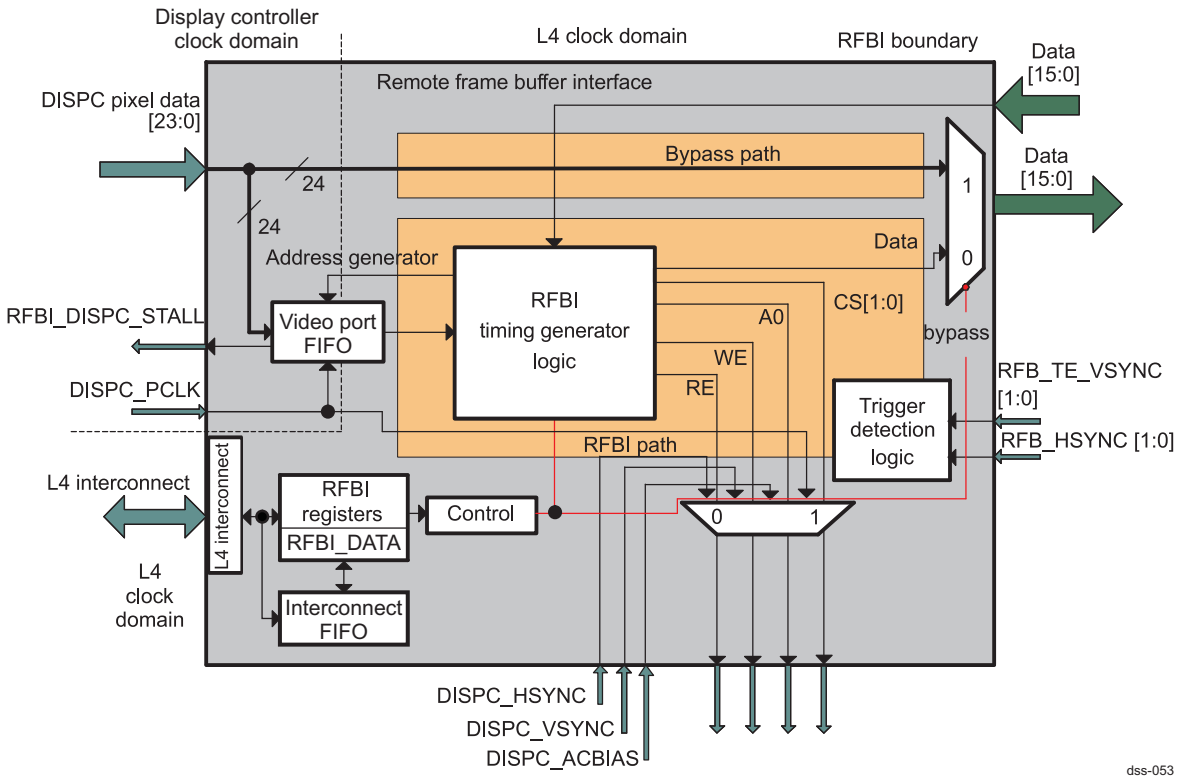
12.4.6 RFBI Functionalities

The RFBI module can capture the output pixel from the display controller and send the data to the RFB in the LCD panel. The application configures the RFBI module, sends commands, reads data, and configures the display controller to send data fetched from the system memory by the display controller DMA engine. The commands/data are sent using an 8-, 9-, 12-, or 16-bit parallel interface.

The display controller is configured to send the data in 12-, 16-, 18-, or 24-BPP format. In the video port FIFO, the encoded pixel values are in an LSB alignment independently of the endianness in system memory.

Figure 12-108 shows an overview of the RFBI architecture.

Figure 12-108. RFBI Architecture Overview



12.4.6.1 RFBI FIFO

The input video port FIFO receives data from the display controller at the pixel clock. The data in the video port FIFO are read by the RFBI and are sent to the LCD panel. The video port FIFO is 24 bits wide and each pixel in 12-, 16-, 18-, and 24-BPP format is stored in the video port FIFO using one 24-bit value aligned on the 24-bit LSB. Section 12.4.6.4, *Output Parallel Modes*, shows an example of an output configuration based on the interface width (16 bits) and the pixel format output (24 bits).

12.4.6.2 RFBI Interconnect FIFO

The interconnect FIFO receives the data from RFBI_DATA write requests to the L4 interconnect slave port. The data in the interconnect FIFO are read by the RFBI and sent to the LCD panel. The width of the interconnect FIFO is 32 bits. The size of the interconnect FIFO is 24 words of 32 bits (that is, 24 words of RFBI_DATA).

12.4.6.3 Input Pixel Formats

The supported pixel formats in the RFBI module are: RGB24-888, RGB18-666, RGB16-565, and RGB12-444 as output from the display controller and from the L4 (for writing parameters). In both cases, the pixels are formatted in accordance with the configuration of the output interfaces (multiple cycles).

12.4.6.4 Output Parallel Modes

The RFBI output modes are 8-, 9-, 12-, and 16-bit interfaces. Any mode can be selected regardless of the pixel format. Set the right configuration in the cycle registers to define a valid configuration for each output cycle.

The following example is an output configuration based on the 16-bit interface width and the 24-bit pixel format (i = 0 or 1) (see also Table 12-36):

- The DSS.RFBI_CONFIGi[10:9] CYCLEFORMAT bit field is set to 0x3 (three cycles for two pixels).

- The DSS.RFBI_DATA_CYCLE1_i register is set to 0x00000010 (16 bits from pixel 1 for the first cycle).
- The DSS.RFBI_DATA_CYCLE2_i register is set to 0x00080808 (8 bits from pixel 1 and pixel 2 and alignment of 8 bits from pixel 2 for the second cycle).
- The DSS.RFBI_DATA_CYCLE3_i register is set to 0x00100000 (16 bits from pixel 2 for the third cycle).

Table 12-36. 16-Bit Interface Configuration/24-Bit Mode

	24-Bit Mode		
	1st Cycle	2nd Cycle	3rd Cycle
Data[15]	R0[7]	B0[7]	G1[7]
Data[14]	R0[6]	B0[6]	G1[6]
Data[13]	R0[5]	B0[5]	G1[5]
Data[12]	R0[4]	B0[4]	G1[4]
Data[11]	R0[3]	B0[3]	G1[3]
Data[10]	R0[2]	B0[2]	G1[2]
Data[9]	R0[1]	B0[1]	G1[1]
Data[8]	R0[0]	B0[0]	G1[0]
Data[7]	G0[7]	R1[7]	B1[7]
Data[6]	G0[6]	R1[6]	B1[6]
Data[5]	G0[5]	R1[5]	B1[5]
Data[4]	G0[4]	R1[4]	B1[4]
Data[3]	G0[3]	R1[3]	B1[3]
Data[2]	G0[2]	R1[2]	B1[2]
Data[1]	G0[1]	R1[1]	B1[1]
Data[0]	G0[0]	R1[0]	B1[0]

12.4.6.5 Unmodified Bits

In a cycle, if every bit in the interface does not have a pixel value, the status of the unused bits can be programmed to be 0, 1, or the previous value (I/O power consumption optimization).

12.4.6.6 Bypass Mode

In bypass mode, the RFBI path is bypassed and the display controller data and signals are sent directly to the output interface of the RFBI.

12.4.6.7 Send Commands

The commands are written through the L4 interconnect and into the DSS.RFBI_CMD register. After a command is sent, another one can be accepted by the module and set. If the processing of a command is not complete, the MPU access to change the command stalls.

12.4.6.8 Read/Write

Depending on the status of A0, WE, and RE, the commands and display/parameter data are written to the panel (handled by the state-machine for the commands/parameter data and stored in memory for the display data), or the display data/status values are read from the LCD panel (status and display data in the LCD panel memory). The polarity of A0 (RFBI_A0 signal), WE (RFBI_WR signal), RE (RFBI_RD signal), and CSx (RFBI_CSx signal, with x = 0, 1) is programmable.

Table 12-37 describes the read/write function.

Table 12-37. Read/Write Function Description

A0 (RFBI_A0)	WE (RFBI_WR)	RE (RFBI_RD)	Function Description
1	0	1	Display data write, parameter data write
1	1	0	Display data read
0	1	0	Status read
0	0	1	Command data write

A minimum of RFBI_Cs cycle time, as defined in Table 12-38, is required to keep the RFBI_CSx signal asserted between write transfers of multiple pixels.

Table 12-38 indicates the minimum cycle time for RFBI_CSx, depending on the source of pixels (display controller or L4 interconnect slave port) and the cycle format (1pixel/cycle, 1 pixel/2 cycles, 1 pixel/3 cycles, or 2 pixels/3 cycles).

Table 12-38. Minimum Cycle Time for CSx/WE Always Asserted

RFBI Performance	RFBI_CONFIGi[10:9] CYCLEFORMAT	RFBI_CONFIGi[8:7] L4FORMAT	Minimum Cycle Time (in Number of L4 Cycles)
L4 interconnect	1 pixel/cycle	1 pixel	5
	1 pixel/2 cycles	1 pixel	4
	1 pixel/3 cycles	1 pixel	4
	2 pixels/3 cycles	1 pixel	6
	1 pixel/cycle	2 pixels	4
	1 pixel/2 cycles	2 pixels	4
	1 pixel/3 cycles	2 pixels	4
	2 pixels/3 cycles	2 pixels	6
Display Controller	1 pixel/cycle	N/A	4
	1 pixel/2 cycles	N/A	3
	1 pixel/3 cycles	N/A	3
	2 pixels/3 cycles	N/A	6

12.4.7 Video Encoder Functionalities

The input formats supported by the encoders are 24-bit 4:4:4 RGB. The encoder output is the DAC (for more information, see Section 12.2, *Display Subsystem Environment*). In the display subsystem, the input format from the display controller is always 24-bit RGB. The RGB-to-YCbCr color space converter converts the 24-bit RGB pixel data to 24-bit YCbCr data.

The remaining Cb and Cr color components enter the 2-to-1 chrominance decimation, which reduces by half the chrominance bandwidth and the amount of chrominance data. After the data manager, the encoder processes in 4:2:2 data path up to the 2x interpolation. A luma delay synchronizes luma to chrominance data.

Figure 12-109 shows an overview of the video encoder architecture.

Figure 12-109. Video Encoder Architecture Overview

NOTE: Output video mode can be either composite video (CVBS output) or Separate video (s-video: Luma and chroma outputs):

- Composite video: DAC1 is used
- Separate video (luma/chroma): Both DAC1 (luma) and DAC2 (chroma) are used
  The selection is programmed with DSS.DSS_CONTROL[6] VENC_OUT_SEL bit. Composite video is the default selection.

12.4.7.1 Test Pattern Generation

For diagnostic purposes, the data manager can be forced to output 100/100 color bar RGB/YCbCr data by setting the SVDS field VENC_F_CONTROL[7:6] register to 0x1.

Table 12-39. 100/100 Color Bar Table

COLOR	R	G	B	Y	Cb	Cr
White	255	255	255	235	128	128
Yellow	255	255	0	210	16	146
Cyan	0	255	255	170	166	16
Green	0	255	0	145	54	34
Magenta	255	0	255	106	202	222
Red	255	0	0	81	90	240
Blue	0	0	255	41	240	110
Black	0	0	0	16	128	128

12.4.7.2 Luma Stage

The luma stage includes a luma pipeline delay, luma shaping, 2x interpolation filter, and luma variable delay. The luma pipeline delay block is used to match luma path length to chroma path length. In the luma gain shaper, a programmable gain is first applied to the luminance data output. The luminance gain is defined by the DSS.VENC_GAIN_Y register. Horizontal sync, vertical sync, and setup insertion are then performed.

Black level and blank level are programmable through the DSS.VENC_BLACK_LEVEL and DSS.VENC_BLANK_LEVEL registers. All the transition edges of the luminance signal, such as sync edges and active video edges, are properly shaped and filtered to keep the bandwidth within the standards.

After all required components of the luminance signal are added, the resulting signal is low-passed and interpolated to 2x-pixel rate. This 2x interpolation simplifies the external analog reconstruction filter design and improves the signal-to-noise ratio.

12.4.7.3 Chroma Stage

The chroma stage includes a low-pass filter, first-stage 2x interpolation, chroma gain shaper, and second-stage 2x interpolation. A pair of programmable gains adjusts the time-multiplexed U/V signal. The gains for U and V are independently controlled by the DSS.VENC_GAIN_U and DSS.VENC_GAIN_V register bits.

12.4.7.4 Subcarrier and Burst Generation

The encoder uses a 32-bit subcarrier increment to synthesize the subcarrier. The value of the subcarrier increments required to generate the desired subcarrier frequency for NTSC and PAL format is found by:

$S\_CARR = ROUND ([F_{sc}/F_{clkenc}] \times 2^{32})$

where:

F_{sc} = Frequency of the subcarrier

F_{clkenc} = Frequency of the internal video encoder

The DSS.VENC_S_CARR register controls the subcarrier frequency. The DSS.VENC_C_PHASE register controls the phase of the subcarrier. The phase of the color subcarrier is reset to DSS.VENC_C_PHASE. Table 12-40 presents the VENC_S_CARR register values depending the standard and pixel type used.

Table 12-40. VENC_S_CARR Register Recommended Values

Standard	Pixel Type	Subcarrier Frequency (Fsc) (MHz)	Fclkenc (MHz)	VENC_S_CARR register value (hexa)
NTSC-M, J	ITU-R601	3.579545	27	0x21F07C1F
PAL-M	ITU-R601	3.5756083125	27	0x21E6EFE3
PAL-B, D, G, H, I	ITU-R601	4.43361875	27	0x2A098ACB
NTSC-M, J	Square pixel	3.579545	24.5454	0x25555555
PAL-M	Square pixel	3.579561149	24.5454	0x1F15C01E
PAL-B, D, G, H, I	Square pixel	4.43361875	29.50	0x26798C0C

CAUTION

In square pixel mode, an external clock generator is needed to provide sampling frequencies (49.09 MHz for NTSC square pixel or 59 MHz for PAL square pixel).

The color subcarrier reset has four modes:

- No reset
- Reset every two lines
- Reset every two fields
- Reset every eight fields

The DSS.VENC_C_PHASE register can be used to adjust the SCH (subcarrier to horizontal sync phase). The DSS.VENC_BSTAMP_WSS_DATA[6:0] BSTAP bit field sets the amplitude of the color burst. The DSS.VENC_M_CONTROL[1] PAL bit enables phase alternation line encoding.

A phase switching subcarrier is generated to encode the chrominance signal when the DSS.VENC_M_CONTROL[1] PAL bit is set to 1. Otherwise, a normal subcarrier is generated. Phase alternation line refers to the encoding scheme in which the subcarrier alternates between two phases every scan line. Two possible alternation sequences are possible, and the DSS.VENC_M_CONTROL[5] PALPHS bit selects one of these sequences.

12.4.7.5 Closed Caption Encoding

The encoder can be programmed to encode closed-caption data and extended data in the selected line. The closed-caption data are sent to the encoder through the L4 interconnect. The data stream consists of 7-bit US-ASCII code and 1 odd-parity bit (see Table 12-41).

Table 12-41. Closed-Caption Data Format

MSB							LSB
Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Odd parity

The standard service encodes closed caption in both fields; the extended service encodes closed caption in even fields. When set to 1, the DSS.VENC_L21_WC_CTL L21EN[0] bit enables closed-caption encoding in odd fields. When set to 1, the DSS.VENC_L21_WC_CTL L21EN[1] bit enables closed-caption encoding in even fields.

To select the scan line where the CC data are encoded, program the VENC_LN_SEL[4:0] SLINE bit field.

CAUTION

The setting of the value of the SLINE[4:0] bit field depends on the video standard:

- PAL mode: Because there is a one-line offset, program the desired line number – 1. To activate the closed caption on line 21 (0x15), program the value 0x15 – 1 = 0x14. The default value is 0x15 + 1 = 0x16 (line 22).
- NTSC mode: Because there is a four-line offset, program the desired line number – 4. To activate the closed caption on line 21 (0x15), program the value 0x15 – 4 = 0x11. The default value is 0x15 + 4 = 0x19 (line 25).

The DSS.VENC_LN_SEL[25:16] LN21_RUNIN bit field should be kept at reset value (0x10B). Four closed-caption data registers contain the data to be encoded. The DSS.VENC_LINE21[15:8] L21O and DSS.VENC_LINE21[7:0] L21O bit fields contain the first and the second bytes, respectively, of closed-caption data to be encoded in the odd field. The DSS.VENC_LINE21[31:24] L21E and DSS.VENC_LINE21[23:16] L21E bit fields contain the first and the second bytes, respectively, of data to be encoded in the even field.

Immediately after the closed-caption data is written to the registers, in either the odd field or even field, the corresponding closed-caption status bit (DSS.VENC_STATUS[4] CCE or DSS.VENC_STATUS[3] CCO) is reset to 0 to indicate that the closed-caption data is available in the closed-caption data registers and yet to be encoded.

Immediately after the closed-caption data is encoded, the DSS.VENC_STATUS[4] CCE bit or the DSS.VENC_STATUS[3] CCO bit is set to 1 to indicate that the closed-caption data has been encoded and is ready to accept new data. As seen in Figure 12-110, a null character is automatically inserted if the closed-caption data is not written to the closed-caption data registers in time for encoding.

The running clock frequency is controlled by the DSS.VENC_CC_CARR_WSS_CARR[15:0] FCC bit field which should be kept at reset value (0x2631) to get 5034960.5Hz (32xfline) for NTSC-601. The closed-caption running clock common frequencies are detailed in Table 12-42.

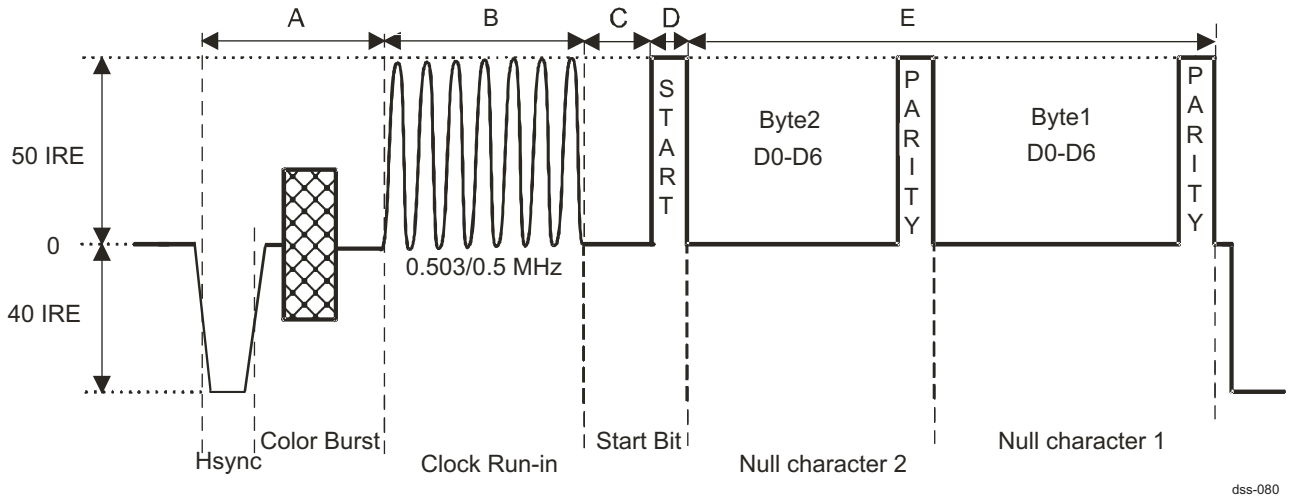
Table 12-42. Closed-Caption Run Clock Frequency Settings

	NTSC-601	PAL-601	NTSC Square Pixel	PAL Square Pixel
VENC_CC_CARR_WSS_CARR[15:0] FCC bit field value	0x2631	0x25ED	0x2A03	0x22B6

The closed-caption data is encoded in nonreturn-to-zero (NRZ) format. Additionally, the data translates to the IRE scale as follows: 0 = 0 IRE; 1 = 50 IRE.

Figure 12-110 shows the parameters of closed-caption line data implemented in different standards.

Figure 12-110. Closed Captioning Timing



NOTE:

- The interval A is controlled by the DSS.VENC_LN_SEL[25:16] LN21_RUNIN bit field.
- The interval B is controlled by DSS.VENC_CC_CARR_WSS_CARR[15:0] FCC bit field.

Table 12-43. Closed-Caption Standard Timing Values

Intervals	Description	Timing Values for Encoding			Timing Values for Decoding		
		Minimal	Nominal	Maximal	Lower Bound	Nominal	Upper Bound
A	HSYNC to clock running	10.250 µs	10.500 µs	10.750 µs	10.000 µs	10.500 µs	11.000 µs
B	Clock running		12.910 µs			12.910 µs	
C	Clock running to third start bit		3.972 µs			3.972 µs	
D	Start bit		1.986 µs			1.986 µs	
E	Data characters		31.778 µs			31.778 µs	

NOTE: All timing values listed in Table 12-43 are measured from the mid-point (half amplitude) on all edges.

For a complete description of copy protection including CGMS-A, please refer to CEA-608-x standard

12.4.7.6 Wide-Screen Signaling (WSS) Encoding

The encoder can embed data, encoded in accordance with the IEC61880 and ITU-R 1119 data insertion standard, within the vertical blanking interval.

The encoder supports WSS data insertion on line 20 of every frame in the NTSC format. WSS data insertion is enabled by activating the DSS.VENC_L21_WC_CTL[14:13] EVEN_ODD_EN bit and by programming the VENC_BSTAMP_WSS_DATA[27:8] WSS_DATA bit field.

The running clock frequency is controlled by the DSS.VENC_CC_CARR_WSS_CARR[31:16] FWSS bit field. The wide-screen signaling running clock common frequencies are detailed in Table 12-44

Table 12-44. Wide-Screen Signaling Run Clock Frequency Settings

	NTSC-601	PAL-601	NTSC Square Pixel	PAL Square Pixel
VENC_CC_CARR_WSS_CARR[31:16] FWSS bit field value	0x043F	0x2F72	0x04AC	0x2B6D

To select the line where the WSS data are encoded, program the DSS.VENC_L21_WC_CTL[12:8] LINE bit field.

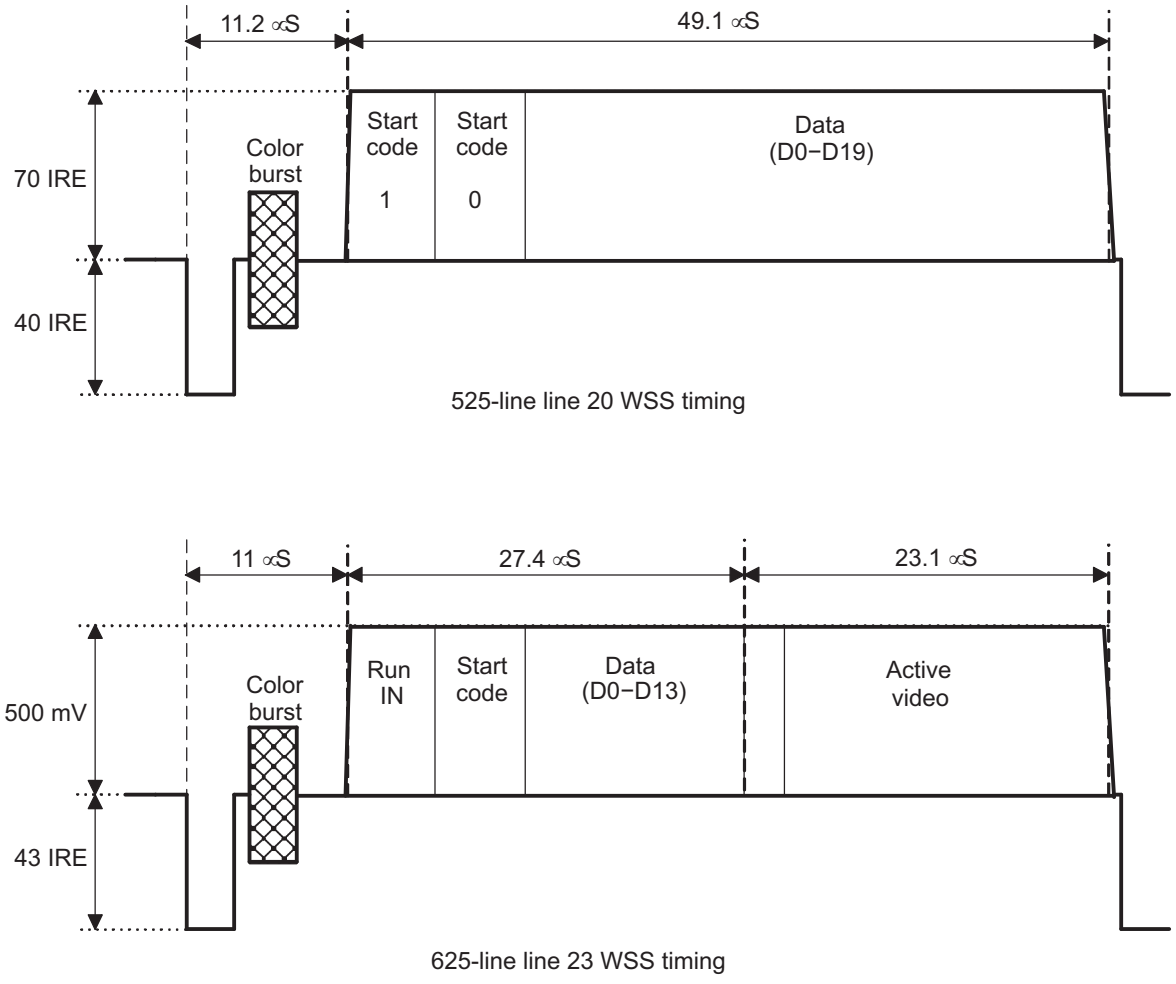
CAUTION

The setting of the LINE[12:8] bit field value depends on the video standard:

- PAL mode: There is an one line offset, so program the wanted line number - 1. The recommended value is line 0x16 + 1 = 0x17 (23rd line). Note that the default value is 0x14 + 1 = 0x15 (21st line).
- NTSC mode: There is a four line offset, so program the wanted line number - 4. The recommended value is line 0x10 + 4 = 0x14 (20th line). Note that the default value is 0x14 + 4 = 0x18 (24th line).

The WSS encoding block assumes that a full 10-bit video range is used to determine the 70 percent of peak-white amplitude of a logic-1 bit. The encoder also supports WSS data insertion on line 23 in the PAL format. Both waveforms are shown in Figure 12-111.

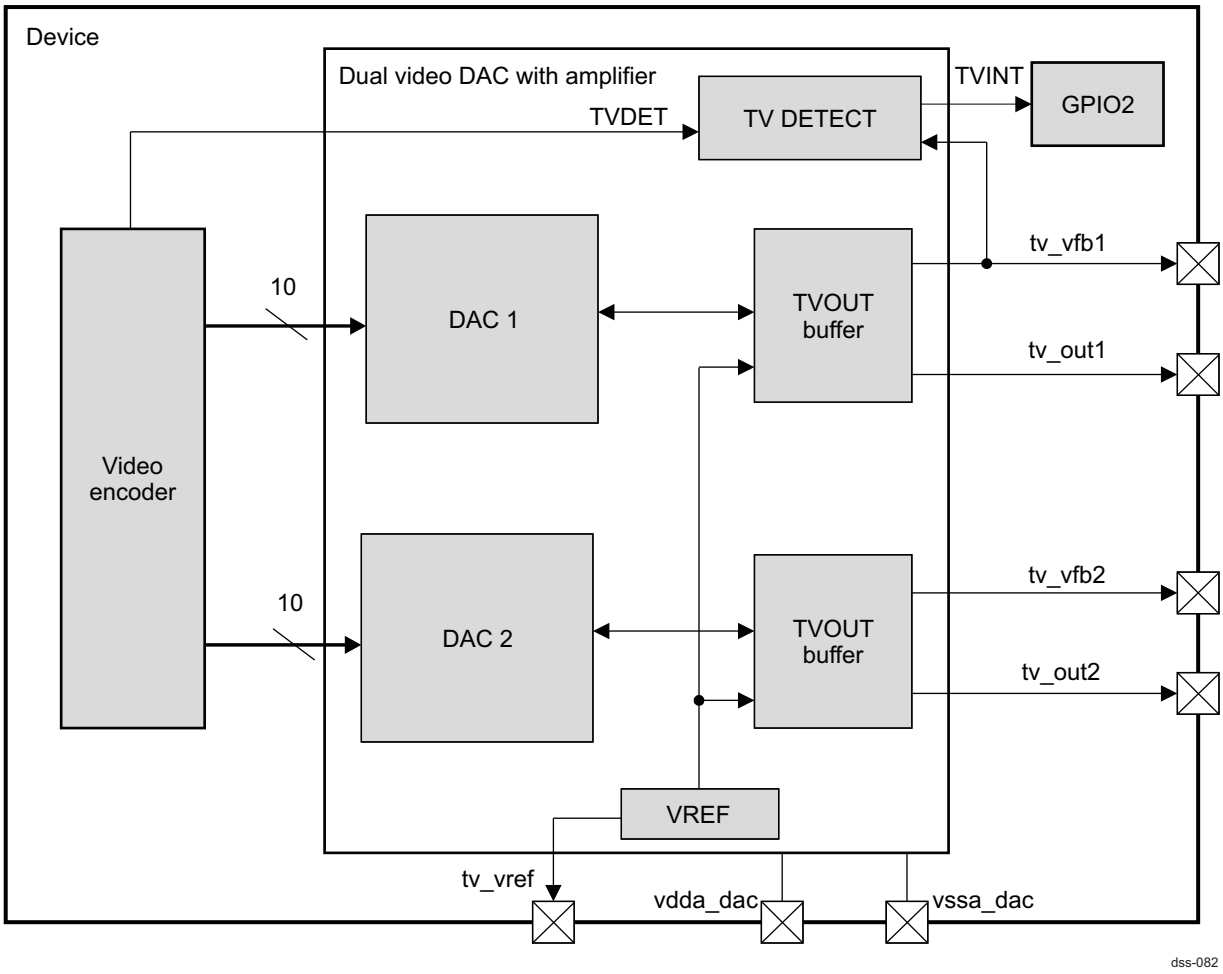
Figure 12-111. WSS Timing



12.4.7.7 Video DAC Architecture

Figure 12-112 shows the architecture of the dual 10-bit video DAC.

Figure 12-112. Dual 10-Bit Video DAC Architecture



The display subsystem provides the necessary control signals to interface the memory frame buffer directly to external displays (TV sets). Two (one per channel) 10-bit current steering DAC is used to generate the video analog signal. The video mode DAC 1 (luma/composite) also includes the TV detection/disconnection and power-down mode features.

12.4.7.8 Video DC/AC Coupled TV Load

The dual 10-bit video DAC supports both dc-coupled and ac-coupled TV loads. The CONTROL.CONTROL_DEVCONF1[11] TVACEN bit is used to define which output coupling is used (0: Dc coupling; 1: Ac coupling). This bit is the first one to be programed according to the TV load on the PCB board.

NOTE: When dc coupling is used, note that there is a 385-mV dc offset at the TVOUT output (tv_out1 for DAC1 and tv_out2 for DAC2).

12.4.7.9 TV Detection/Disconnection Pulse Generation and Usage

12.4.7.9.1 TV Detection/Disconnection Pulse Generation

The TV detect block is an integral part of the dual video DAC module.

NOTE:

- The TV detection/disconnection feature is supported only for VDAC1.
- The TV disconnection feature is recommended for power saving purpose. The TV detection/disconnection is only operational when video out is active. Therefore to detect cable connection automatically, it is necessary to periodically activate the video out to test for cable presence.

This block compares the output (tv_out1) to reference (tv_vref) voltages, to sense the condition of the load. To operate, the TV detect requires two digital signals, TVACEN and TVDET. The TVACEN signal indicates to both TVOUT buffer and the TV detect circuit if the load is ac or dc coupling to adjust accordingly. TVDET is a digital pulse at the frequency of the TV sync pulses. The operation of the circuit is based on the difference in voltage levels in the output of the buffer depending of the load status. The TV detect block compares the output against a couple of references and the result is latched at the start of every sync pulse. The status is read later in with the TVDET pulse rising edge.

The following registers are used to set the TV detection/disconnection pulse:

- The DSS.VENC_TVDETGP_INT_START_STOP_X register defines which pixels are used to start and stop the pulse inside their respective line.
- The DSS.VENC_TVDETGP_INT_START_STOP_Y register defines which lines are used to start and stop the pulse.
- The DSS.VENC_GEN_CTRL[0] EN bit enables or disables the TVDET pulse (0: Disable; 1: Enable).
- The DSS.VENC_GEN_CTRL[16] TVDP bit sets the TVDET pulse polarity (0: Active low; 1: Active high).

12.4.7.9.2 TV Detection Procedure

The TV detection procedure is the following:

1. Initial setup:
   - Program the DSS.VENC_TVDETGP_INT_START_STOP_X and DSS.VENC_TVDETGP_INT_START_STOP_Y registers to define on which pixels and lines, respectively, the TVDET pulse will start and stop.
   - Set the DSS.VENC_GEN_CTRL[16] TVDP bit to 1 (reset value) for a TVDET pulse active high polarity.
   - Set the DSS.VENC_GEN_CTRL[0] EN bit to 1 to enable the TVDET pulse generation.
2. The TVDET signal is set to low by hardware according to the settings in the DSS.VENC_TVDETGP_INT_START_STOP_X and DSS.VENC_TVDETGP_INT_START_STOP_Y registers.
3. Set the VENC_OUTPUT_CONTROL[0] LUMA_ENABLE bit (in s-video mode) or the VENC_OUTPUT_CONTROL[1] COMPOSITE_ENABLE (in composite video mode) to 1 to enable the video DAC1 output.
4. Power up the vdda_dac voltage for VDAC and the TVOUT buffer (repeat every TV field during the horizontal synchronization). This is software controlled through an I^2C interface connected to the power IC .
5. The TVDET pulse is set high by hardware according to the settings in the DSS.VENC_TVDETGP_INT_START_STOP_X and DSS.VENC_TVDETGP_INT_START_STOP_Y registers.
6. Check the TVINT output signal: When TVINT is set to 1, the load is connected.

CAUTION

- If ac coupling is selected, two TVDET pulses are required to set high the TVINT signal. Due to the internal logic of the video DAC1, the TVINT signal is generated after the next positive edge of the TVDET signal that happens during the next VSYNC timing.
- If dc coupling is selected, only one TVDET pulse is required to set high the TVINT signal.

12.4.7.9.3 TV Disconnection Procedure

The TV disconnection procedure is the following:

1. Initial setup:
   - Program the DSS.VENC_TVDETGP_INT_START_STOP_X and DSS.VENC_TVDETGP_INT_START_STOP_Y registers to define on which pixels and lines, respectively, the TVDET pulse will start and stop.
   - Set the DSS.VENC_GEN_CTRL[16] TVDP bit to 1 (reset value) for a TVDET pulse active high polarity.
   - Set the DSS.VENC_GEN_CTRL[0] EN bit to 1 to enable the TVDET pulse generation.
2. The TVDET signal is set to low by hardware according to the settings in the DSS.VENC_TVDETGP_INT_START_STOP_X and DSS.VENC_TVDETGP_INT_START_STOP_Y registers.
3. The TVDET pulse is set high by hardware according to the settings in the DSS.VENC_TVDETGP_INT_START_STOP_X and DSS.VENC_TVDETGP_INT_START_STOP_Y registers.
4. Check the TVINT output signal: When TVINT is reset to 0, the load is disconnected.
5. Reset the VENC_OUTPUT_CONTROL[0] LUMA_ENABLE bit (in s-video mode) or the VENC_OUTPUT_CONTROL[1] COMPOSITE_ENABLE (in composite video mode) to 0 to disable the video DAC1 output.
6. Power-down the vdda_dac voltage for VDAC and the TVOUT buffer (repeat every TV field during the horizontal synchronization). This is software controlled through an I^2C interface connected to the power IC.

CAUTION

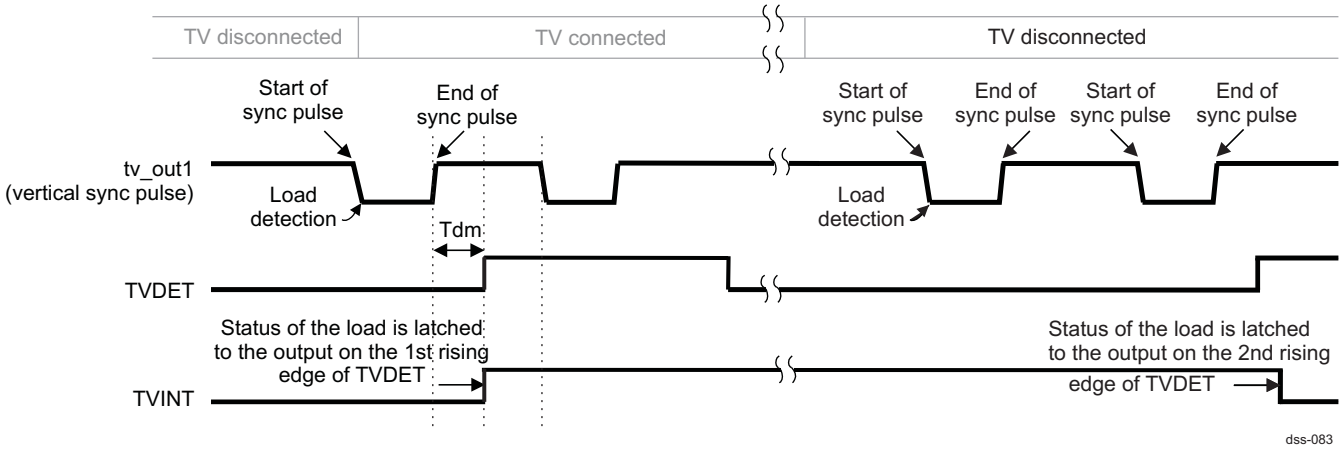
- If dc coupling is selected, two TVDET pulses are required to set low the TVINT signal. Due to the internal logic of the video DAC1, the TVINT signal is generated after the next positive edge of the TVDET signal that happens during the next VSYNC timing.
- If ac coupling is selected, only one TVDET pulse is required to set low the TVINT signal.

12.4.7.9.4 Recommended TV Detection/Disconnection Pulse Waveform

To enable the detection/disconnection of the load, the circuit requires that the TVDET pulse resembles the following waveform. As explained in Section 12.4.7.9.2, *TV Detection Procedure*, and in Section 12.4.7.9.3, *TV Disconnection Procedure* by using the video encoder registers, the TVDET pulse polarity, start and stop is programmable. The only critical parameter is Tdm, which should be longer than the delay through the DAC and TVOUT buffer, which is at least 750 ns.

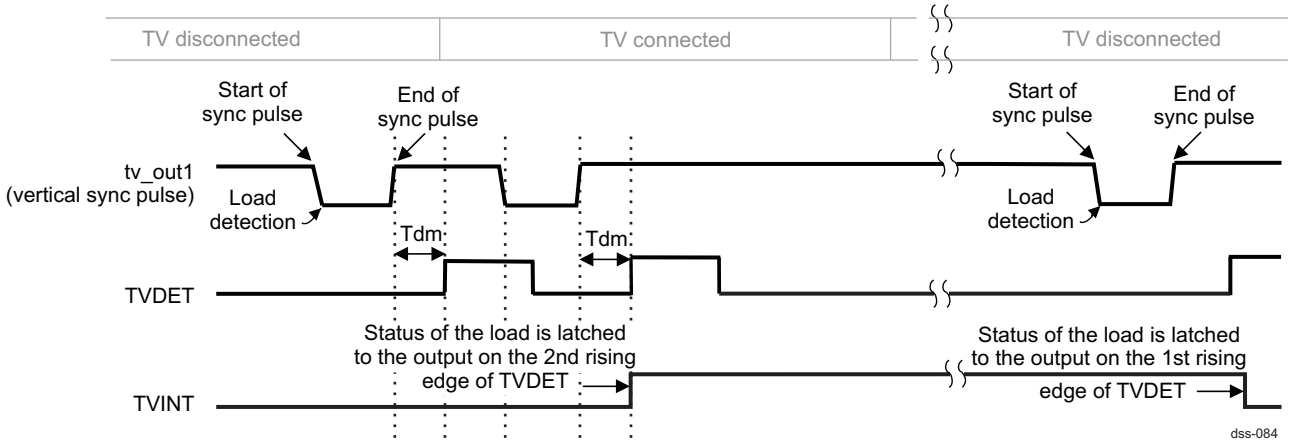
If dc-coupling is selected, the TVINT output signal for TV detection is latched at the rising edge of the first TVDET signal but the TVINT output signal for TV disconnection is latched after the next rising edge of the TVDET signal that happens during the next VSYNC timing. Figure 12-113 shows the waveforms for the dc-coupling TV detect pulse (TVDET) when load is connected and disconnected.

Figure 12-113. DC-Coupling TV Detect Waveforms for TV Connected and Disconnected



If ac-coupling is selected, the TVINT output signal for TV detection is latched after the next rising edge of the TVDET signal that happens during the next VSYNC timing but the TVINT output signal for TV disconnection is latched at the rising edge of the first TVDET signal. Figure 12-114 shows the waveforms for the ac-coupling TV detect pulse (TVDET) when load is connected and disconnected.

Figure 12-114. AC-Coupling TV Detect Waveforms for TV Connected and Disconnected



NOTE:

- When setting the DSS.VENC_TVDETGP_INT_START_STOP_X register, software users must ensure that the TVDET signal is in the active area. To avoid any problem, the TVDET signal must not be longer than one line.
- The activation of the TVDET signal will not have a visual impact on the tv_out1 output signal.

12.4.7.9.5 TV Detection/Disconnection Usage

The TV-detection/TV-disconnection is based on the difference in voltage levels in the output of the TV buffer depending on the load status. The operation is slightly different for ac and for dc operation. For dc operation, the tv_out1 voltage is compared against a voltage reference (tv_vref) that makes the comparator trigger in each sync pulse while the load is connected. For ac operation, the tv_out1 voltage is compared against a voltage reference (tv_vref) that makes the comparator trigger in each sync pulse while the load is disconnected. In both cases, the TVINT output signal produces a logic 1 when a load is connected and a logic 0 when a load is disconnected.

For dc-coupling mode, see Figure 12-115 and for ac-coupling mode, see Figure 12-116

NOTE: Because the video DAC and the video encoder must be awake for connection detection, consider that the video DAC can take up to 10 µs to wake up.

Figure 12-115. GPIO Signal Waveform Proposal for TV Detection/Disconnection in DC-Coupling Mode

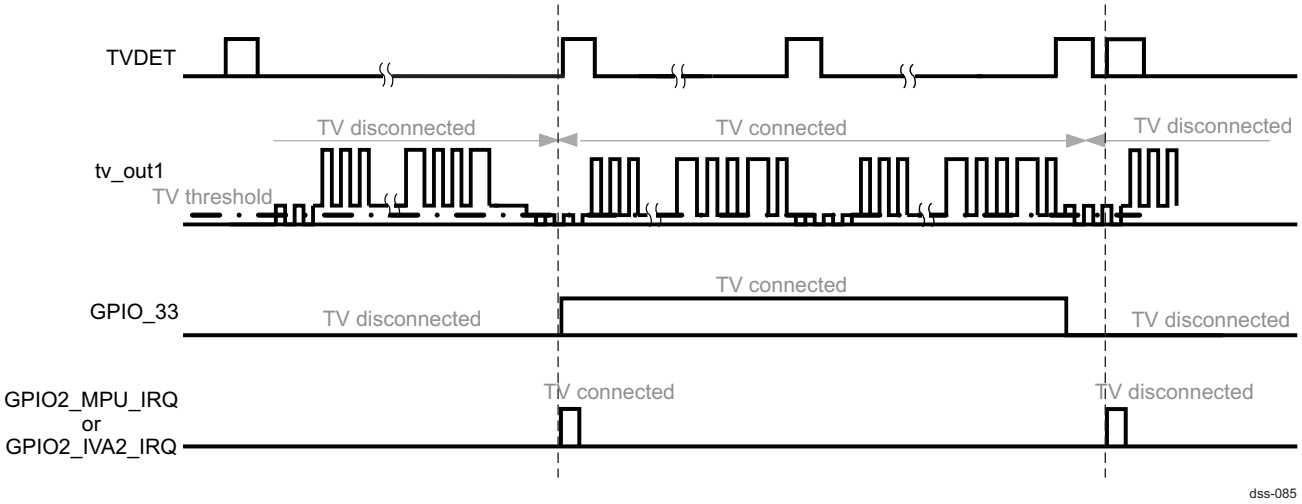
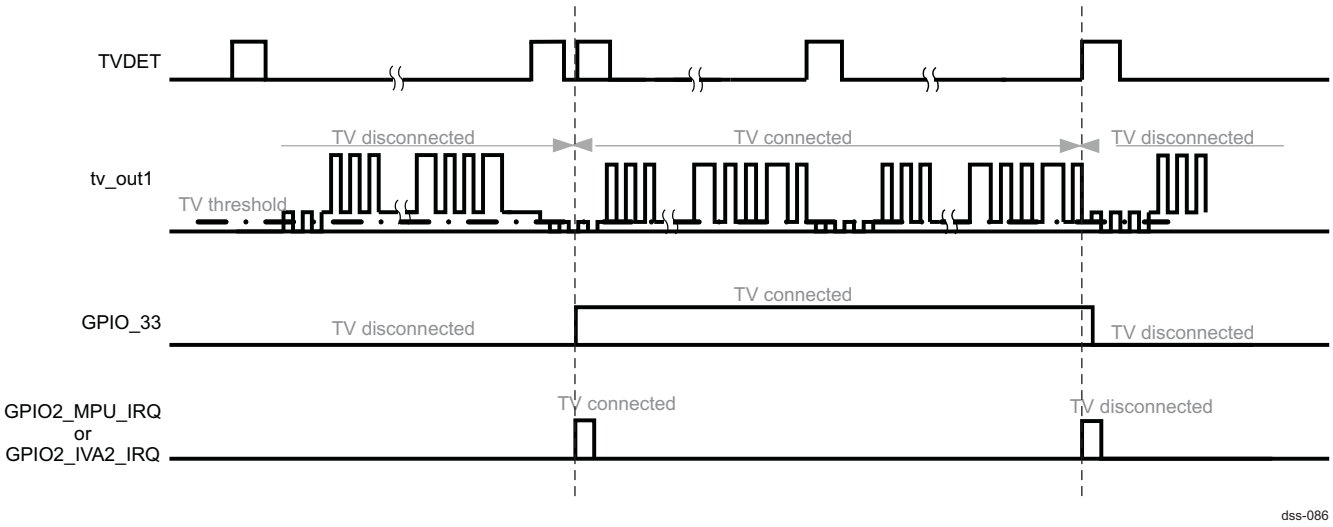


Figure 12-116. GPIO Signal Waveform Proposal for TV Detection/Disconnection in AC-Coupling Mode



12.4.7.10 Video DAC Bypass Mode

The dual 10-bit video DAC has a TVOUT buffer bypass mode that turns off the TVOUT buffer and redirects directly the DAC output to the VFB pins (tv_vfb1 for DAC1 and tv_vfb2 for DAC2).

This bypass mode is activated by setting the CONTROL.CONTROL_DEVCONF1[18] TVOUTBYPASS bit to 1. The reset value of the CONTROL.CONTROL_DEVCONF1[18] TVOUTBYPASS bit is 0 (that is, the TVOUT buffer is not bypassed).

NOTE: In bypass mode:

- the tv_out1 pin requires a RSET resistor connected to the ground. The typical value of the RSET resistor is 3.92 K.
- both tv_vfb1and tv_vfb2 pins require a RLOAD resistor connected to the ground. The typical value of the RLOAD resistor is 976.

CAUTION

- The TV detect feature is not available in bypass mode.
- In bypass mode, an external amplifier is needed on the tv_vbf1 and tv_vbf2 pins.

12.4.7.11 Video Dual-DAC Test Mode

The DAC can be tested for debug using either 10-bit external data or 10-bit internal register values directly connected to the DAC. See Figure 12-117 for video DAC test in composite video mode, and see Figure 12-118 for video DAC test in separate video mode.

Figure 12-117. DAC Test Mode in Composite Video Mode

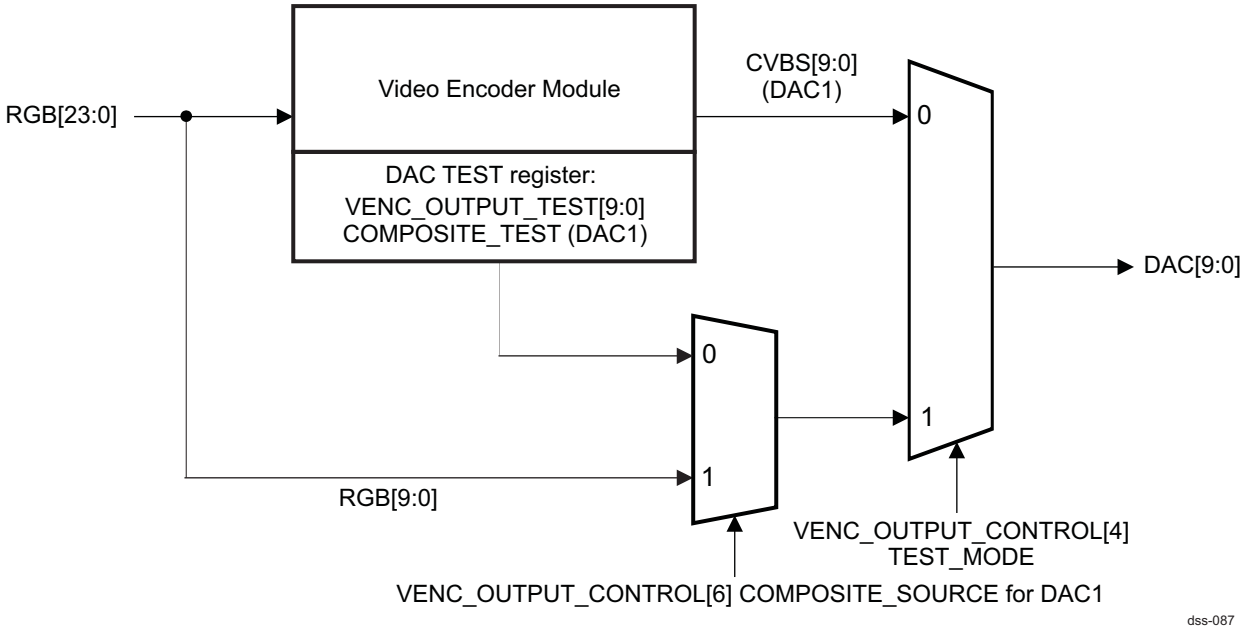
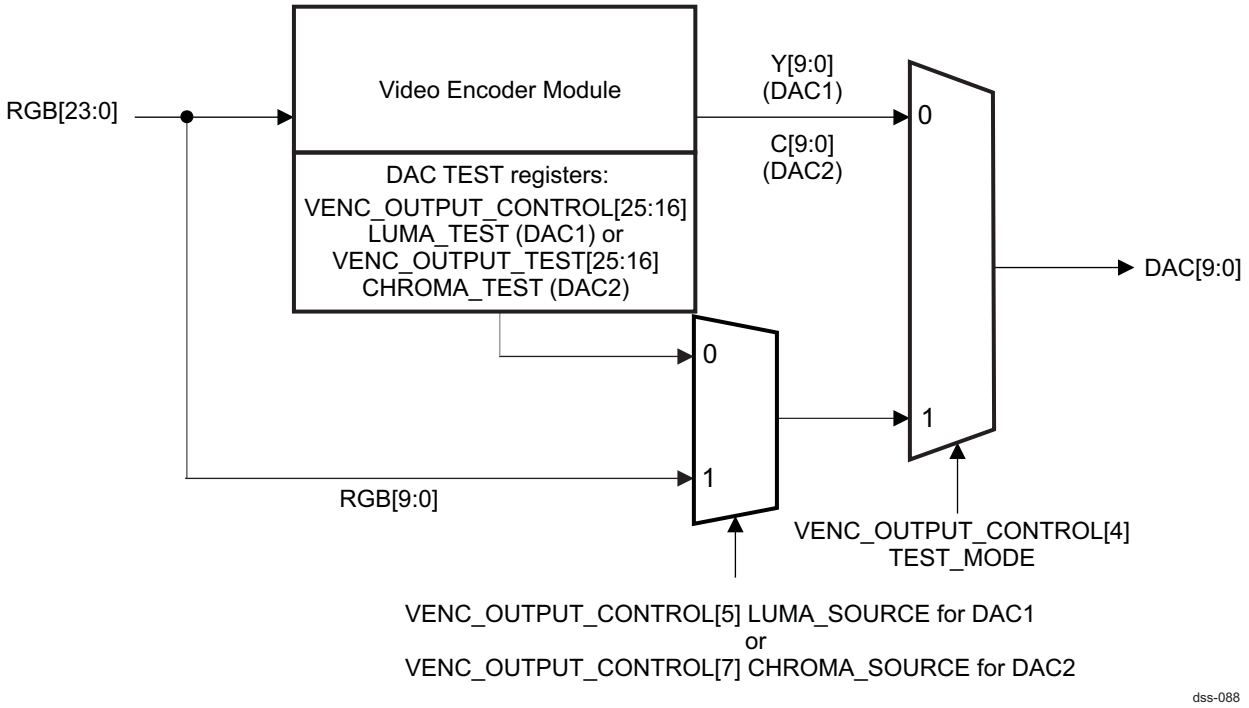


Figure 12-118. DAC Test Mode in Separate video Mode



- Use the DSS.VENC_OUTPUT_CONTROL[4] TEST_MODE bit to select between DAC normal mode (0x0) and DAC test mode (0x1).
- Use the DSS.VENC_OUTPUT_CONTROL[7] CHROMA_SOURCE bit for DAC2 and either the DSS.VENC_OUTPUT_CONTROL[6] COMPOSITE_SOURCE bit (in composite video mode) or the DSS.VENC_OUTPUT_CONTROL[5] LUMA_SOURCE bit (in s-video mode) for DAC1 to select the test mode:
  - 0x0: From the internal register DSS.VENC_OUTPUT_TEST[25:16] CHROMA_TEST bit field for DAC2 and either DSS.VENC_OUTPUT_TEST[9:0] COMPOSITE_TEST bit field (composite video) or DSS.VENC_OUTPUT_CONTROL[25:16] LUMA_TEST (s-video mode) for DAC1
  - 0x1: From the video port G[1:0], B[7:0]

NOTE: In the external data test mode (bypass mode), the display controller must provide the data (G[1:0], B[7:0]) externally. To do this, configure the video encoder to generate correct timing signals, without which the display controller cannot operate (even if the encoder core is bypassed from the data path perspective).

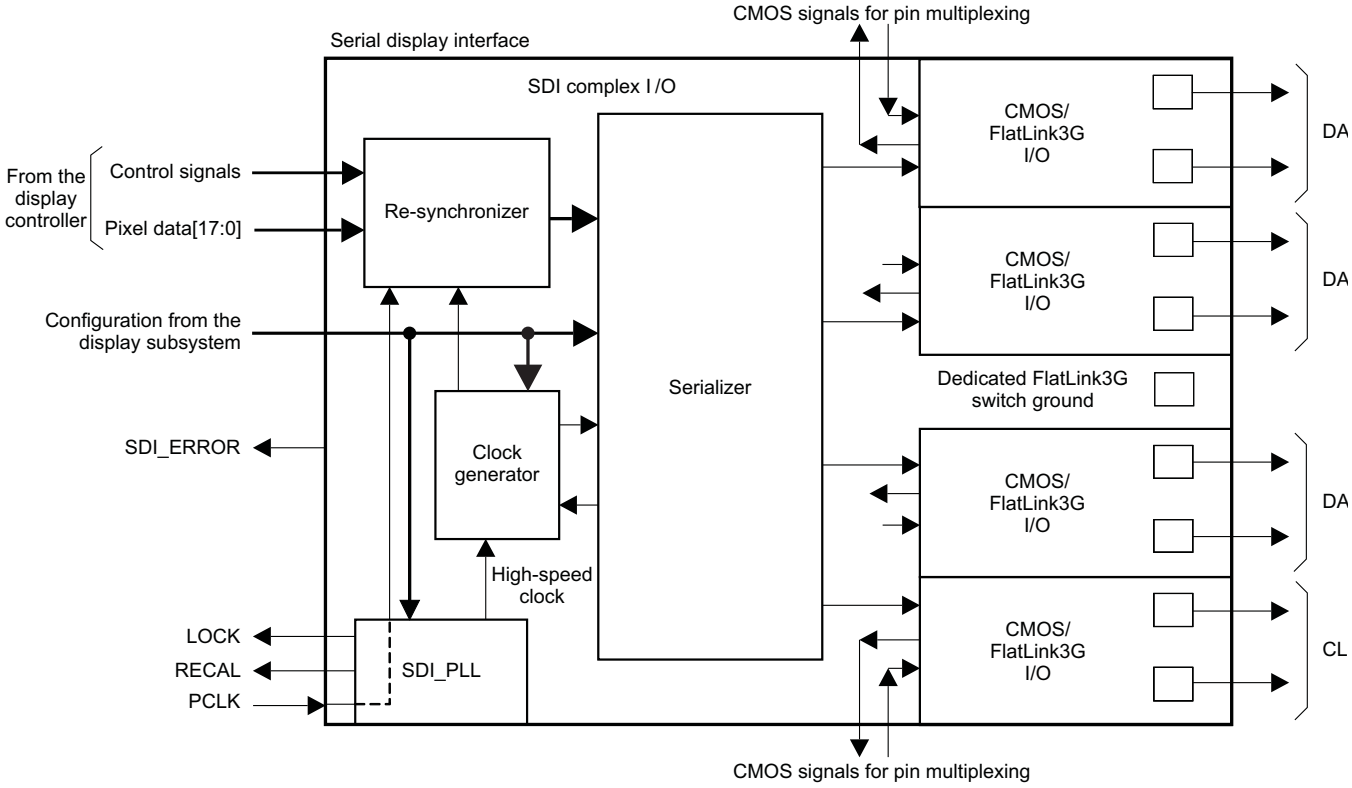
12.4.8 SDI Functionalities

The SDI interface has three main functions:

- A PLL (SDI_PLL) to multiply or divide the pixel clock frequency (PCLK) up to the required data rate. The pixel clock frequency range is 4-65 MHz. This clock may be divided by 2 above 32 MHz to meet the PLL input reference timing requirements.
- A specific IP block to output the data in the chosen serial format
- Output buffers to transmit the small swing differential signals

  The last two functions are combined in the complex I/O module. Figure 12-119 shows the SDI architecture.

Figure 12-119. SDI Architecture Overview



NOTE:

- The RECAL status indicates when the SDI_PLL must be recalibrated (0: No recalibration is required; 1: Recalibration is required).

  This status is connected to the GPIO_80 internal signal of the GPIO3 module. For additional information, see Chapter 21, *General-Purpose Interface*.
- The LOCK status indicates when the SDI_PLL is locked (0: PLL not locked; 1: PLL locked). This status is connected to the GPIO_81 internal signal of the GPIO3 module. For additional information, see Chapter 21, *General-Purpose Interface*.

For more information about lock and recalibration, see , *Power, Reset, and Clock Management*.

- The SDI_ERROR status indicated when the SDI module is in error mode (0: No error; 1: SDI in error). This status allows monitoring error conditions, such as inconsistent configuration or internal loss of synchronization. It is connected to the GPIO_82 internal signal of the GPIO3 module. For additional information, see Chapter 21, *General-Purpose Interface*.

12.5 Display Subsystem Basic Programming Model

This section describes how to configure the display subsystem for the desired functionalities and also describes the programming models of the display controller, the RFBI, the SDI and the video encoder.

The main configuration scenarios are:

- LCD panel support (bypass or RFBI mode)

  Configure the RFBI module (only if in RFBI mode; otherwise, the default values must remain), and then configure the display controller to the desired functionalities before the activities start.
- TV set support

  Configure the video encoder and then the display controller.
- Both LCD panel support (bypass or RFBI mode) and TV set support

  Configure the RFBI module (only if in RFBI mode; otherwise, leave the default values), configure the video encoder, and then configure the display controller.
- TI FlatLink 3G-compliant LCD panel support

  Configure the SDI, configure the video encoder, and then configure the display controller.

12.5.1 Display Subsystem Reset

The display subsystem can receive a software reset that is propagated through all of the submodules to initialize the subsystem. The following procedure describes a possible sequence:

1. If the LCD is on, stop the LCD by setting the DSS.DISPC_CONTROL[0] LCDENABLE bit to 0.

   (a) Reset the frame done status bit by writing 1 in the DSS.DISPC_IRQSTATUS[0] FRAMEDONE bit.

   (b) Wait until the DSS.DISPC_IRQSTATUS[0] FRAMEDONE bit is set to 1. This shows that the end of frame has taken place and the LCD stop is complete.
2. To take the display subsystem out of reset, all clocks related to the display subsystem must be enabled and the DPLL4 must be enabled. The following clocks must be enabled to take the display subsystem out of reset:
   - PRCM.CM_FCLKEN_DSS[0] EN_DSS1 bit set to 1
   - PRCM.CM_FCLKEN_DSS[1] EN_DSS2 bit set to 1
   - PRCM.CM_FCLKEN_DSS[2] EN_TV bit set to 1
   - PRCM.CM_ICLKEN_DSS[0] EN_DSS bit set to 1

   Once the clocks are enabled as shown, the display subsystem can be taken out of reset.
3. Write 1 in the DSS.DSS_SYSCONFIG[1] SOFTRESET bit to apply the soft reset to the subsystem.
4. Read the DSS.DSS_SYSSTATUS[0] RESETDONE bit. If this bit is 1, the reset sequence is complete; otherwise, read this bit again (the reset sequence is not completed).

12.5.2 Display Subsystem Configuration Phase

The display subsystem configuration phase is important to configure the data flow for using the LCD panel or the TV set. Use the following flow:

1. To configure the top level of the functional clock of the display controller clock, set the DSS.DSS_CONTROL[0] DSS_CLK_SWITCH bit.
2. To configure the top level of the video encoder, set the DSS.DSS_CONTROL[2] VENC_CLOCK_MODE bit and the DSS.DSS_CONTROL[3] VENC_CLOCK_X4 bit for TV set support.
3. To configure the top level of the DAC, set the DSS.DSS_CONTROL[4] DAC_DEMEN bit for TV set support (if required).
4. Configure the TI FlatLink 3G-compliant LCD panel through the SPI interface to use the serial interface display.
5. Configure the RFBI module and/or the SDI module and/or the video encoder as needed.
6. Configure the display controller.

12.5.3 Display Controller Basic Programming Model

Some display controller registers are termed *shadow registers*, which are associated with the digital output and/or the LCD output. A shadow register change has no direct effect on the configuration of the display controller unless the DSS.DISPC_CONTROL[5] GOLCD bit is set for the LCD output and/or the DSS.DISPC_CONTROL[6] GODIGITAL bit is set for the digital output.

In the case of the digital output, after programming the shadow registers, the DSS.DISPC_CONTROL[6] GODIGITAL bit must be set to 1. If this bit is not set, the configuration of the display controller will have no effect. This setting indicates that all display controller shadow registers are programmed and that hardware can update the internal registers at the external EVSYNC.

In the case of the LCD output, after programming the shadow registers, the DSS.DISPC_CONTROL[5] GOLCD bit must be set to 1. If this bit is not set, the configuration of the display controller will have no effect. This setting indicates that all display controller shadow registers are programmed and that hardware can update the internal registers at the VFP start period.

Before setting either the DSS.DISPC_CONTROL[5] GOLCD or DSS.DISPC_CONTROL[6] GODIGITAL bit, ensure that the bit is cleared.

Table 12-45 lists the shadow registers.

Table 12-45. Shadow Registers

Shadow Register Name	Updated on VFP Start Period (LCD output)	Updated on External VSYNC (Digital output)
DSS.DISPC_CONTROL	X(1)	X(1)
DSS.DISPC_CONFIG	X	X
DSS.DISPC_DEFAULT_COLOR_m (m = 0)	X	
DSS.DISPC_DEFAULT_COLOR_m (m = 1)		X
DSS.DISPC_TRANS_COLOR_m (m = 0)	X	
DSS.DISPC_TRANS_COLOR_m (m = 1)		X
DSS.DISPC_LINE_NUMBER	X	
DSS.DISPC_TIMING_H	X	
DSS.DISPC_TIMING_V	X	
DSS.DISPC_POL_FREQ	X	
DSS.DISPC_DIVISOR	X	
DSS.DISPC_SIZE_DIG		X
DSS.DISPC_SIZE_LCD	X	
DSS.DISPC_GFX_BAj (j = 0,1)	X	X
DSS.DISPC_GFX_POSITION	X	X
DSS.DISPC_GFX_SIZE	X	X
DSS.DISPC_GFX_ATTRIBUTES	X	X
DSS.DISPC_GFX_FIFO_THRESHOLD	X	X
DSS.DISPC_GFX_ROW_INC	X	X
DSS.DISPC_GFX_PIXEL_INC	X	X
DSS.DISPC_GFX_WINDOW_SKIP	X	X
DSS.DISPC_GFX_TABLE_BA	X	X
DSS.DISPC_GFX_PRELOAD	X	X
DSS.DISPC_CPR_COEF_R	X	
DSS.DISPC_CPR_COEF_G	X	
DSS.DISPC_CPR_COEF_B	X	
DSS.DISPC_VIDn_BAj (j= 0,1)	X	X
DSS.DISPC_VIDn_POSITION	X	X
DSS.DISPC_VIDn_SIZE	X	X
DSS.DISPC_VIDn_ATTRIBUTES	X	X

(1) Some of the register bit fields are shadow bits. For more information, see Section 12.7, *Display Subsystem Register Manual*.

Table 12-45. Shadow Registers (continued)

Shadow Register Name	Updated on VFP Start Period (LCD output)	Updated on External VSYNC (Digital output)
DSS.DISPC_VIDn_FIFO_THRESHOLD	X	X
DSS.DISPC_VIDn_ROW_INC	X	X
DSS.DISPC_VIDn_PIXEL_INC	X	X
DSS.DISPC_VIDn_FIR	X	X
DSS.DISPC_VIDn_PICTURE_SIZE	X	X
DSS.DISPC_VIDn_ACCUl (l = 0,1)	X	X
DSS.DISPC_VIDn_FIR_COEF_Hi (i = 0 to 7)	X	X
DSS.DISPC_VIDn_FIR_COEF_HVi (i = 0 to 7)	X	X
DSS.DISPC_VIDn_FIR_COEF_Vi (i = 0 to 7)	X	X
DSS.DISPC_VIDn_CONV_COEFi (i = 0 to 4)	X	X
DSS.DISPC_VIDn_PRELOAD	X	X
DSS.DISPC_DATA_CYCLEk (k = 0 to 3)	X	

12.5.3.1 Display Controller Configuration

The following registers define the display controller configuration:

- DSS.DISPC_SYSCONFIG
- DSS.DISPC_SYSSTATUS
- DSS.DISPC_IRQSTATUS
- DSS.DISPC_IRQENABLE

12.5.3.2 Graphics Layer Configuration

The graphics layer configuration is common to the LCD and the TV set.

12.5.3.2.1 Graphics DMA Registers

The following registers define the graphics DMA engine configuration:

- DSS.DISPC_CONTROL
- DSS.DISPC_GFX_BAj
- DSS.DISPC_GFX_ATTRIBUTES
- DSS.DISPC_GFX_ROW_INC
- DSS.DISPC_GFX_PIXEL_INC
- DSS.DISPC_GFX_FIFO_THRESHOLD
- DSS.DISPC_GFX_TABLE_BA

The following fields define the attributes of the graphics DMA engine:

- Graphics layer enable (DSS.DISPC_GFX_ATTRIBUTES[0] GFXENABLE bit): The default value of this bit at reset time is 0x0 (Disabled). The graphics DMA engine fetches encoded pixels from the system memory only when the graphics layer is enabled (a valid configuration is programmed for the graphics layer). The graphics window is present and the graphics pipeline is active.
- Burst size (DSS.DISPC_GFX_ATTRIBUTES[7:6] GFXBURSTSIZE field): The default burst size at reset time is 4 x 32 bytes. The possible values are 4 x 32, 8 x 32, and 16 x 32 bytes. The burst size is initialized at boot time by the software and never changes as long as the display controller is enabled. This field indicates the maximum burst size for the specific pipeline. In case of misalignment, the DMA engine may issue single and/or smaller burst requests because the burst size must be aligned to the burst boundary.
- Preload configuration (DSS.DISPC_GFX_ATTRIBUTES[11] GFXFIFOPRELOAD bit): The default preload configuration uses the DSS.DISPC_GFX_PRELOAD register value (the reset value is 256

bytes) to define the number of bytes to be fetched from system memory into the display controller graphics FIFO. By programming the DSS.DISPC_GFX_ATTRIBUTES[11] GFXFIFOPRELOAD bit, software users select between preload register (with 256 bytes as the reset value) and the high threshold value for preload of the encoded pixels. For best performance, the configuration of thresholds is defined using the FIFO size (in bytes) minus 1 for the high threshold, and the FIFO size (in bytes) minus the burst size (in bytes) for the low threshold, which provides 960, 992, and 1008, respectively, for burst sizes 16x32, 8x32, and 4x32. Note also that the preload value is defined based on the following display types:

  – Active matrix (TFT) display: DSS.DISPC_GFX_PRELOAD[11:0] PRELOAD = 0x60 (value is 96)
  – Color passive matrix (STN) display: DSS.DISPC_GFX_PRELOAD[11:0] PRELOAD = 0x72 (value is 114)
  – Monochrome passive matrix (STN) display: DSS.DISPC_GFX_PRELOAD[11:0] PRELOAD = 0xE0 (value is 224)

- Base address of the graphics buffer in system memory (DSS.DISPC_GFX_BAj registers): The default value of these two registers at reset time is 0x0. The horizontal resolution is one pixel because the base address is aligned on a pixel size boundary. In case of 4 BPP, the resolution is two pixels; for 2 BPP, resolution is four pixels; for 1 BPP, resolution is eight pixels; and for RGB24 packed format, the resolution is four pixels. The vertical resolution is one line. The register DSS.DISPC_GFX_BA0 defines the base address of the even field; and DSS.DISPC_GFX_BA1 defines the base of the odd field in the case of an external synchronization and based on the value of the input signal DISPC_FID and the polarity. To improve system throughput, the base address should be aligned on the burst size boundary.
- Graphics FIFO threshold (DSS.DISPC_GFX_FIFO_THRESHOLD register): The low threshold (DSS.DISPC_GFX_FIFO_THRESHOLD[11:0] GFXFIFOLOWTHRESHOLD) and the high threshold (DSS.DISPC_GFX_FIFO_THRESHOLD[27:16] GFXFIFOHIGHTHRESHOLD) values define the FIFO DMA behavior. When the low level is reached, one or more requests are issued to the L3-based interconnect to fill up the FIFO to reach the high threshold. A request is issued as long as the FIFO has enough space available to accept a burst. The DMA engine then waits until the low level is reached to restart the requests. By setting the DSS.DISPC_CONFIG[14] FIFOMERGE bit to 1, users merge the three FIFOs (GFX, VID1, and VID2). In this case, the low threshold (the DSS.DISPC_GFX_FIFO_THRESHOLD[11:0] GFXFIFOLOWTHRESHOLD bit field) and the high threshold (DSS.DISPC_GFX_FIFO_THRESHOLD[27:16] GFXFIFOHIGHTHRESHOLD bit field) values must be programmed with a multiplier factor of three (3 x value). By default, the FIFOs are not merged (the DSS.DISPC_CONFIG[14] FIFOMERGE bit reset value is 0).
- Palette/gamma table used (DSS.DISPC_CONFIG[3] PALETTEGAMMATABLE bit): The bit indicates if the palette must be loaded before the graphics data for the following frame. The bit is set by software and reset by hardware.
- Base address of the palette/gamma table buffer in system memory (DSS.DISPC_GFX_TABLE_BA register): The default value of this register at reset time is 0x0. The base address is aligned on a 32-bit address. Depending on the pixel size of graphics data (1, 2, 4, or 8 BPP), 16 (1, 2, or 4 BPP), or 256 (8 BPP) x 32-bit values are loaded from system memory into the internal table memory. To load the table when using the memory as a gamma table, the graphics pipeline is enabled and then disabled by the software when the palette loaded interrupt is generated. The overlay manager ignores the graphics pipe when the table is used as a gamma table.

NOTE: In case of RGB16 format and optimization enabled, the base address is aligned on a 32-bit boundary and the number of bytes to skip is a multiple of 4 bytes.

- Graphics Priority (DSS.DISPC_GFX_ATTRIBUTES[14] GFXARBITRATION): The default value at reset time is 0x0. It is used to change between normal priority (value of 0) to high priority (value of 1) to change priority for the graphics channel vs. video channels. It can be used to give higher priority to the pipelines with real time constraint vs. non real time pipelines. For that is, pipelines associated to the LCD output in RFBI mode should have lower priority than pipelines associated to TV output.
- Graphics Self-Refresh (DSS.DISPC_GFX_ATTRIBUTES[15] GFXSELFREFRESH): The default value at reset time is 0x0. It is used to use the DMA FIFO without accessing the interconnect for multiple frames. Once, the data have been loaded to the DMA FIFO for displaying the frame, they are used for the following frames.

The sequence to activate the self-refresh is the following:

– Frame t: The bit field should be set at anytime during frame

– Frame t+1: Fetch of the data in the DMA FIFO and display of the frame

– Frame t+2: No access to the L3 interconnect, DMA FIFO uses to provide the pixels

The sequence to deactivate the self-refresh is the following:

– Frame t: No access to the L3 interconnect, DMA FIFO uses to provide the pixels, bit field can be changed at any time during the frame

– Frame t+1: Fetch of the data from system memory using the L3 interconnect

12.5.3.2.2 Graphics Layer Configuration Registers

The following registers define the graphics layer configuration:

- DSS.DISPC_CONFIG
- DSS.DISPC_GFX_POSITION
- DSS.DISPC_GFX_SIZE
- DSS.DISPC_GFX_ATTRIBUTES

The graphics layer is enabled/disabled by setting/resetting the DSS.DISPC_GFX_ATTRIBUTES[0] GFXENABLE bit. When the graphics layer is disabled, the graphics window does not exist on the screen and the graphics pipeline and DMA are inactive.

Set a valid configuration before enabling the graphics layer. After a register change, either the DSS.DISPC_CONTROL[6] GODIGITAL or DSS.DISPC_CONTROL[5] GOLCD bit must be set. The software must wait for the hardware to reset the bit before setting it. The software reset is not recommended because the application cannot ensure that the bit is reset before the hardware reset.

12.5.3.2.3 Graphics Window Attributes

The following fields define the attributes of the graphics window:

- Graphics format (DSS.DISPC_GFX_ATTRIBUTES[4:1] GFXFORMAT bit field): The default value of this bit field at reset time is 0x0 (BITMAP 1-BPP). The graphics format can be either: BITMAP1, BITMAP2, BITMAP4, or BITMAP8 (CLUT) or RGB12, RGB16, or RGB24 (true-color formats).
- Graphics window X-position (DSS.DISPC_GFX_POSITION[10:0] GFXPOSX bit field): The default value at reset time is 0x0. The window X-position is from 0 to 2047 columns. All integer values in the range [0:2047] are allowed.
- Graphics window Y-position (DSS.DISPC_GFX_POSITION[26:16] GFXPOSY bit field): The default value of this bit field at reset time is 0x0.The window Y-position is from 0 to 2047 rows. All integer values in the range [0:2047] are allowed.
- Graphics window width (DSS.DISPC_GFX_SIZE[10:0] GFXSIZEX bit field): The default value at reset time is 0x0 (1 pixel). The window width is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed for the following formats: 8 BPP, RGB12, RGB16, and RGB24. The width must be a multiple of eight pixels for 1 BPP, four pixels for 2 BPP, and two pixels for 4 BPP. The maximum bandwidth efficiency for accessing the pixels in system memory is reached when the width of the graphics window (in bytes) is a multiple of the graphics burst size defined in the DSS.DISPC_GFX_ATTRIBUTES[7:6] GFXBURSTSIZE bit field (in bytes).

NOTE: When the RGB24 packed format is selected, the width must be a multiple of 12 bytes when the DSS.DISPC_GFX_ROW_INC register is not 1. When DSS.DISPC_GFX_ROW_INC register is 1, the width can be any size from 1 to 2048 pixels.

The entire pixels of the graphics window must be inside the LCD screen. Depending on the width of the buffer to be displayed in the graphics layer and the position, the width should be adjusted by software to limit the right edge of the window inside the screen.

- Graphics window height (DSS.DISPC_GFX_SIZE[26:16] GFXSIZEY bit field): The default value at reset time is 0x0 (1 pixel). The window height is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed. The entire pixels of the graphics window must be inside the LCD screen.

Depending on the height of the buffer to be displayed in the graphics layer and the position, the height should be adjusted by software to limit the bottom edge of the window inside the screen

- Graphics data endianness (DSS.DISPC_GFX_ATTRIBUTES[10] GFXENDIANNESS bit): This bit indicates the endianness (little or big) of the graphics pixels. The default value at reset time is 0x0 (little endian).
- Graphics data nible mode (DSS.DISPC_GFX_ATTRIBUTES[9] GFXNIBBLEMODE bit): This bit indicates the nibble mode of the graphics pixels. The default value at reset time is 0x0 (Disable).
- Graphics replication logic enable (DSS.DISPC_GFX_ATTRIBUTES[5] GFXREPLICATIONENABLE bit): The default value at reset time is 0x0 (Disable). The encoded pixel data in RGB format (RGB16) can be extended to 24-bit format with or without replication of the MSB part to fill up the LSB due to the 24-bit left alignment. If the replication logic is turned off, the LSB part is filled up with 0s.
- Graphics window skip enable (DSS.DISPC_CONTROL[12] OVERLAYOPTIMIZATION bit): By setting/resetting the bit, the overlay optimization is enabled or disabled. Before enabling the overlay optimization, the DSS.DISPC_GFX_WINDOW_SKIP[31:0] GFXWINDOWSKIP bit field must be set according to the video1 and graphics windows overlap. The default value at reset time is 0x0 (Disable). When video1 is not present, the DSS.DISPC_GFX_WINDOW_SKIP[31:0] GFXWINDOWSKIP bit field should be reset. When the color key is used, the DSS.DISPC_GFX_WINDOW_SKIP[31:0] GFXWINDOWSKIP bit field should be reset.
- Graphics window skip (DSS.DISPC_GFX_WINDOW_SKIP[31:0] GFXWINDOWSKIP bit field): The bit field represents the number of bytes to skip while fetching the graphics-encoded pixels when reaching the beginning of the video window. The optimization allows fetching only the visible graphics pixels. The color key cannot be selected because the graphics pixels under the video window are not present. The default value at reset time is 0x0 (0 byte).

  Figure 12-120 through Figure 12-123 give examples of how to program the GFXWINDOWSKIP field for overlay optimization:

Figure 12-120. Overlay Optimization: Case 1

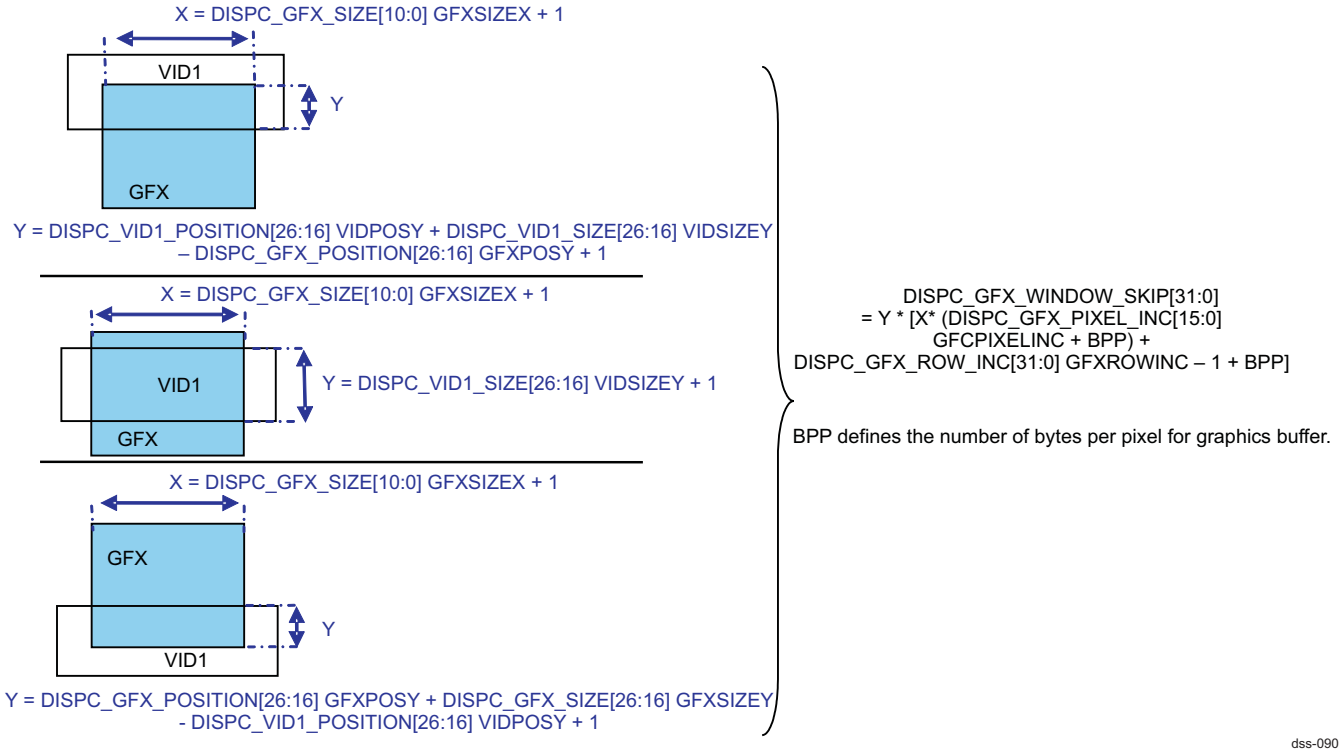
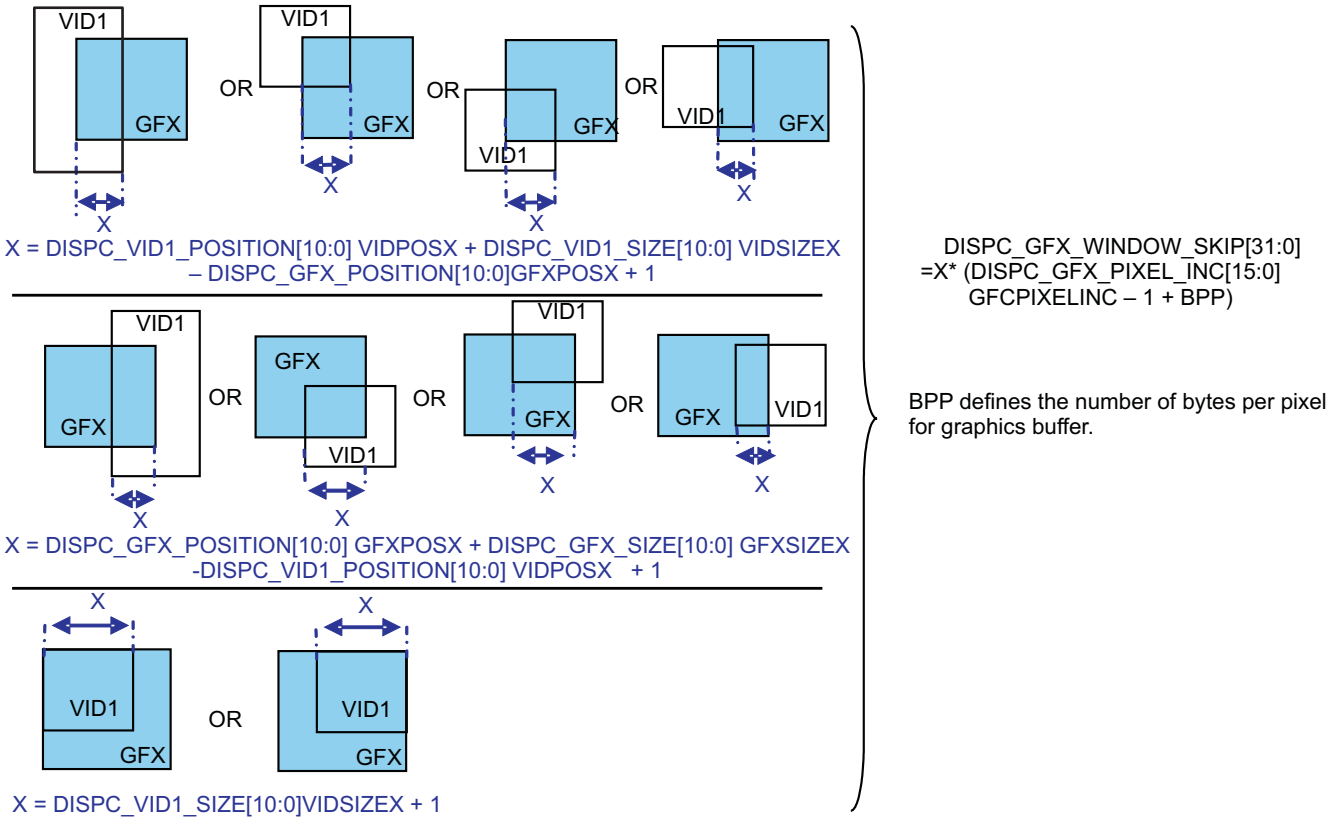
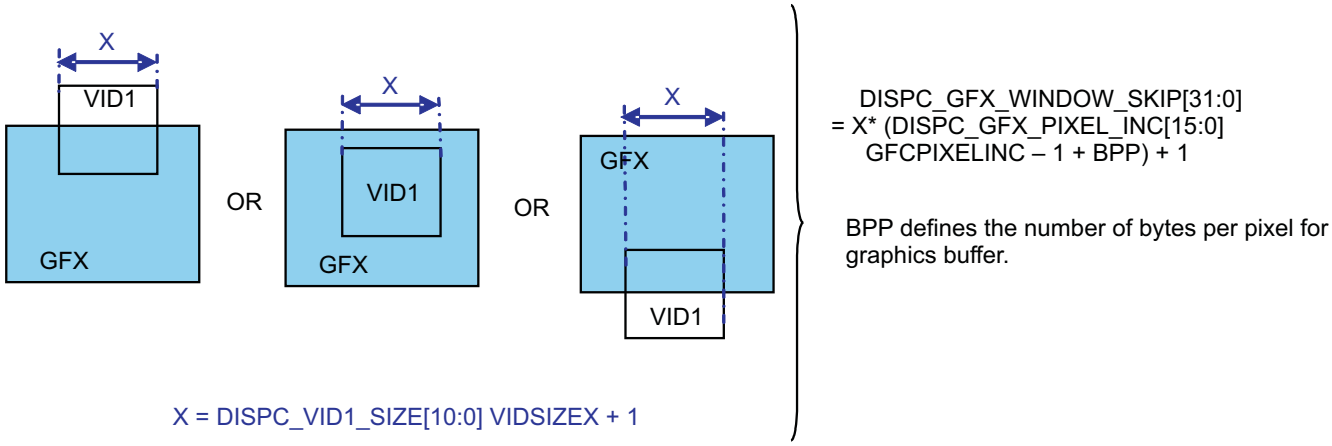


Figure 12-121. Overlay Optimization: Case 2



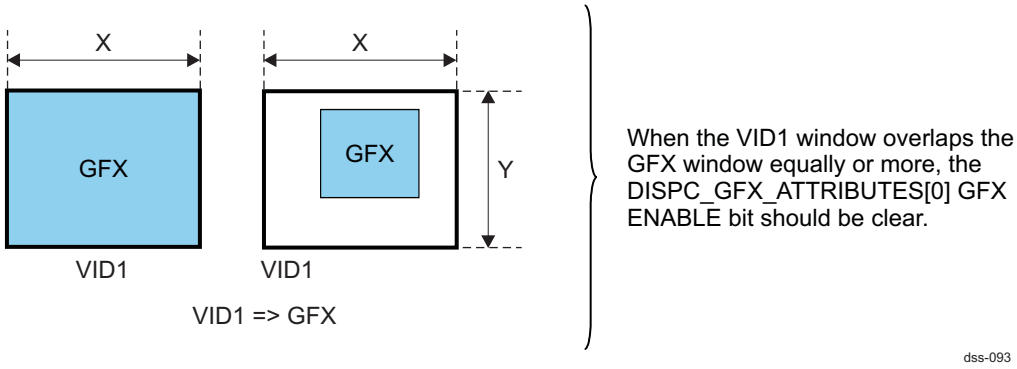
dss-091

Figure 12-122. Overlay Optimization: Case 3



dss-092

Figure 12-123. Overlay Optimization: Case 4



12.5.3.3 Video Layer Configuration

The video layer configuration is common to the LCD and the TV set.

12.5.3.3.1 Video DMA Registers

The following registers define the video DMA engine configuration:

- DSS.DISPC_CONTROL
- DSS.DISPC_VIDn_BA0
- DSS.DISPC_VIDn_ATTRIBUTES
- DSS.DISPC_VIDn_ROW_INC
- DSS.DISPC_VIDn_PIXEL_INC
- DSS.DISPC_VIDn_FIFO_THRESHOLD
- DSS.DISPC_VIDn_PICTURE_SIZE

The following fields define the attributes of the graphics DMA engine:

- Video layer enable (DSS.DISPC_VIDn_ATTRIBUTES[0] VIDENABLE bit): The default value of this bit at reset time is 0x0 (Disabled). The video DMA engine fetches encoded pixels from the system memory only when the video layer is enabled (a valid configuration is programmed for the video layer). The video window is present and the video pipeline is active.
- Burst size (DSS.DISPC_VIDn_ATTRIBUTES[15:14] VIDBURSTSIZE bit field): The default burst size at reset time is 4 x 32 bytes. The possible values are 4 x 32, 8 x 32, and 16 x 32 bytes. The burst size is initialized at boot time by the software and never changes as long as the display controller is enabled. This bit field indicates the maximum burst size for the specific pipeline. In case of misalignment, the DMA engine may issue single and/or smaller burst requests, because the burst size must be aligned to the burst boundary.
- Preload configuration (DSS.DISPC_VIDn_ATTRIBUTES[19] VIDFIFOPRELOAD bit): The default preload configuration uses the DSS.DISPC_VIDn_PRELOAD register value (the reset value is 256 bytes) to define the number of bytes to be fetched from system memory into the display controller graphics FIFO. By programming the DSS.DISPC_VIDn_ATTRIBUTES[19] VIDFIFOPRELOAD bit, software users select between preload register (with 256 bytes as the reset value) and the high threshold value for preload of the encoded pixels. For best performance, the configuration of thresholds is defined using the FIFO size (in bytes) minus 1 for the high threshold, and the FIFO size (in bytes) minus the burst size (in bytes) for the low threshold, which provides 960, 992, and 1008, respectively, for burst sizes 16x32, 8x32, and 4x32. Note also that the preload value is defined based on the following display types:
  - Active matrix (TFT) display: DSS.DISPC_VIDn_PRELOAD[11:0] PRELOAD = 0xB0 (value is 176)
  - Color passive matrix (STN) display: DSS.DISPC_VIDn_PRELOAD[11:0] PRELOAD = 0x110 (value is 272)
  - Monochrome passive matrix (STN) display: DSS.DISPC_VIDn_PRELOAD[11:0] PRELOAD = 0x1B0 (value is 432)

- Base address of the video buffer in system memory (DSS.DISPC_VIDn_BAj registers): The default value at reset time is 0x0. The horizontal resolution is one pixel because the base address is aligned on pixel size boundary. In case of YCbCr 4:2:2 formats, the resolution is 2 pixels. In case of RGB24 packed format, the resolution is 4 pixels. The vertical resolution is one line. The register DSS.DISPC_VIDn_BA0 defines the base address of the even field, and DSS.DISPC_VIDn_BA1 defines the base of the odd field in the case of an external synchronization and based on the value of the input signal DISPC_FID and the polarity. To improve system throughput, the base address should be aligned on the burst size boundary.
- Video FIFO threshold (DSS.DISPC_VIDn_FIFO_THRESHOLD register): The low threshold (DSS.DISPC_VIDn_FIFO_THRESHOLD[11:0] VIDFIFOLOWTHRESHOLD) and the high threshold (DSS.DISPC_VIDn_FIFO_THRESHOLD[27:16] VIDFIFOHIGHTHRESHOLD) values define the FIFO DMA behavior. When the low level is reached, one or more requests are issued to the L3-based interconnect to fill up the FIFO to reach the high threshold. A request is issued as long as the FIFO has enough space available to accept a burst. The DMA engine then waits until the low level is reached to restart the requests. By setting the DSS.DISPC_CONFIG[14] FIFOMERGE bit to 1, users merge the three FIFOs (GFX, VID1, and VID2). In this case, the low threshold (the DSS.DISPC_VIDn_FIFO_THRESHOLD[11:0] VIDFIFOLOWTHRESHOLD bit field with n corresponding to the active video channel 1 or 2) and the high threshold (DSS.DISPC_VIDn_FIFO_THRESHOLD[27:16] VIDFIFOHIGHTHRESHOLD bit field with n corresponding to the active video channel 1 or 2) values must be programmed with a multiplier factor of three (3 x value). By default, the FIFOs are not merged (the DSS.DISPC_CONFIG[14] FIFOMERGE bit reset value is 0).
- Video buffer width (DSS.DISPC_PICTURE_SIZE[10:0] VIDORGSIZEX): The default value at reset time is 0x0 (1 pixel). The buffer width in system memory is from 1 up to 2048 pixels. All the integer values in the range [1:2048] are allowed. Software users must program this bit field to the value minus 1.
- Video buffer height (DSS.DISPC_PICTURE_SIZE[26:16] VIDORGSIZEY): The default value at reset time is 0x0 (1 pixel). The buffer height in system memory is from 1 up to 2048 pixels. All the integer values in the range [1:2048] are allowed. Software users must program this field to the value minus 1.
- Video data endianness (DSS.DISPC_VIDn_ATTRIBUTES[17] VIDENDIANNESS bit, with n=1 or 2): This bit indicates the endianness (little or big) of the video pixels. The default value at reset time is 0x0 (little endian).

12.5.3.3.2 Video Configuration Register

The following shadow registers define video layer n (with n = 1 or 2) configuration:

- DSS.DISPC_CONFIG
- DSS.DISPC_VIDn_POSITION
- DSS.DISPC_VIDn_SIZE
- DSS.DISPC_VIDn_ATTRIBUTES
- DSS.DISPC_VIDn_FIR
- DSS.DISPC_VIDn_PICTURE_SIZE
- DSS.DISPC_VIDn_FIR_COEF_Hi (with i = 0 to 7)
- DSS.DISPC_VIDn_FIR_COEF_HVi (with i = 0 to 7)
- DSS.DISPC_VIDn_CONV_COEFi (with i = 0 to 4)
- The video layer n (with n = 1 or 2) is enabled/disabled by setting/resetting the DSS.DISPC_VIDn_ATTRIBUTES[0] VIDENABLE field. If the video layer is disabled, the video window does not exist on the screen and the whole video pipeline and DMA are inactive. Before enabling the video layer, a valid configuration must be set. After a register change, either the DSS.DISPC_CONTROL[6] GODIGITAL or DSS.DISPC_CONTROL[5] GOLCD bit must be set. The software must wait for the hardware to reset the bit before setting this bit. The software reset is not recommended because the application cannot ensure that the bit is reset before the hardware reset.

12.5.3.3.3 Video Window Attributes

The following fields define the attributes of video window n:

- Video format (DSS.DISPC_VIDn_ATTRIBUTES[4:1] VIDFORMAT bit field, with n = 1 or 2): The default value at reset time is 0x0 (BITMAP 1 BPP, nonsupported format by the video pipeline).The video format can be RGB16, RGB24, YUV2 4:2:2 co-DSS sited, and UYVY 4:2:2 co-sited.
- Video window X-position (DSS.DISPC_VIDn_POSITION[10:0] VIDPOSX bit field, with n = 1 or 2): The default value at reset time is 0x0 (first column starting on the left edge of the screen). The window X-position is from 0 to 2047 columns. All integer values in the range [0:2047] are allowed.
- Video window Y-position (DSS.DISPC_VIDn_POSITION[26:16] VIDPOSY bit field, with n = 1 or 2): The default value at reset time is 0x0 (first row starting at the top of the screen). The window Y-position is from 0 to 2047 rows. All integer values in the range [0:2047] are allowed.
- Video window width (DSS.DISPC_VIDn_SIZE[10:0] VIDSIZEX bit field, with n = 1 or 2): The default value at reset time is 0x0 (1 pixel). The window width is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed. The maximum bandwidth efficiency for accessing the pixels in system memory is reached when the width (in bytes) of the video window is a multiple of the video burst size defined in the DSS.DISPC_VIDn_ATTRIBUTES[15:14] VIDBURSTSIZE bit field (in bytes).

  NOTE: When the RGB24 packed format is selected, the width must be a multiple of 12 bytes when the DSS.DISPC_VIDn_ROW_INC register is not 1. When the DSS.DISPC_VIDn_ROW_INC register is 1, the width can be any size from 1 to 2048 pixels.

  The entire pixels of the video window must be inside the LCD screen. Depending on the width of the buffer to be displayed in the video layer and the position, the width should be adjusted by software to limit the right edge of the window inside the screen.

- Video window height (DSS.DISPC_VIDn_SIZE[26:16] VIDSIZEY bit field, with n = 1 or 2): The default value at reset time is 0x0 (1 pixel). The window height is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed. The entire pixels of the video window must be inside the LCD screen. Depending on the height of the buffer to be displayed in the video layer and the position, the height should be adjusted by software to limit the bottom edge of the window inside the screen.
- Video picture width in system memory (DSS.DISPC_VIDn_PICTURE_SIZE[10:0] VIDORGSIZEX bit field, with n = 1 or 2): The default value at reset time is 0x0 (1 pixel). The window width is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed with RGB16 and RGB24 video data. For YUV2 4:2:2 and UYVY 4:2:2 formats, the width must be a multiple of two pixels. The maximum bandwidth efficiency for accessing the pixels in system memory is reached when the width (in bytes) of the video picture is a multiple of the video burst size defined in the DSS.DISPC_VIDn_ATTRIBUTES[15:14] VIDBURSTSIZE bit field (in bytes).
- Video picture height in system memory (the DSS.DISPC_VIDn_PICTURE_SIZE[26:16] VIDORGSIZEY bit field, with n = 1 or 2): The default value at reset time is 0x0 (1 pixel). The window width is from 1 to 2048 pixels. All integer values in the range [1:2048] are allowed.
- Video Priority (DSS.DISPC_VIDn_ATTRIBUTES[23] VIDARBITRATION): The default value at reset time is 0x0. It is used to change between normal priority (value of 0) to high priority (value of 1) to change priority for the video channel vs. other channels. It can be used to give higher priority to the pipelines with real time constraint vs. non real time pipelines. For that is, pipelines associated to the LCD output in RFBI mode should have lower priority than pipelines associated to TV output.
- Video Self-Refresh (DSS.DISPC_VIDn_ATTRIBUTES[24] VIDSELFREFRESH): The default value at reset time is 0x0. It is used to use the DMA FIFO without accessing the interconnect for multiple frames. Once, the data have been loaded to the DMA FIFO for displaying the frame, they are used for the following frames.
  The sequence to activate the self-refresh is the following:
  - Frame t: The bit field should be set at anytime during frame
  - Frame t+1: Fetch of the data in the DMA FIFO and display of the frame
  - Frame t+2: No access to the L3 interconnect, DMA FIFO uses to provide the pixels

  The sequence to deactivate the self-refresh is the following:
  - Frame t: No access to the L3 interconnect, DMA FIFO uses to provide the pixels, bit field can be changed at any time during the frame
  - Frame t+1: Fetch of the data from system memory using the L3 interconnect

12.5.3.3.4 Video Up-/Down-Sampling Configuration

The video horizontal up-/downsampling block for video pipeline n (with n = 1 or 2) is enabled/disabled by setting/resetting the DSS.DISPC_VIDn_ATTRIBUTES[5] VIDRESIZEENABLE bit.

The video vertical up-/downsampling block for video pipeline n is enabled/disabled by setting/resetting the DSS.DISPC_VIDn_ATTRIBUTES[6] VIDRESIZEENABLE bit.

Set a valid configuration before enabling the video up-/downsampling block.

NOTE: Vertical and horizontal downsampling are limited to a 1/4 resize factor.

After a register change, either the DSS.DISPC_CONTROL[6] GODIGITAL or DSS.DISPC_CONTROL[5] GOLCD bit must be set. The software must wait until the hardware resets this bit before setting it. The software reset is not recommended because the application cannot ensure that the bit is reset before the hardware reset.

The following fields define the configuration of the video up-/downsampling block for video pipeline n:

- Vertical up-/downsampling increment value (DSS.DISPC_VIDn_FIR[27:16] VIDFIRVINC bit field, with n = 1 or 2): The unsigned integer value range is [1:4096]. The software calculates the value using the following equation:

$$VIDFIRVINC[12:0] = 1024 \times \left(\frac{VIDORGSIZEY[10:0]}{VIDSIZEY[10:0]} \right) \quad \text{dss-E093} \tag{5}$$

NOTE:

- If the VIDFIRVINC[11:0] bit field value is greater than 4096, it is clipped to 4096. If VIDSIZEY[10:0] equals 0x1, VIDSIZEY[10:0] is replaced by 0x2 in the previous equation.
- The VIDORGSIZEY[10:0] and VIDSIZEY[10:0] bit field values must be programmed with the value desired minus 1.

- Horizontal up-/downsampling increment value (DSS.DISPC_VIDn_FIR[11:0] VIDFIRHINC bit field, with n = 1 or 2): The unsigned integer value range is [1:4096]. The software calculates the value using the following equation:

$$VIDFIRHINC[12:0] = 1024 \times \left(\frac{VIDORGSIZEX[10:0]}{VIDSIZEX[10:0]} \right) \quad \text{dss-E094} \tag{6}$$

NOTE:

- If the VIDFIRHINC[11:0] bit field value is greater than 4096, it is clipped to 4096. If VIDSIZEX[10:0] equals 1, VIDSIZEX[10:0] is replaced by 2 in the previous equation.
- The VIDORGSIZEX[10:0] and VIDSIZEX[10:0] bit field values must be programmed with the value desired minus 1.

- Vertical up-/downsampling accumulator value (DSS.DISPC_VIDn_ACCUI[25:16] VIDVERTICALACCU bit field): The unsigned integer value range is [0:1023]. The accumulator value indicates in which phase the vertical filtering starts. The value 0 indicates that 0 is the first phase used by the hardware to generate the first data (see Table 12-46).
- Vertical up-/downsampling line buffer configuration (DSS.DISPC_VIDn_ATTRIBUTES[22] VIDLINEBUFFERSPLIT bit): The default value at reset time is 0x0 (line buffers are not split). The backward compatibility is maintained versus OMAP2420 and OMAP2430 devices. When the bit field is set, each line buffer is split into two line buffers to be able to use six line buffers instead of three.
- Vertical up-/downsampling line buffer configuration (DSS.DISPC_VIDn_ATTRIBUTES[21] VIDVERTICALTAPS bit): The default value at reset time is 0x0 (3-tap configuration is used). If the bit field is reset, the 3-tap configuration is used. The backward compatibility is maintained versus OMAP2420 and OMAP2430 devices. When the bit field is set, the 5-tap configuration is used and the DSS.DISPC_VIDn_ATTRIBUTES[22] VIDLINEBUFFERSPLIT bit must be set to 1.

- Vertical up-/downsampling line buffer configuration (DSS.DISPC_VIDn_ATTRIBUTES[20] VIDDMAOPTIMIZATION bit): The default value at reset time is 0x0 (no optimization). If the bit is set, the DMA engine fetches two pixels for each 32-bit OCP request (RGB16 and YUV422) while doing 90- and 270-degree rotation. If the bit is clear, the DMA engine fetches one pixel for each 32-bit OCP request (RGB16 and YUV422) while doing 90- and 270-degree rotation. The width and height of picture should be even to use the optimization.

NOTE: If the 5-tap resizer is used for RGB16 and YUV422 picture formats, the width of the input picture must be a multiple of 2 pixels and more than 5 pixels. This leads to the following register configuration:

DISPC_VIDn_ATTRIBUTES[21] VIDVERTICALTAPS == 1

DISPC_VIDn_PICTURE_SIZE[10:0] VIDORGSIZEX > 4 and even

- Horizontal up-/downsampling accumulator value (DSS.DISPC_VIDn_ACCUI[9:0] VIDHORIZONTALACCU bit field): The unsigned integer value range is [0:1023]. The accumulator value indicates in which phase the horizontal filtering starts. The value 0 indicates that 0 is the first phase used by the hardware to generate the first data (see Table 12-46).

Table 12-46. Vertical/Horizontal Accumulator Phase

Accumulator Value	Phases f
0	0
128	1
256	2
384	3
512	4
640	5
768	6
896	7

- Vertical up-/downsampling coefficients (DSS.DISPC_VIDn_FIR_COEF_HVi registers, with n = 1 or 2, i = 0 to 7): The 3-tap vertical up-/downsampling coefficients are defined in these registers. There are eight registers for the eight phases with three coefficients for each, or a total of 24 programmable coefficients for the vertical up-/downsampling block. Each register contains two 8-bit signed coefficients and one 8-bit unsigned coefficient (the central one).

  In addition, there are 2-tap vertical up/downsampling coefficients defined in DSS.DISPC_VIDn_FIR_COEF_Vi registers. There are 8 registers for the 8 phases with 2 coefficients for each of them so a total of 16 programmable coefficients for the vertical up-/downsampling block used in addition of the 3-tap registers defined above. Each register contains two 8-bit signed coefficients. In case of 5-tap configuration, both sets of registers DSS.DISPC_VIDn_FIR_COEF_HVi and DSS.DISPC_VIDn_FIR_COEF_Vi are used. In case of 3-tap configuration, only one set of registers DSS.DISPC_VIDn_FIR_COEF_HVi is used.
- Horizontal up-/downsampling coefficients (DSS.DISPC_VIDn_FIR_COEF_Hi and DISPC_VIDn_FIR_COEF_HVi registers, with n = 1 or 2, i = 0 to 7): The DSS.DISPC_VIDn_FIR_COEF_Hi register and the DSS.DISPC_VIDn_FIR_COEF_HVi register define the 5-tap horizontal up-/downsampling coefficients. There are eight registers for the eight phases with five coefficients for each register, or a total of 40 programmable coefficients for the horizontal up-/downsampling block.

  Each DSS.DISPC_VIDn_FIR_COEF_Hi register contains three 8-bit signed coefficients and one 8-bit unsigned coefficient (the central one). Each DSS.DISPC_VIDn_FIR_COEF_HVi register contains one 8-bit signed coefficient.

  The programmable coefficient for the FIR up-/downsampling method must be adjusted based on application needs. For more details on scaling programming settings, see Section 12.6.1.

12.5.3.3.5 Video Color Space Conversion Configuration

The DSS.DISPC_VIDn_CONV_COEFi registers (with i = 0 to 4) has nine 11-bit coefficients defined for the programmable color space conversion block for video pipeline n (with n = 1 or 2).

The standard register coefficients are:

YCbCr-to-RGB Registers (VidFullRange=0)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y - 16 \\ Cr - 128 \\ Cb - 128 \end{bmatrix}$$

dssE095 (7)

YCbCr to RGB Registers (VidFullRange=1)

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y \\ Cr - 128 \\ Cb - 128 \end{bmatrix}$$

dss-E096 (8)

Table 12-47 lists the color space conversion register values.

Table 12-47. Color Space Conversion Register Values

Coefficients	BT.601-5	BT.601-5 Range [0:255]	BT.709	BT.709 Range [0:255]
RY	298	256	298	256
RCr	409	351	459	394
RCb	0	0	0	0
GY	298	256	298	256
GCr	–208	–179	–137	–118
GCb	–100	–86	–55	–47
BY	298	256	298	256
BCr	0	0	0	0
BCb	517	443	541	465
VidFullRange	0	1	0	1

12.5.3.4 Rotation/Mirroring Display Subsystem Settings

This section describes rotation/mirroring settings. The device provides flexible mechanisms for an efficient implementation of rotation using the display-subsystem, its DMA engine, and the rotation engine of the SMS module. Depending on whether the image data is located in on-chip SRAM or external SDRAM, either a DMA rotation or a VRFB rotation is used. When configuring the rotation, the image data format (RGB or YUV) must also be taken into account. The video pipelines also perform the interpolation of YUV image data and the color conversion from YUV into RGB format for displaying the images on an LCD screen.

12.5.3.4.1 Image Data Formats

To understand the programming of the rotation mechanisms described underneath, the supported representations of the image data in memory must also be considered. Differences exist between the supported formats on the graphics and video pipelines. The graphics pipeline supports RGB and RGB with a color look-up table (CLUT), whereas the video pipelines support two versions of YUV 4:2:2 and RGB16. In the case of YUV, the interpolation and color conversion hardware in the video pipelines converts the image data to the RGB format suitable for the LCD screen.

The graphics pipeline supports:

- 1-, 2- ,4-, and 8-bits-per-pixel color look-up table
- 12-, 16-, and 24-bits-per-pixel RGB

The video pipelines support the following data formats (always 2 bytes/pixel):

- RGB16
- YUV2
- UYVY

For more information on the graphics data formats, please refer to Section 12.4.2.2, *Graphics Pipeline*.

For more information on the video data formats, please refer to Section 12.4.2.3, *Video Pipeline*.

In the video pipeline, the YUV 4:2:2 format is converted into a full YUV 4:4:4 format by interpolation of the chrominance values from neighboring pixels. After this interpolation is completed, the conversion to the RGB format (suitable for displaying the image on the LCD screen) is performed.

For more information on YUV 4:2:2 to RGB conversion, please refer to Section 12.4.2.3.2, *Color Space Conversion*.

12.5.3.4.2 Image Data from On-Chip SRAM

For image data located in the on-chip SRAM, the DSS DMA is used to perform 90-degree, 180-degree, and 270-degree rotation. This is done by using the double-indexed addressing mode of the DMA. This addressing mode allows a pixel and a row increment to be specified. These address increments are used after each pixel or each row (line), respectively. Figure 12-124 illustrates the principle steps for a 90-degree rotation.

Figure 12-124. 90° DMA Rotation Example

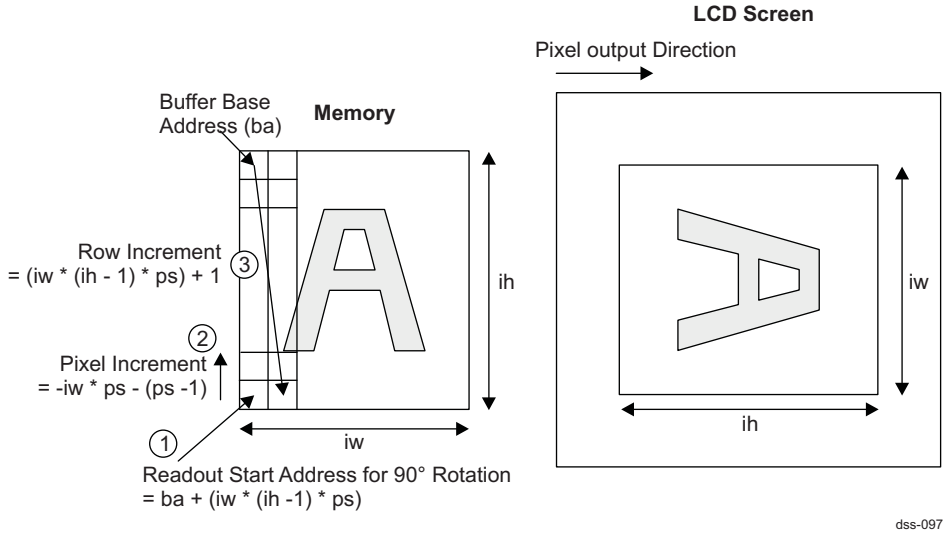


Table 12-48. 90-degree DMA Rotation Example Description

Parameter	Description	Additional parameters for formulas
ba	Buffer Base Address in Memory	
IW	Image Width in pixels	iw = IW-1
IH	Image Height in pixels	ih = IH-1
ps	Pixel Size (in bytes)	

Figure 12-124 shows how the image is stored in memory and how it is read out to achieve a 90-degree rotated orientation. The first pixel for the 0-degree orientation is located at the buffer base address (ba). If the image is to be shown on an LCD screen with a 90-degree rotation, the readout starts at the 90-degree base address (1). To proceed from one pixel to the next in the same line in the rotated orientation, the pixel increment (2) must be applied. At the end of each line of the rotated view, the row increment (3) is the offset to advance to the beginning of the next line in the memory buffer.

Hence, by setting the three DMA parameters (base address, pixel increment, and row increment), a 90-degree rotation can be achieved, as can 180-degree and 270-degree rotation. Each of the parameters can also be combined with an optional mirroring on the vertical axis.

Following there is a description the setup required to perform the rotation via the DSS DMA. This rotation mechanism is used when the image data is stored in internal SRAM.

12.5.3.4.2.1 Rotation/Mirroring Registers

To set up the rotation and/or mirroring, the following registers must be programmed:

- Graphics pipeline (GFX):
  - DSS.DISPC_GFX_BAj
  - DSS.DISPC_GFX_PIXEL_INC
  - DSS.DISPC_GFX_ROW_INC
  - DSS.DISPC_GFX_ATTRIBUTES
  - DSS.DISPC_GFX_POSITION
  - DSS.DISPC_GFX_SIZE
  - DSS.DISPC_GFX_FIFO_THRESHOLD
- Video pipelines (VID) 1 and 2:
  - DSS.DISPC_VIDn_BAj
  - DSS.DISPC_VIDn_PIXEL_INC
  - DSS.DISPC_VIDn_ROW_INC
  - DSS.DISPC_VIDn_ATTRIBUTES
  - DSS.DISPC_VIDn_POSITION
  - DSS.DISPC_VIDn_SIZE
  - DSS.DISPC_VIDn_CONV_COEF0 to DSS.DISPC_VIDn_CONV_COEF4
  - DSS.DISPC_VIDn_FIFO_THRESHOLD
- DSS.DISPC_xxx_BAj: These registers contain the base address of the image data at which the DSS DMA transfer of the image starts. The register values depend on the rotation chosen (see Table 12-49 for the formulas). When using the LCD interface, the registers DISPC_xxx_BA0 are used. The registers DISPC_xxx_BA1 are only used for the TV output.
- DSS.DISPC_xxx_PIXEL_INC[15:0]: This bit field contains the DMA addressing increment after each pixel. This bit field is used to perform the DSS DMA rotation (see Table 12-49 for the formula).

NOTE: When the RGB24 packet format is selected, the only valid value is 1.

- DSS.DISPC_xxx_ROW_INC[31:0]: This bit field contains the DMA addressing increment after each row (line) of pixels. This bit field is used to perform the DSS DMA rotation (see Table 12-49 for the formula).

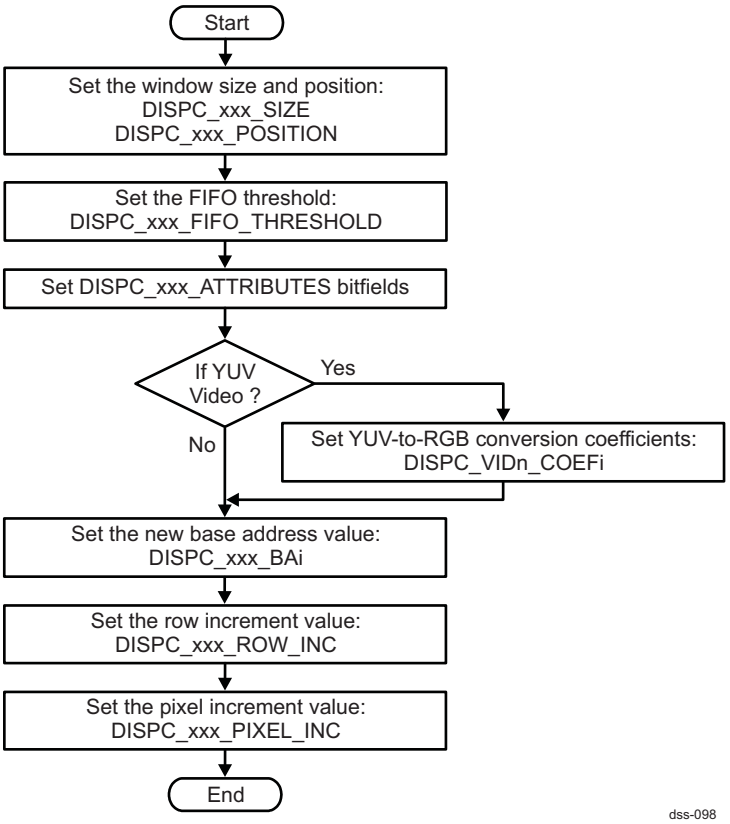
NOTE: When the RGB24 packet format is selected, the valid values are 1 and any value multiples of 12 bytes (4x32 bit). When the value is a multiple of 12 bytes, the width must be a multiple of 12 bytes. When the value is 1, the width can be any size from 1 to 2048 pixels.

- DSS.DISPC_xxx_ATTRIBUTES: These registers contain the main settings for the pipeline, such as the image data format, the rotation value, and the enable bit for the pipeline. The bit fields of these registers play a role in the rotation and in the image data format setup.
  - The following bit fields are used by the graphics pipeline to set up the image format:
    - GFXENABLE: Set this field to activate the hardware path in use.
    - GFXFORMAT: Use this field to specify the format of the graphic frame.
    - GFXROTATION: Set this field to the value corresponding to the rotation angle desired only if the frame contains RGB24 pixel data; otherwise, set it to 0x0 regardless of the degree of rotation.
    - GFXREPLICATIONENABLE: Use this bit to determine whether the encoded pixel data in RGB formats (RGB12 and RGB16) is extended to 24-bit format with or without replication of the MSB to fill the LSBs of each color component. If the replication logic is turned off, the LSB parts are filled with 0s. It is recommended to always enable this feature.
    - GFXCHANNELOUT: Set this field based on whether the frame is to be rendered on the LCD or on the TV set.
  - The following bit fields for the two video pipelines:
    - VIDENABLE: Set this field to activate the hardware path in use.
    - VIDFORMAT: Use this field to specify the format of the video frame (RGB16 or YUV422).
    - VIDCOLORCONVENABLE: If the video is in YUV422 format, set this field to enabled.
    - VIDROTATION: Set this field to the value corresponding to the rotation angle desired only if the frame contains non-RGB pixel data; otherwise, set it to 0x0 regardless of the degree of rotation. See Section 12.5.3.4.4 for more information.
    - VIDROWREPEATENABLE: Set this field to enabled only if the frame contains YUV pixel data and the rotation is 90-degree or 270-degree so that the row pixel data are fetched twice to extract both Y components. See Section 12.5.3.4.4 for more information.
    - VIDCHANNELOUT: Set this field based on whether the frame is to be rendered on the LCD or on the TV set.
- DSS.DISPC_xxx_POSITION: Use this register to configure the position of the window.
- DSS.DISPC_xxx_SIZE: Use this register to configure the size of the window.
- DSS.DISPC_VIDn_CONV_COEF0, DSS.DISPC_VIDn_CONV_COEF1, DSS.DISPC_VIDn_CONV_COEF2, DSS.DISPC_VIDn_CONV_COEF3, and DSS.DISPC_VIDn_CONV_COEF4: These registers contain the conversion coefficients required for YUV-to-RGB color conversion.
- DSS.DISPC_xxx_FIFO_THRESHOLD: Set the low threshold (DSS.DISPC_GFX_FIFO_THRESHOLD[11:0] GFXFIFOLOWTHRESHOLD) and the high threshold (DSS.DISPC_GFX_FIFO_THRESHOLD[27:16] GFXFIFOHIGHTHRESHOLD) values to define the FIFO DMA behavior. When the low level is reached, one or more requests are issued to the L3-based interconnect to fill up the FIFO to reach the high threshold. The DMA engine then waits until the low level is reached to restart the requests.

12.5.3.4.2.2 DMA Register Settings

To configure the display controller for rotation and/or mirroring, use the following settings:

Figure 12-125. Rotation/Mirroring Settings



NOTE: These registers are shadow registers. To take into account the new values, software users must set the DSS.DISPC_CONTROL[5] GOLCD bit to 1.

Table 12-49 details the base address, the row and pixel increment values to access the buffer in memory (contiguous pixels) except for the RGB24 packet format. Table 12-50 lists the rotation register settings for RGB24 packet format (only for the two video pipelines).

Table 12-49. DMA Rotation Register Settings

Rotation	Registers	GFX/VIDx (1)
0 degree	DSS.DISPC_xxx_BAj	ba
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	1
90 degrees	DSS.DISPC_xxx_BAj	ba + (iw x (ih-1) x ps)
	DSS.DISPC_xxx_PIXEL_INC	-(iw x ps) - 1
	DSS.DISPC_xxx_ROW_INC	(iw x (ih-1)) x ps + 1
180 degrees	DSS.DISPC_xxx_BAj	ba + (iw x ih-1) x ps
	DSS.DISPC_xxx_PIXEL_INC	-2 x ps
	DSS.DISPC_xxx_ROW_INC	-2 x ps

(1)
- ba = start address of image buffer in memory
- iw = image width in pixels per row - 1 (for YUV: pixels per row divided by 2)
- ih = image height - 1 (number of rows)
- ps = pixel size in bytes (RGB: 2 bytes per pixel, YUV: 4 bytes per pixel)

See Table 12-48 for more information of these parameters.

Table 12-49. DMA Rotation Register Settings (continued)

Rotation	Registers	GFX/VIDx [1]
270 degrees	DSS.DISPC_xxx_BAj	ba + (iw -1) x ps
	DSS.DISPC_xxx_PIXEL_INC	(iw -1) x ps + 1
	DSS.DISPC_xxx_ROW_INC	-(iw x (ih-1)) x ps - ps + 1

NOTE: In case of RGB16 format and optimization enabled, the base address is aligned on a 32-bit boundary and the number of bytes to skip is a multiple of 4 bytes.

Table 12-50. Video Rotation Register Settings (With RGB24 Packet Format)

Rotation	Registers (with n = 1 or 2)	SDRAM Direct Access [1]
0 degree	DSS.DISPC_VIDn_BAj	ba
	DSS.DISPC_VIDn_PIXEL_INC	1
	DSS.DISPC_VIDn_ROW_INC	1
180 degrees	DSS.DISPC_VIDn_BAj	ba+(iw*ih*3)-4
	DSS.DISPC_VIDn_PIXEL_INC	-7
	DSS.DISPC_VIDn_ROW_INC	-7

(1)

- ba = physical base address of the buffer in the system memory (top-left corner of the picture)
- iw = number of pixels per row - 1 (original picture in system memory) for BITMAP and RGB formats and number of pixels per row divided by 2 for YUB422 formats
- h = number of lines - 1 (original picture in system memory)
- ps = pixel size in bytes (BITMAP8: 1 byte; RGB16: 2 bytes; YUV422: 4 bytes)

See Table 12-48 for more information of these parameters.

The DMA rotation described above can be also combined with an optional mirroring on the vertical axis (see Figure 12-126). The only settings that must be changed to achieve this mirroring are the registers described above: DISPC_xxx_BAj, DISPC_xxx_PIXEL_INC, and DISPC_xxx_ROW_INC. Table 12-51 provides the DMA setup formulas for rotation with mirroring to access the buffer in memory (contiguous pixels) except for the RGB24 packet format.

Figure 12-126. 90° Rotation With Mirroring

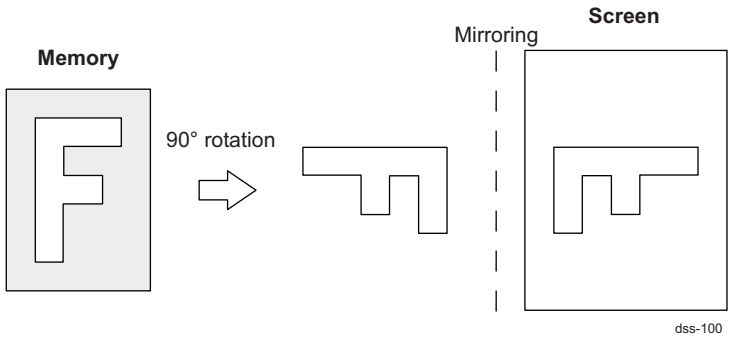


Table 12-51. Register Settings for DMA Rotation With Mirroring

Rotation	Registers	GFX/VIDx [1]
0 degree	DSS.DISPC_xxx_BAj	ba + (iw-1) x ps
	DSS.DISPC_xxx_PIXEL_INC	-2 x ps + 1
	DSS.DISPC_xxx_ROW_INC	2 x (iw-1) x ps + 1
90 degrees	DSS.DISPC_xxx_BAj	ba
	DSS.DISPC_xxx_PIXEL_INC	(iw-1) x ps
	DSS.DISPC_xxx_ROW_INC	(-iw x (ih-1)) x ps
180 degrees	DSS.DISPC_xxx_BAj	ba + iw x (ih-1) x ps
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	-2 x iw x ps
270 degrees	DSS.DISPC_xxx_BAj	ba + (iw x ih - 1) x ps
	DSS.DISPC_xxx_PIXEL_INC	(iw -1) x ps + 1
	DSS.DISPC_xxx_ROW_INC	- (iw x (ih-1)) x ps - ps + 1

(1)
- ba = start address of image buffer in memory
- iw = image width in pixels per row - 1 (for YUV: pixels per row divided by 2)
- ih = image height - 1 (number of rows)
- ps = pixel size in bytes (RGB: 2 bytes per pixel, YUV: 4 bytes per pixel)

See Table 12-48 for more information of these parameters.

12.5.3.4.3 Image Data from External SRAM

The device offers a rotation engine in the SMS called the VRFB. This rotation method must be used for maximum efficiency when the image data is located in SDRAM. To use the VRFB rotation, both the SMS module and the DSS DMA must be configured.

In the DSS, the same registers must be set up as described for DMA rotation (see Section 12.5.3.4.2, *Image Data from On-Chip SRAM*). The differences are in the setup of the following registers:

- DISPC_xxx_ROW_INC: This field contains the DMA addressing increment after each row (line) of pixels. This field is used to add the remaining delta at the end of each line, to reach the 2048-pixel line size of the VRFB (see Table 12-52 for the formula).
- DISPC_xxx_PIXEL_INC: This field contains the DMA addressing increment after each pixel. For VRFB rotation, this value is set always to 1 (see Table 12-52).

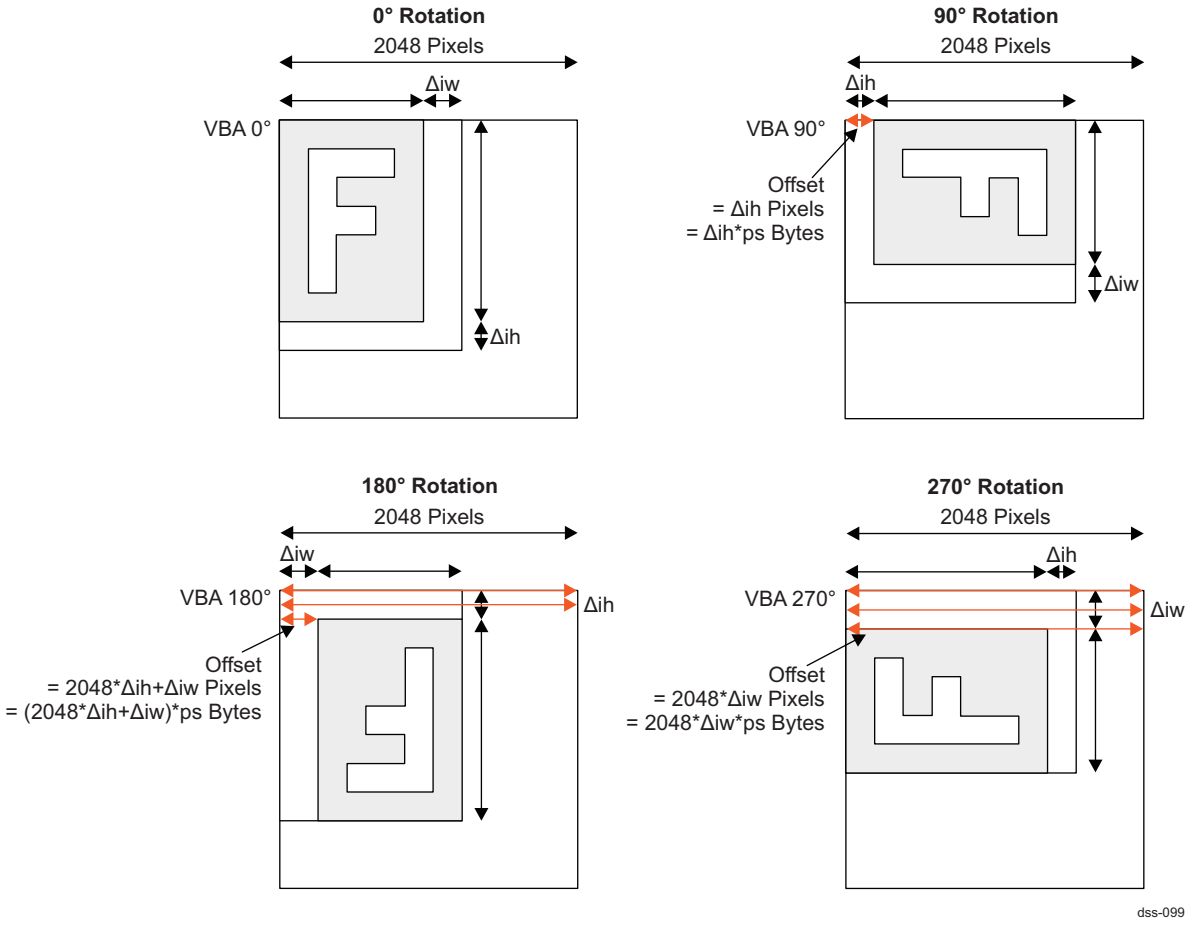
Table 12-52. VRFB Rotation - DMA Settings

Rotation	Registers	GFX/VIDx [1]
0 degree	DSS.DISPC_xxx_BAj	VBA0
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	(2048 - iw) x ps + 1
90 degrees	DSS.DISPC_xxx_BAj	VBA90 + offset
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	(2048 - ih) x ps + 1
180 degrees	DSS.DISPC_xxx_BAj	VBA180 + offset
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	(2048 - iw) x ps + 1
270 degrees	DSS.DISPC_xxx_BAj	VBA270 + offset
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	(2048 - ih) x ps + 1

(1)
- VBAx = virtual address of the chosen VRFB context and orientation
- iw = image width (width in pixels for RGB, width in pixels divided by 2 for YUV)
- ih = image height
- ps = pixel size in bytes (2 bytes per pixel for RGB, 4 bytes per pixel for YUV)
- Offset = see below

Figure 12-127 provides the offset values that must be added to the virtual base addresses for 90-degree, 180-degree, and 270-degree rotation. This offset is applicable only when the defined image size in the VRFB module is greater than the actual image size, because it must be a multiple of the page width and height. In the example discussed above, the image height was set to 256 lines, instead of 240, because the page height was 32 lines. This offset must be added to the virtual base addresses, because the VRFB module is not aware of the actual image size. Figure 12-127 illustrates why this occurs and how the offset is calculated.

Figure 12-127. Offset for VRFB Rotation



Δiw = Image width delta between the actual image width and the programmed image width because of the page width

Δih = Image height delta between the actual image height and the programmed image height because of the page height:

- Offset 90-degree: Δih pixels = Δih x ps bytes
- Offset 180-degree: 2048 x Δih+ Δiw pixels = 2048 x Δih x ps+ Δiw x ps bytes
- Offset 270-degree: 2048 x Δiw pixels = 2048 x Δiw x ps bytes

In the example given above, the delta in the image height is 256 – 240 = 16 lines (Δih = 16), whereas the exact value of the width can be programmed (Δiw = 0). In that case, the resulting offset values are:

- YUV:
  - Offset 90-degree: 16 x 4 bytes
  - Offset 180-degree: 2048 x 16 x 4 bytes
  - Offset 270-degree: 0 bytes
- RGB:
  - Offset 90-degree: 16 x 2 bytes

- Offset 180-degree: 2048 x 16 x 2 bytes
- Offset 270-degree: 0 bytes

As with DMA rotation, mirroring along the vertical axis can be added on top of the rotation when using the VRFB. This mirroring is achieved by combining an appropriate VRFB rotation with a readout of the VRFB by the DSS DMA, going backwards from the last line to the first line of the rotated image. Table 12-53 provides the DMA settings for VRFB rotation with mirroring.

Table 12-53. VRFB Rotation With Mirroring - DMA Settings

Rotation	Registers	GFX/VIDx (1)
0 degree	DSS.DISPC_xxx_BAj	VBA180 + 2048 x (ih - 1) x ps + offset
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	-(2048 + iw) x ps + 1
90 degrees	DSS.DISPC_xxx_BAj	VBA270 + 2048 x (iw - 1) x ps + offset
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	-(2048 + ih) x ps + 1
180 degrees	DSS.DISPC_xxx_BAj	VBA0 + 2048 x (ih - 1) x ps
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	-(2048 + iw) x ps + 1
270 degrees	DSS.DISPC_xxx_BAj	VBA90 + 2048 x (iw - 1) x ps + offset
	DSS.DISPC_xxx_PIXEL_INC	1
	DSS.DISPC_xxx_ROW_INC	-(2048 + ih) x ps + 1

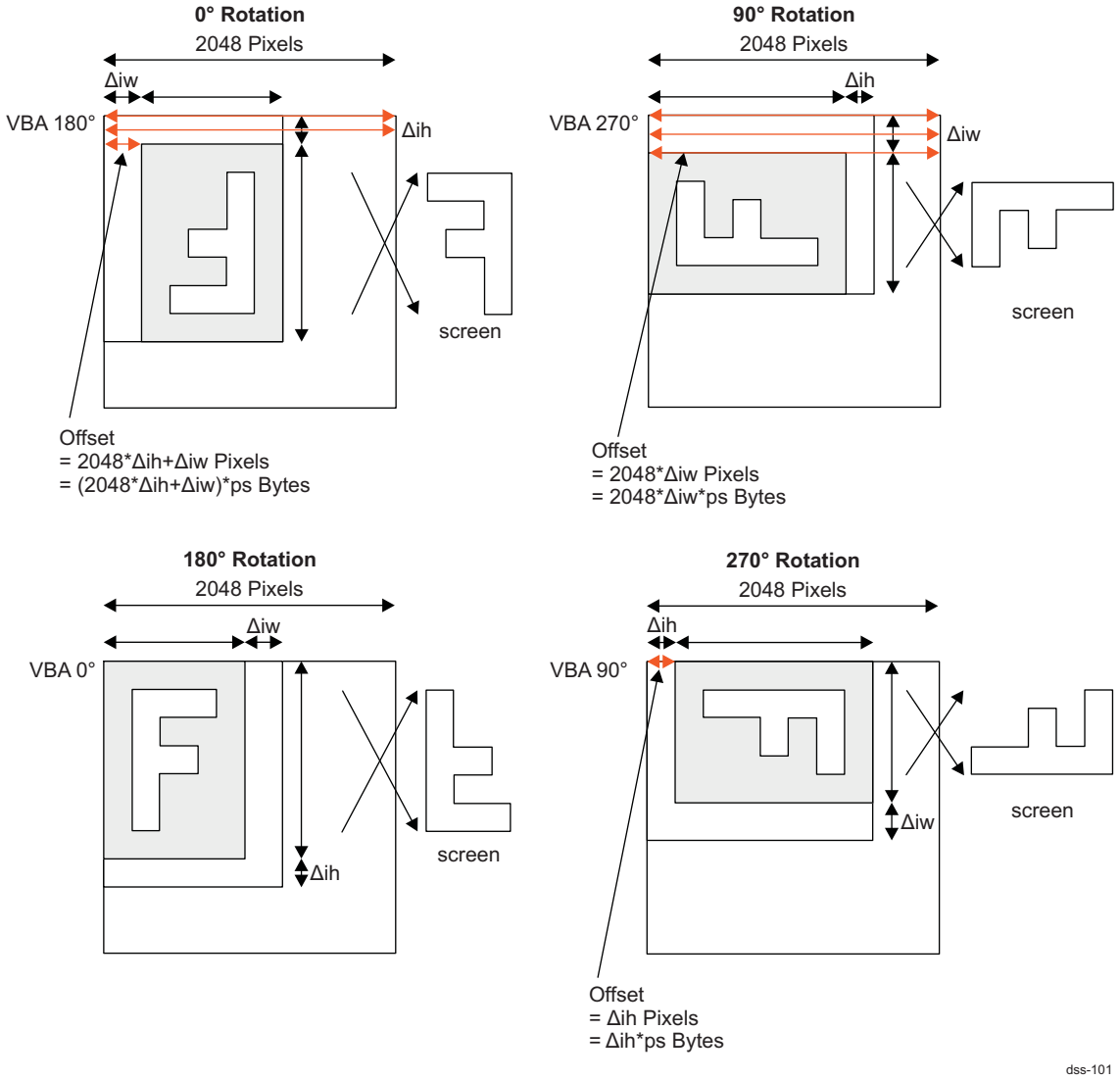
(1)

- VBAx = virtual address of the chosen VRFB context and orientation
- iw = image width (width in pixels for RGB, width in pixels divided by 2 for YUV)
- ih = image height
- ps = pixel size in bytes (2 bytes per pixel for RGB, 4 bytes per pixel for YUV)
- Offset = see below

Some offsets are required if there is a delta between the programmed image size in the VRFB and the actual image size. As shown in Figure 12-128, this applies to 0-degree rotation with mirroring (using VBA 180°), 90° rotation with mirroring (VBA 270-degree), and 270-degree rotation with mirroring (VBA 90-degree). The offsets are calculated in this manner:

- Offset 0-degree (mirroring): 2048 x Δih + Δiw pixels = 2048 x Δih x ps + Δiw x ps bytes
- Offset 90-degree (mirroring): 2048 x Δiw pixels = 2048 x Δiw x ps bytes
- Offset 270-degree (mirroring): Δih pixels = Δih x ps bytes

Figure 12-128. Offset for VRFB Rotation With Mirroring



12.5.3.4.4 Additional configuration when using YUV format

The rotation flag (DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION bit , with n = 1 or 2) and the repeat flag (DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE) indicate the rotation to apply to the video-encoded pixels from the SDRAM and SRAM. The 2D addressing mode is used, but even when accessing the SDRAM buffer, some post-processing must be performed on the YUV 4:2:2 data depending on the rotation. These bits are set only when the video format is not RGB; otherwise, the bit field is reset to 0. Table 12-54 and Table 12-55 describe the configuration of these registers.

Table 12-54. Video Rotation Register Settings (YUV Only)

Rotation	Registers (with n = 1 or 2)	SDRAM + Rotation engine
0 degree	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x0
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0
90 degrees	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x1
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x1
180 degrees	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x2
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0

Table 12-54. Video Rotation Register Settings (YUV Only) (continued)

Rotation	Registers (with n = 1 or 2)	SDRAM + Rotation engine
270 degree	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x3
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x1

Table 12-55. Video Rotation With Mirroring Register Settings (YUV only)

Rotation	Registers (with n = 1 or 2)	SDRAM + Rotation engine
0 degree	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x2
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0
90 degrees	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x1
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x1
180 degrees	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x0
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0
270 degree	DSS.DISPC_VIDn_ATTRIBUTES[13] VIDROTATION	0x3
	DSS.DISPC_VIDn_ATTRIBUTES[18] VIDROWREPEATENABLE	0x1

NOTE: For YUV 4:2:2 video-encoded pixels, the hardware must always fetch a 32-bit value from the system memory to generate a YUV 4:4:4 value before YUV-to-RGB conversion.

- For 90- and 270-degree rotation, the missing chrominance samples for the odd pixels are generated by duplicating the chrominance samples of the previous even pixels.
- For 0- and 180-degree rotation, the missing chrominance samples for the odd pixels are generated by averaging the contiguous chrominance samples.

12.5.3.5 LCD-Specific Control Registers

The following registers define the LCD output configuration:

- DSS.DISPC_CONTROL
- DSS.DISPC_CONFIG
- DSS.DISPC_DEFAULT_COLOR_m (m=0)
- DSS.DISPC_TRANS_COLOR_m (m=0)
- DSS.DISPC_TIMING_H
- DSS.DISPC_TIMING_V
- DSS.DISPC_POL_FREQ
- DSS.DISPC_DIVISOR
- DSS.DISPC_SIZE_LCD
- DSS.DISPC_DATA_CYCLEk
- DSS.DISPC_CPR_COEF_R, DSS.DISPC_CPR_COEF_G, DSS.DISPC_CPR_COEF_B

Setting/resetting the DSS.DISPC_CONTROL[0] LCDENABLE bit enables/disables the LCD output. A valid configuration must be set before enabling the LCD output.

12.5.3.5.1 LCD Attributes

The following fields define the attributes of the panel connected to the display controller:

- Monochrome or color panel (the DSS.DISPC_CONTROL[2]MONOCOLOR bit)
- Passive Matrix or active Matrix panel (the DSS.DISPC_CONTROL[3] STNTFT bit)
- Color depth (the DSS.DISPC_CONTROL[9:8] TFTDATALINES bit field)

- Number of lines per panel (the DSS.DISPC_SIZE_LCD[26:16] LPP bit field)
- Number of pixels per line (the DSS.DISPC_SIZE_LCD[10:0] PPL bit field)
- 4- or 8-bit interface for Passive Matrix monochrome panel (the DSS.DISPC_CONTROL[4] M8B bit)

12.5.3.5.2 LCD Timings

The following bit fields define the timing generation of HSYNC/VSYNC:

- Horizontal front porch (the DSS.DISPC_TIMING_H[19:8] HFP bit field)
- Horizontal back porch (the DSS.DISPC_TIMING_H[31:20] HBP bit field)
- Horizontal synchronization pulse width (the DSS.DISPC_TIMING_H[7:0] HSW bit field)
- Vertical front porch (the DSS.DISPC_TIMING_V[19:8] VFP bit field)
- Vertical back porch (the DSS.DISPC_TIMING_V[31:20] VBP bit field)
- Vertical synchronization pulse width (the DSS.DISPC_TIMING_V[7:0] VSW bit field)
- On/Off control of HSYNC/VSYNC pixel clock (the DSS.DISPC_POL_FREQ[17] ONOFF bit)
- Program HSYNC/VSYNC rise or fall (the DSS.DISPC_POL_FREQ[16] RF bit)
- Invert HSYNC (the DSS.DISPC_POL_FREQ[13] IHS bit)
- Invert VSYNC (the DSS.DISPC_POL_FREQ[12] IVS bit)
- HSYNC gated (the DSS.DISPC_CONFIG[6] HSYNCGATED bit)
- VSYNC gated (the DSS.DISPC_CONFIG[7] VSYNCGATED bit)

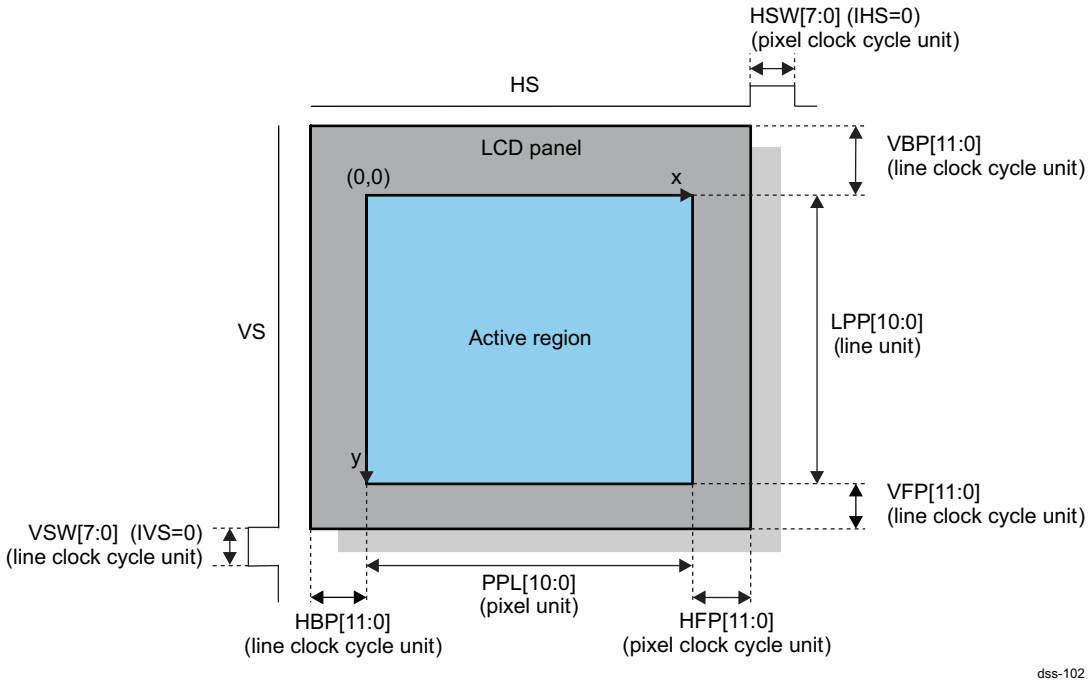
Table 12-56 describes the programming rules for LCD timing.

Table 12-56. Programming Rules

	No Downsampling	Downsampling H or V	Downsampling H + V
(HBP + HSW + HFP) * PCD	> 8	> 10	> 20

Figure 12-129 shows the timing values description in the case of an active matrix display.

Figure 12-129. Timing Values Description (Active Matrix Display)



The following bit fields define the timing generation of ac-bias (output enable in active matrix mode):

- Invert output enable (DSS.DISPC_POL_FREQ[15] IEO bit)
- ac-bias pin frequency (DSS.DISPC_POL_FREQ[7:0] ACB bit field)
- ac-bias pin transitions per interrupt (DSS.DISPC_POL_FREQ[11:8] ACBI bit field)
- ac-bias gated (DSS.DISPC_CONFIG[8] ACBIASGATED)

The following bit fields define the timing generation of the pixel clock:

- Pixel clock divisor (DSS.DISPC_DIVISOR[7:0] PCD bit field)
- Invert pixel clock (DSS.DISPC_POL_FREQ[14] IPC bit)
- Pixel clock gated (DSS.DISPC_CONFIG[5] PIXELCLOCKGATED bit)

The 8-bit pixel clock divider (the DSS.DISPC_DIVISOR[7:0] PCD bit field) selects the pixel clock frequency. This bit field generates a range of pixel clock frequencies from LC/1 to LC/255, where LC is the logic clock from the divided functional clock of the display controller by the DSS.DISPC_DIVISOR[23:16] LCD bit field.

The pixel clock is defined by the following equation:

Pixel Clock = (FunctionalClock/LCD[7:0])/PCD[7:0]

Table 12-57 through Table 12-60 show the pixel clock frequency limitations depending the panel type (active or passive matrix) and the mode (color or monochrome).

Table 12-57. Pixel Clock Frequency Limitations - RGB16 and YUV422 Active Matrix Display

Min PCD Values			Horizontal Resampling				
			Off	Up	1:1 - 1:2	1:2 - 1:3	1:3 - 1:4
Vertical Resampling	Off		2 (1)[(1)]	2 [(1)]	2	3	4
	Up		2 [(1)]	2 (1)[(1)]	2	3	4
	1:1 - 1:2	3-tap	2	2	4	6	8
		5-tap	PCDmin	PCDmin	PCDmin	PCDmin	PCDmin
	1:2 - 1:4		PCDmin	PCDmin	PCDmin	PCDmin	PCDmin

(1) The PCD value can be 1 in case all the data and synchronization signals are asserted and deasserted on the rising edge of the pixel clock.

Table 12-58. Pixel Clock Frequency Limitations - RGB16 and YUV422 Passive Matrix Display - Mono4

Min PCD Values			Horizontal Resampling				
			Off	Up	1:1 - 1:2	1:2 - 1:3	1:3 - 1:4
Vertical Resampling	Off		4	4	8	12	16
	Up		4	4	8	12	16
	1:1 - 1:2	3-tap	8	8	16	24	32
		5-tap	4xPCDmin	4xPCDmin	4xPCDmin	4xPCDmin	4xPCDmin
	1:2 - 1:4		4xPCDmin	4xPCDmin	4xPCDmin	4xPCDmin	4xPCDmin

Table 12-59. Pixel Clock Frequency Limitations - RGB16 and YUV422 Passive Matrix Display - Mono8

Min PCD Values			Horizontal Resampling				
			Off	Up	1:1 - 1:2	1:2 - 1:3	1:3 - 1:4
Vertical Resampling	Off		8	8	16	24	32
	Up		8	8	16	24	32
	1:1 - 1:2	3-tap	16	16	32	48	64
		5-tap	8xPCDmin	8xPCDmin	4xPCDmin	8xPCDmin	8xPCDmin
	1:2 - 1:4		8xPCDmin	8xPCDmin	4xPCDmin	8xPCDmin	8xPCDmin

Table 12-60. Pixel Clock Frequency Limitations - RGB16 and YUV422 Passive Matrix Display - Color

Min PCD Values			Horizontal Resampling				
			Off	Up	1:1 - 1:2	1:2 - 1:3	1:3 - 1:4
Vertical Resampling	Off		3	3	6	9	12
	Up		3	3	6	9	12
	1:1 - 1:2	3-tap	6	6	12	18	24
		5-tap	3xPCDmin	3xPCDmin	3xPCDmin	3xPCDmin	3xPCDmin
	1:2 - 1:4		3xPCDmin	3xPCDmin	3xPCDmin	3xPCDmin	3xPCDmin

NOTE: In case of RGB24 format, Figure 12-130 is still valid, except the PCDmin values which must be multiplied by two.

The PCDmin for vertical downsampling only is defined by the following equations:

Figure 12-130. PCDmin Formulas (V Down-Sampling Only)

$$h\_ratio = \frac{DISPC\_PPL\_SIZE\_LCD[10:0]PLL}{DISPC\_VIDn\_SIZE[10:0]VIDZSIZEX}$$

$$v\_ratio = \frac{DISPC\_VIDn\_PICTURE\_SIZE[10:0]VIDORGSIZEX}{DISPC\_VIDn\_SIZE[10:0]VIDSIZEX}$$

$$PCD\min = \frac{v\_ratio}{2 \times h\_ratio} \quad 1 < v\_ratio \le 2$$

$$PCD\min = \max\left(\frac{v\_ratio}{2 \times h\_ratio}, \frac{v\_ratio - 2}{2 \times (h\_ratio - 1)}\right) \quad 2 < v\_ratio \le 4$$

dss-E103

The PCDmin for horizontal downsampling only is defined by the following formula:

While downsampling by n, *PCDmin = n*

For H+V downsampling, the formula is the following:

PCDmin = max(PCDmin H only, PCDmin V only) as defined above

The refresh rate depends on the following parameters:

- Horizontal front porch (the DSS.DISPC_TIMING_H[19:8] HFP bit field)
- Horizontal back porch (the DSS.DISPC_TIMING_H[31:20] HBP bit field)
- Horizontal synchronization pulse width (the DSS.DISPC_TIMING_H[7:0] HSW bit field)
- Vertical front porch (the DSS.DISPC_TIMING_V[19:8] VFP bit field)
- Vertical back porch (the DSS.DISPC_TIMING_V[31:20] VBP bit field)
- Vertical synchronization pulse width (the DSS.DISPC_TIMING_V[7:0] VSW bit field)
- Number of lines per panel (the DSS.DISPC_SIZE[26:16] LPP bit field)
- Number of pixels per line (the DSS.DISPC_SIZE[10:0] PPL bit field)
- 4- or 8-bit interface for the passive matrix monochrome panel (the DSS.DISPC_CONTROL[4] M8B bit)

The following bit fields define the behavior of the internal blocks:

- Spatial/temporal dithering logic enabled (DSS.DISPC_CONTROL[7] SPATIALTEMPORALDITHERENABLE bit)
- Spatial/temporal dithering logic number of frames (DSS.DISPC_CONTROL[31:30] SPATIALTEMPORALDITHERFRAMES bit field). The default value of this bit field at reset time is 0x0, which is 1 frame only (spatial processing without temporal dithering). The possible values are 0x0 (one frame), 0x1 (two frames), and 0x2 (four frames). The number of frames is initialized before enabling the spatial/temporal dithering unit. The software must not change this bit field value while the spatial/temporal unit is enabled.

The following bit field defines the clock gating strategy:

- In active matrix mode, the pixel clock is always gated or only when valid data are present (the DSS.DISPC_CONFIG[0] PIXELGATED bit).

12.5.3.5.3 LCD Overlay

The following bit fields define the overlay attributes of the LCD output:

- Transparency color key (the DSS.DISPC_TRANS_COLOR0i register (i = 0))
- Transparency color key enable (the DSS.DISPC_CONFIG[10] TCKLCDENABLE bit)
- Transparency color key selection between the destination graphics transparency color key and the source video transparency color key (the DSS.DISPC_CONFIG[11] TCKLCDSELECTION bit)
- The default solid background color is defined in the DSS.DISPC_DEFAULT_COLOR_m[23:0] DEFAULTCOLOR bit field (i=0).
- Alpha blender Enable (DSS.DISPC_CONFIG[18] LCDALPHABLENDERENABLE)
- Global alpha blending values (DSS.DISPC_GLOBAL_ALPHA[23:16] VID2GLOBALALPHA and DSS.DISPC_GLOBAL_ALPHA[7:0] GFXGLOBALALPHA). The value 0xFF corresponds to 100% opaque and 0 to 100% transparent

NOTE: The destination graphics transparency color key is available only to the overlay with which the graphics pipeline is connected. The software must set the correct configuration of the LCD and digital overlays.

NOTE: When the alpha blender is enabled, the destination transparency color key is not available and the source transparency color key applies to the graphics pixels and not the video pixels.

When all of these fields are set to the appropriate values, set the DSS.DISPC_CONTROL[5] GOLCD bit to indicate that all shadow registers of the pipelines connected to the LCD output are latched by the hardware (only if the DSS.DISPC_CONTROL[0] LCDENABLE bit is already set to 1). If the LCD output is disabled, the new values will be updated when the DSS.DISPC_CONTROL[0] LCDENABLE bit will be set to 1.

12.5.3.5.4 LCD TDM

The following fields define the multiple cycle output configuration:

- First cycle (the DSS.DISPC_DATA_CYCLEk (k=0) register)
- Second cycle (the DSS.DISPC_DATA_CYCLEk (k=1) register)
- Third cycle (the DSS.DISPC_DATA_CYCLEk (k=2) register)
- Enable (the DSS.DISPC_CONTROL[20] TDMENABLE bit)
- Parallel mode (the DSS.DISPC_CONTROL[22:21] TDMPARALLEMODE field)
- Cycle format (the DSS.DISPC_CONTROL[24:23] TDMCYCLEFORMAT field)
- Unused bits (the DSS.DISPC_CONTROL[26:25] TDMUNUSEDBITS field)

When all of these bit fields are set to the appropriate values, set the DSS.DISPC_CONTROL[5] GOLCD bit to indicate that all shadow registers of the pipelines connected to the LCD output are latched by the hardware (only if the DSS.DISPC_CONTROL[0] LCDENABLE bit is already set to 1). If the LCD output is disabled, the new values will be updated when the DSS.DISPC_CONTROL[0] LCDENABLE bit will be set to 1.

12.5.3.5.5 LCD Spatial/Temporal Dithering

The following bit fields define the LCD spatial/temporal dithering configuration:

- Number of frames (the DSS.DISPC_CONTROL[31:30] SPATIALTEMPORALDITHERINGFRAMES bit field) with:
  - 0x0 Spatial only (default value)

  - 0x1 Spatial + Temporal over two frames
  - 0x2 Spatial + Temporal over four frames
  - 0x3 Reserved
- Enable (the DSS.DISPC_CONTROL[7] SPATIALTEMPORALDITHERENABLE bit)
  - 0x0 Disabled (default value)
  - 0x1 Enabled

When all of these bit fields are set to the appropriate values, set the DSS.DISPC_CONTROL[5] GOLCD bit to indicate that all shadow registers of the pipelines connected to the LCD output are latched by the hardware (only if the DSS.DISPC_CONTROL[0] LCDENABLE bit is already set to 1). If the LCD output is disabled, the new values will be updated when the DSS.DISPC_CONTROL[0] LCDENABLE bit will be set to 1.

12.5.3.5.6 LCD Color Phase Rotation

The following bit fields define the color phase rotation configuration:

- Enable (the DSS.DISPC_CONFIG[15] CPR bit)
  - 0x0 Disabled (default value)
  - 0x1 Enabled
- Red 10-bit signed coefficients used by the color phase rotation matrix (the DSS.DISPC_CPR_COEF_R register)
- Green 10-bit signed coefficients used by the color phase rotation matrix (the DSS.DISPC_CPR_COEF_G register)
- Blue 10-bit signed coefficients used by the color phase rotation matrix (the DSS.DISPC_CPR_COEF_B register)

The programmable color phase rotation block for the LCD output has nine 10-bit coefficients defined in the DSS.DISPC_CPR_COEF_R, DSS.DISPC_CPR_COEF_G, and DSS.DISPC_CPR_COEF_B, as described in Figure 12-131 through Figure 12-134.

Figure 12-131. Color Phase Rotation Matrix

$$\begin{bmatrix} Rout \\ Gout \\ Bout \end{bmatrix} = \frac{1}{256} * \left(\begin{bmatrix} RR & RG & RB \\ GR & GG & GB \\ BR & BG & BB \end{bmatrix} * \begin{bmatrix} Rin \\ Gin \\ Bin \end{bmatrix} \right)$$

dss-E105

Figure 12-132. Color Phase Rotation Matrix (R Component Only)

$$Riout = \frac{1}{256} * (RR * Rin + RG * Gin + RB * Bin)$$

dss-E106

Figure 12-133. Color Phase Rotation Matrix (G Component Only)

$$Gout = \frac{1}{256} * (GR * Rin + GG * Gin + GB * Bin)$$

dss-E107

Figure 12-134. Color Phase Rotation Matrix (B Component Only)

$$Bout = \frac{1}{256} * (BR * Rin + BG * Gin + BB * Bin)$$

dss-E104

When all of these bit fields are set to the appropriate values, set the DSS.DISPC_CONTROL[5] GOLCD bit to indicate that all shadow registers of the pipelines connected to the LCD output are latched by the hardware (only if the DSS.DISPC_CONTROL[0] LCDENABLE bit is already set to 1). If the LCD output is disabled, the new values will be updated when the DSS.DISPC_CONTROL[0] LCDENABLE bit will be set to 1.

12.5.3.5.6.1 Color Phase Rotation - Diagonal Matrix

The Color Phase Rotation feature is useful when using an LCD backlight that is not white. By using a correct configuration of the R, G and B coefficients of CPR, the color bias of the screen can be corrected. The following paragraphs give an example of configuration of the CPR feature.

The easiest example of CPR configuration is a diagonal matrix. This way, the output colors depends on one input color only. Figure 12-135 gives the example of a diagonal matrix and the corresponding equation of the output components.

Figure 12-135. Diagonal Matrix Configuration

$$\begin{bmatrix} Rout \\ Gout \\ Bout \end{bmatrix} = \frac{1}{256} * \left(\begin{bmatrix} RR & 0 & 0 \\ 0 & GG & 0 \\ 0 & 0 & BB \end{bmatrix} * \begin{bmatrix} Rin \\ Gin \\ Bin \end{bmatrix} \right)$$

$$\rightarrow \quad Rout = \frac{1}{256} * \left(RR * Rin \right)$$

$$\rightarrow \quad Gout = \frac{1}{256} * \left(GG * Gin \right)$$

$$\rightarrow \quad Bout = \frac{1}{256} * \left(BB * Bin \right)$$

dss-200

According to these 3 new equations, each output component only depends on the corresponding input color. The coefficients can easily be used to reduce the impact of a non-white backlight.

Let's take the example of a "blue" backlight. In this case, users have the feeling that a blue film has been added on the screen, and then each color seems to be "too much blue". The goal is then to reduce the "Blue" component and to keep the "Red" and "Green" ones unchanged. The following matrix can be used for a reduction by a half of the blue component. Figure 12-136 gives the corresponding matrix and equations for each component.

Figure 12-136. Example - Diagonal Matrix Configuration

$$\begin{bmatrix} Rout \\ Gout \\ Bout \end{bmatrix} = \frac{1}{256} * \left(\begin{bmatrix} 256 & 0 & 0 \\ 0 & 256 & 0 \\ 0 & 0 & 128 \end{bmatrix} * \begin{bmatrix} Rin \\ Gin \\ Bin \end{bmatrix} \right)$$

$$\rightarrow \quad Rout = \frac{1}{256} * \left(256 * Rin \right) \Rightarrow Rout = Rin$$

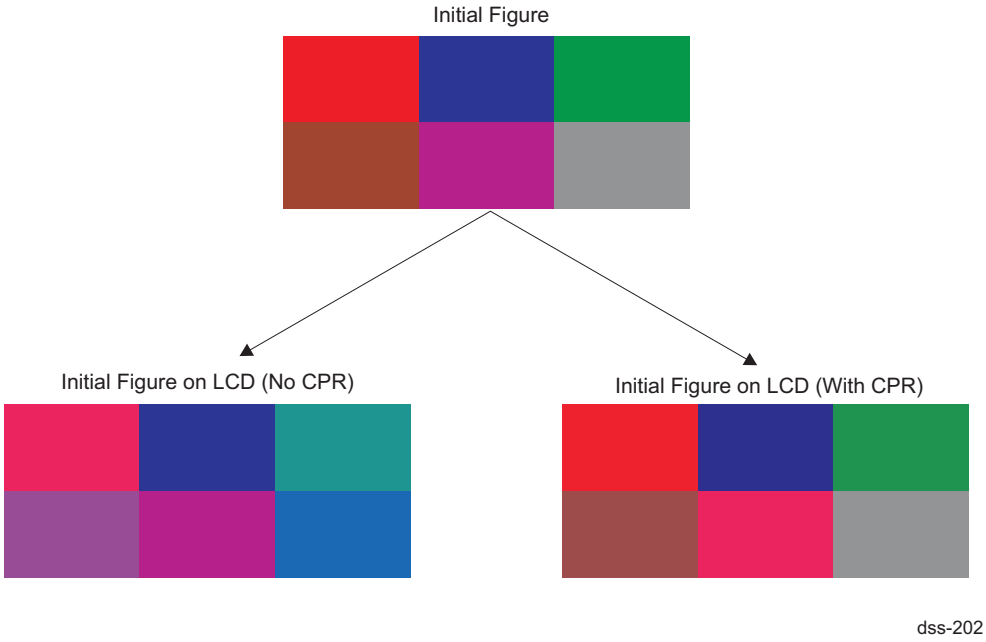
$$\rightarrow \quad Gout = \frac{1}{256} * \left(256 * Gin \right) \Rightarrow Gout = Gin$$

$$\rightarrow \quad Bout = \frac{1}{256} * \left(128 * Bin \right) \Rightarrow Bout = 0.5 * Bin$$

dss-201

Figure 12-137 shows the result of an image on a "blue" backlight screen with and without CPR.

Figure 12-137. Image With and Without CPR (Diagonal Matrix)



A drawback of this diagonal matric is that the color reduction is linear. The contrast is then different from the initial image. It is then necessary to use the 6 other coefficients of the CPR matrix to better correct a non-white backlight. The goal is to find the correct coefficients that remove the color offset added by the non-white backlight.

12.5.3.5.6.2 Color Phase Rotation - Standard Matrix

In the following example, the LCD backlight adds an offset of 128 (B_offset) to the Blue component. Figure 12-138 shows an example of matrix that reduces the offset of the screen, the corresponding equations and the resulting output colors.

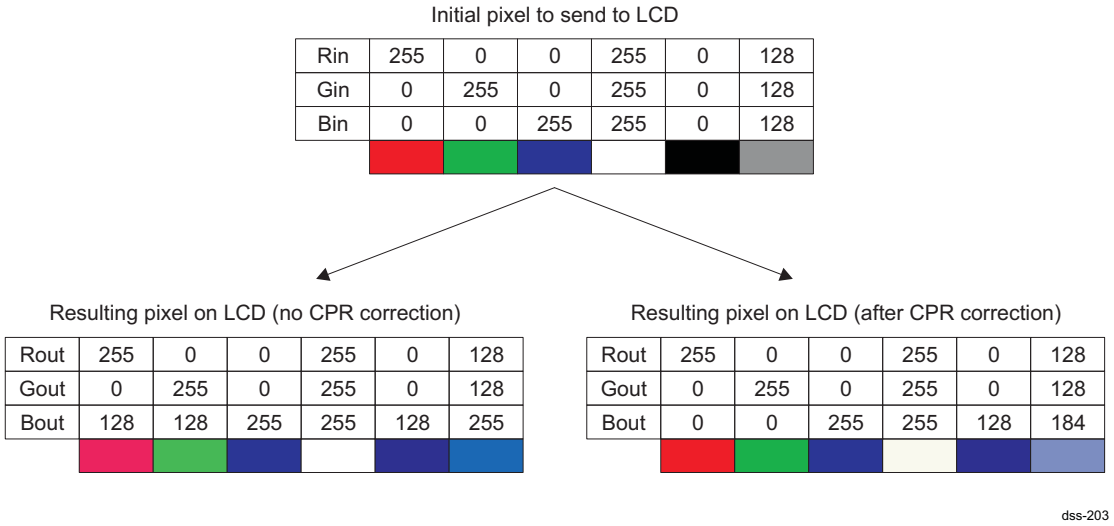
Figure 12-138. Example - Image With and Without CPR (Standard Matrix)

$$\begin{bmatrix} Rout \\ Gout \\ Bout \end{bmatrix} = \frac{1}{256} * \left(\begin{bmatrix} 256 & 0 & 0 \\ 0 & 256 & 0 \\ -129 & -129 & 370 \end{bmatrix} * \begin{bmatrix} Rin \\ Gin \\ Bin \end{bmatrix} \right)$$

$$\rightarrow \quad Rout = \frac{1}{256} * \left(256 * Rin \right)$$

$$\rightarrow \quad Gout = \frac{1}{256} * \left(256 * Gin \right)$$

$$\rightarrow \quad Bout = \frac{1}{256} * \left(-129 * Rin + -129 * Gin + 370 * Bin \right) + B\_offset$$



This CPR matrix gives inputs and outputs very close. However, black can not be corrected because of it's zero-components. No matter which coefficients are used in the matrix, the result will always be equal to the offset added by the LCD backlight.

12.5.3.6 TV Set-Specific Control Registers

The following registers define the digital output configuration:

- DSS.DISPC_CONTROL
- DSS.DISPC_CONFIG
- DSS.DISPC_DEFAULT_COLOR_m (m=1)
- DSS.DISPC_TRANS_COLOR_m (m=1)
- DSS.DISPC_SIZE_DIG

The digital output is enabled/disabled by setting/resetting the DSS.DISPC_CONTROL[1] DIGITALENABLE bit. A valid configuration must be set before the digital output can be enabled.

Perform the initialization sequence as follows:

1. Initialize the video encoder and the display controller configuration registers.
2. Set the DSS.DISPC_CONTROL[6] GODIGITAL bit and the DSS.DISPC_CONTROL[1] DIGITALENABLE bit to 1.
3. Wait for the first VSYNC pulse signal.
4. Clear the SYNCLOSTDIGITAL interrupt by setting the DSS.DISPC_IRQSTATUS[15] SYNCLOSTDIGITAL bit to 1.
5. Enable the SYNCLOSTDIGITAL interrupt by setting the DSS.DISPC_IRQENABLE[15]

SYNCLOSTDIGITAL bit to 1.

12.5.3.6.1 Digital Timings

The following bit fields define the timing information:

- Data hold time (the DSS.DISPC_CONTROL[19:17] HT bit field)
- Logic clock divisor (the DSS.DISPC_DIVISOR[23:16] LCD bit field)

The 8-bit pixel clock divider (DSS.DISPC_DIVISOR[23:16]) bit field is used to select the logic clock frequency. The LCD generates a range of pixel clock frequencies from FCK/1 to FCK/255, where FCK is the input functional clock of the display controller.

12.5.3.6.2 Digital Frame/Field Size

The following bit fields define the field size (frame if progressive mode):

- Number of lines per panel (the DSS.DISPC_SIZE_DIG[26:16] LPP bit field)
- Number of pixels per line (the DSS.DISPC_SIZE_DIG[10:0] PPL bit field)

12.5.3.6.3 Digital Overlay

The following bit fields define the overlay attributes of the digital output:

- Transparency color key (the DSS.DISPC_TRANS_COLOR_m register (m=1))
- Transparency color key enable (the DSS.DISPC_CONFIG[12] TCKDIGENABLE bit)
- Transparency color key selection between the destination graphics transparency color key and the source video transparency color key (the DSS.DISPC_CONFIG[13] TCKDIGSELECTION bit)
- The default solid background color is defined in the DSS.DISPC_DEFAULT_COLOR_m[23:0] DEFAULTCOLOR bit field (i=1).
- Alpha blender Enable (DSS.DISPC_CONFIG[19] TVALPHABLENDERENABLE)
- Global alpha blending values (DSS.DISPC_GLOBAL_ALPHA[23:16] VID2GLOBALALPHA and DSS.DISPC_GLOBAL_ALPHA[7:0] GFXGLOBALALPHA). The value 0xFF corresponds to 100% opaque and 0 to 100% transparent

NOTE: The destination graphics transparency color key is available only to the overlay with which the graphics pipeline is connected. The software must set the correct configuration of the LCD and digital overlays.

NOTE: When the alpha blender is enabled, the destination transparency color key is not available and the source transparency color key applies to the graphics pixels and not the video pixels.

When this bit field is set to the appropriate values, set the DSS.DISPC_CONTROL[6] GODIGITAL bit to indicate that all shadow registers of the pipelines connected to the digital output are latched by the hardware (only if the DSS.DISPC_CONTROL[1] DIGITALENABLE bit is already set to 1). If the digital output is disabled, the new values will be updated when the DSS.DISPC_CONTROL[1] DIGITALENABLE bit will be set to 1.

12.5.4 DSI Protocol Engine Basic Programming Model

This section describes the programming model of the DSI protocol engine.

12.5.4.1 Software Reset

The DSI protocol engine can be reset by software. This reset can be done for debug purposes or after a protocol error and has the same effect as the hardware reset. The DSI protocol engine can be reset by setting the DSS.DSI_SYSCONFIG[1] SOFT_RESET bit to 1. The software can monitor the DSS.DSI_SYSSTATUS[0] RESET_DONE status bit to wait for the completion of the reset procedure. If after 5 reads, the DSS.DSI_SYSSTATUS[0] RESET_DONE status bit still returns 0, it can be assumed that an error occurred during the reset stage.

NOTE: This software reset is optional as an hardware reset is always performed on the DSI protocol engine at device reset.

12.5.4.2 Power Management

The power management behavior of the DSI protocol engine is controlled by the DSS.DSI_SYSCONFIG register. This register completely controls the way the module interferes with the PRCM module. The DSS.DSI_SYSCONFIG[0] AUTO_IDLE bit should be set to 1 (default value) to enable automatic clock gating in the module.

12.5.4.3 Interrupts

There is a single interrupt request: DSI_IRQ. This interrupt line is merged with another interrupt line from the DISPC_IRQ in a single interrupt request DSS_IRQ. The DSI_IRQ events are generated only for the enabled VC(s). Two registers are used to enable and monitor the DSI interrupt events:

- DSS.DSI_IRQENABLE register: This register indicates the enabled/disabled event for the VCs. Each event for the VC is configured in the DSS.DSI_VCn_IRQENABLE register dedicated to the VC number. In addition, it includes one bit for the enable of error reporting for the complex I/O: Error signaling from the complex I/O: The interrupt is triggered when any error is received from the complex I/O (ErrSyncEsc[4:0], ErrEsc[4:0] (edge trigger interrupt), ErrControl[4:0] and ErrContentionLP0[4:0], ErrContentionLP1[4:0] from the complex I/O).
- DSS.DSI_IRQSTATUS register : The register DSI_IRQSTATUS flags which VC(s) is/have generated an interrupt. Based on the VC number, the register DSI_VCn_IRQSTATUS indicates the event generating the interrupt. In addition, it includes one bit for the status of error reporting for the complex I/O.

12.5.4.4 Global Register Controls

Prior to receive data from the DSI complex I/O, the DSI_PHY_SCP registers in the DSI complex I/O must be configured. Refer to Section 12.5.6 for more details. Table 12-61 details the register access width limitations for all the DSI modules.

Table 12-61. Register Access Width Limitations

Register Name	Register Access Width
All DSI complex I/O register (DSI_PHY_SCP)	32-bit only
All DSI PLL control module registers	32-bit only
DSI_VCn_LONG_PACKET_HEADER	32-bit only
DSI_VCn_SHORT_PACKET_HEADER	32-bit only
DSI_VCn_LONG_PACKET_PAYLOAD	16-bit, 32-bit
All others DSI protocol engine registers	8-bit, 16-bit and 32-bit

CAUTION

In case of different access width detailed in Table 12-61, an OCP error is generated in response to the write using SResp=ERR.

The DSI protocol engine is globally controlled by the DSS.DSI_CTRL register. The interface to the complex I/O is enabled by setting the DSS.DSI_CTRL[0] IF_EN bit. When the interface is disabled, it is possible to provide data to the TX FIFO and read pending data in the RX FIFO. When the DSS.DSI_CTRL[0] IF_EN bit is set to 1, the pending packets should be sent to the DSI complex I/O, the data transfer from the video port should be ignored only when received the next Vertical Sync Event which is received but not send to the DSI complex I/O.

When the DSS.DSI_CTRL[0] IF_EN bit is reset by software, the hardware should finish the transfer of the pending data in the TX FIFO and wait for response if BTA has been sent (Protocol engine is receive mode), then the hardware resets the DSS.DSI_CTRL[0] IF_EN bit. When using the video mode, the VC associated with the video port should be enabled prior to enable the interface according to the following sequence:

- DSS.DSI_CTRL[0] IF_EN bit is equal to 0
- Enable the VC associated with video mode by setting the DSS.DSI_VCn_CTRL[0] VC_EN
- Set the DSS.DSI_CTRL[0] IF_EN bit to 1

12.5.4.5 Virtual Channels

There is one set of registers for each VC. The attributes of the VC define the following characteristics:

- Transfer mode (DSS.DSI_VCn_CTRL[4] MODE bit):
  - Video mode
  - Command mode
- Data type
- Source (DSS.DSI_VCn_CTRL[1] SOURCE bit)
  - Video port
  - L4 interconnect port
- HS or LP forward transmission
- Automatic bus turn-around generation
  - Short packets (DSS.DSI_VCn_CTRL[2] BTA_SHORT_EN bit)
  - Long packets (DSS.DSI_VCn_CTRL[3] BTA_LONG_EN bit)
- DMA request configurations for RX and TX
  - DMA request number (DSS.DSI_VCn_CTRL[29:27] DMA_RX_REQ_NB bit field for RX FIFO and DSS.DSI_VCn_CTRL[23:21] DMA_TX_REQ_NB bit field for TX FIFO)
  - DMA threshold (DSS.DSI_VCn_CTRL[26:24] DMA_RX_THRESHOLD bit field for RX FIFO and DSS.DSI_VCn_CTRL[19:17] DMA_TX_THRESHOLD bit field for TX FIFO)
- Mode speed (DSS.DSI_VCn_CTRL[9] MODE_SPEED bit)
- ECC transmission (DSS.DSI_VCn_CTRL[8] ECC_TX_EN bit)
- CS transmission (DSS.DSI_VCn_CTRL[7] CS_TX_EN bit)

The VC ID not calculated by the DSI module but provided while writing into the registers DSI_VCn_SHORT_PACKET_HEADER and DSI_VCn_LONG_PACKET_HEADER.

12.5.4.6 Packets

The DSS.DSI_VCn_SHORT_PACKET_HEADER register is used to send only short packets (ECC can be calculated by hardware or by software user for debug purpose). The register is not used for video mode data since the short packets are generated by the hardware using the following information:

- synchronization events received on the video port (assertion/deassertion of the HSYNC and VSYNC input signals)
- DSS.DSI_CTRL[18] VP_HSYNC_END
- DSS.DSI_CTRL[17] VP_HSYNC_START
- DSS.DSI_CTRL[16] VP_VSYNC_END
- DSS.DSI_CTRL[15] VP_VSYNC_START

- DSS.DSI_CTRL[10] VP_HSYNC_POL
- DSS.DSI_CTRL[11] VP_VSYNC_POL
- DSS.DSI_VCn_CTRL[1] SOURCE

The DSS.DSI_VCn_LONG_PACKET_HEADER register is used to provide header for long packets (ECC is always calculated by hardware). The register is used for video mode and command mode. If the video mode is enabled for the VC, it is not possible to transfer concurrently (interleaved in a frame) data using long packets received on the video port and on the L4 interconnect port since the DSS.DSI_VCn_LONG_PACKET_HEADER register is used by the video mode. The register can be unprogrammed by users to send long packets received on the L4 interconnect port only when software users know that there is no expected data on the video port. The software should correctly program the register to send sequentially long packets in video and command modes.

The DSS.DSI_VCn_LONG_PACKET_PAYLOAD register is used to provide payload data for long packets (Check-sum is calculated by hardware when DSS.DSI_VCn_CTRL[7] CS_TX_EN is set to 1 otherwise the value 0x00 is used). The register is not used in video mode since payload data are provided by the video port. The software should ensure that the following sequence for write accesses to the header and payload registers (DSS.DSI_VCn_LONG_PACKET_HEADER and DSS.DSI_VCn_LONG_PACKET_PAYLOAD, respectively) is followed:

- A long packet header value with WC=0 written in DSS.DSI_VCn_LONG_PACKET_HEADER register can be followed by any access.
- A long packet header value with WC>0 written in DSS.DSI_VCn_LONG_PACKET_HEADER register should be followed by one or more writes to the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register defined by the WC value before writing again to the same DSS.DSI_VCn_LONG_PACKET_HEADER register.

CAUTION

If this sequence is not followed, no error is generated. The access to other DSI registers during this sequence is allowed.

12.5.4.7 DSI Complex I/O

Prior to send/receive any data from the complex I/O, the DSI complex I/O timings should be set according to the display module timings. The DSI complex I/O pads must also be configured first. See Section 12.5.6, *DSI Complex I/O Basic Programming Model*.

12.5.4.8 Video Mode

The DSS.DSI_VM_TIMING1, DSS.DSI_VM_TIMING2, DSS.DSI_VM_TIMING3, DSS.DSI_VM_TIMING4, DSS.DSI_VM_TIMING5, DSS.DSI_VM_TIMING6, and DSS.DSI_VM_TIMING7 registers define the timings of the video mode.

The DSS.DSI_CTRL[20] BLANKING_MODE bit defines if the long blanking packets or LPS state are used during the blanking periods (except HFP, HBP, HSA defined by other bits) when there is no pending data in TX FIFO ready to be sent. The software should ensure that there is no data in the TX FIFO, no BTA, no RESET trigger sent, and the DSS.DSI_VCn_CTRL[9] MODE_SPEED bit is set to 1 (High-Speed mode) to keep the video mode transfer is HS mode during blanking periods (except for the last blanking period since it is required to go LPS at least once per frame).

The DSS.DSI_CTRL[21] HFP_BLANKING_MODE, DSS.DSI_CTRL[22] HBP_BLANKING_MODE and DSS.DSI_CTRL[23] HSA_BLANKING_MODE define if these blanking can send packets from the TX FIFO or should be kept in HS mode using only long blanking packets.

To ensure that the writes to the register DSS.DSI_VCn_LONG_PACKET_HEADER are correctly handled as header information for video mode long packets, the following registers should be programmed:

- DSS.DSI_VCn_CTRL[0] VC_EN bit set to 1
- DSS.DSI_VCn_CTRL[4] MODE bit set to 1
- DSS.DSI_VCn_LONG_PACKET_HEADER register access

- DSS.DSI_VCn_CTRL[0] VC_EN bit set to 1

NOTE: The DSS.DSI_VCn_CTRL[1] SOURCE and DSS.DSI_VCn_CTRL[9] MODE_SPEED bits are ignored by hardware when the video mode is selected (DSS.DSI_VCn_CTRL[4] MODE bit set to 1)

The interrupt events SYNC_LOST_IRQ and RESYNCHRONIZATION_IRQ indicates if the DSI protocol engine has not been able to resynchronize the video port timing to its own timing base or if it has been done. The RESYNCHRONIZATION_IRQ indicates software users that the video port works but the configuration of the timings for the display controller (DISPC) and for DSI Protocol engine may need to be modified to avoid the resynchronization to occur. The SYNC_LOST_IRQ and RESYNCHRONIZATION_IRQ events can be respectively monitored in DSS.DSI_IRQSTATUS[18] SYNC_LOST_IRQ and DSS.DSI_IRQSTATUS[5] RESYNCHRONIZATION_IRQ status bits.

The DSS.DSI_VM_TIMING2[27:24] WINDOW_SYNC bit field defines the synchronization period. The recommended value is 0x4 based on the implementation of the resynchronization scheme.

12.5.4.9 Video Port Data Bus

The DSS.DSI_CTRL[7:6] VP_DATA_BUS_WIDTH bit field is used to determine the width of the data bus on the video port. The supported formats are 16-bit, 18-bit and 24-bits.

12.5.4.10 Command Mode

12.5.4.10.1 Command Mode TX FIFO

The single TX FIFO is used on the L4 interconnect port to receive the data to be sent to the peripheral. The configuration of the FIFO for a specific VC should be done only when the VC is disabled.

Users should not enable the VC if there is still some pending data in the TX FIFO for the corresponding space allocated for the VC from previous active period. When the VC space in the TX FIFO is empty, the VC can be enabled.

For each VC, two dedicated DSS.DSI_VCn_LONG_PACKET_HEADER and DSS.DSI_VCn_LONG_PACKET_PAYLOAD registers are used to provide data for long packets. The register DSS.DSI_VCn_SHORT_PACKET_HEADER is used to provide data for short packets (32-bit long).

For each long packet, the DSS.DSI_VCn_LONG_PACKET_HEADER register should be written first and then the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register. The only exception is when the word count defined in the header is equal to 0. In that case, it is not required to write into the payload register. For consecutive long packets, the header should be written into the DSS.DSI_VCn_LONG_PACKET_HEADER register even if the value remains the same.

The TX FIFO stores all the pending bytes to be sent to the peripheral(s). Multiple receivers can be addressed using the VC capability.

The 32-bit write requests only for each VC to the TX FIFO should be kept in order while sending the data to the DSI_PHY inside the VC requests. The only exception is in the case of last 32-bit write for the last bytes of the payload data since it could be 1, 2, 3, or 4 bytes.

Also in case the last transfer is a 32-bit write but the number of valid bytes is 1, 2, or 3 only (calculated using the header word count and the number of bytes are received for the payload), the hardware should store the 32-bit value into the TX FIFO but the invalid bytes are not sent, and are discarded.

When the word count defined in DSS.DSI_VCn_LONG_PACKET_HEADER register is not a multiple of the request threshold value defined in DSS.DSI_VCn_CTRL[19:17] DMA_TX_THRESHOLD bit field, 32-bit requests and/or bytes should be discarded by the hardware to store in FIFO only the exact number of valid bytes.

The DSI protocol module should be able to determine if the bytes in the TX FIFO correspond to a short or long packet without decoder the DT field. When the bytes are written into the DSS.DSI_VCn_SHORT_PACKET_HEADER, DSS.DSI_VCn_LONG_PACKET_HEADER, and DSS.DSI_VCn_LONG_PACKET_PAYLOAD registers, the hardware should store the information concerning long or short packet. A 1-bit flag should be used for each entry of the TX FIFO.

When the VC is disabled, the remaining bytes in the FIFO should be sent to the DSI link. The start event to send data to the DSI link one of the following events:

- All bytes have been received in the FIFO (header + payload).
- The space of the FIFO allocated for the VC is full.
- The space of the FIFO allocated for the VC is not enough to request more data using DMA request (threshold value bigger than space left in the TX FIFO for the VC).

NOTE: In case the video mode is active, the blanking period should be large enough to allow the transfer of the packet(s).

Sequential arbitration must be set (TX_FIFO_ARBITRATION[3] DSI_CTRL bit = 0x1) if only one VC is used to send multiple packets during the same blanking period.

When consecutive packets should be sent in HS mode, to ensure that there is no LP transition between them at least one of the following condition should be valid:

- Packets from the same VC
- Short packets or long packets with a payload size multiple of 4 bytes

To flush the FIFO (discard of the data) for some pending bytes, the software should change the allocated size of the chuck FIFO by:

- First disabling the VC resetting DSS.DSI_VCn_CTRL[0] VC_EN bit to 0
- Second change the size of the space of FIFO to 0 by writing the DSS.DSI_TX_FIFO_VC_SIZE VCn_FIFO_SIZE (n is the VC value between 0 and 3)

If there is an on-going packet transfer from the TX FIFO to DSI_PHY, the flush of the FIFO should stop immediately the transfer. Because the FIFO is accessible to users, it must ensure that there are no bytes left in the FIFO before starting the flush operation. But it can be required in dead-lock situation to flush the FIFO even if there are still bytes in the FIFO, in that case, the software should also take care of having a known state of the whole DSI protocol engine module (software reset may be required) To start of new transfer through the TX FIFO.

Users can check that there is no pending request before changing the size of the allocated FIFO for the VC by reading the relevant DSS.DSI_VCn_CTRL[15] VC_BUSY bit or by using the interrupt PACKET_SENT_IRQ: All the packets have been sent by counting the transferred requests to the L4 interconnect port and the number of requests sent to the DSI complex I/O. This interrupt can be monitoring by reading the DSS.DSI_VCn_IRQSTATUS[2] PACKET_SENT_IRQ status bit.

The DSS.DSI_CTRL[3] TX_FIFO_ARBITRATION bit defines if the arbitration scheme is:

- Round-robin between enabled VCs with pending ready requests (pending ready request means that all bytes for the packets are in the FIFO or the space of the FIFO for the VC is full) starting from the VC which has the least VC ID number.
- Sequential: All the pending ready requests for one VC are sent before moving to another VC. The condition of "space of the FIFO is full" should be evaluated after the end of each packet.

If users want to use sequential arbitration for all requests for all channels, a single VC should be used. (the VC ID defined in the header provided to the hardware using either the DSS.DSI_VCn_LONG_PACKET_HEADER or DSS.DSI_VCn_SHORT_PACKET_HEADER register is not used and not modified by the DSI protocol engine).

The register DSS.DSI_TX_FIFO_VC_SIZE defines the allocated number of 33-bit values for each VC in the TX FIFO and the start address for each VC. The size of the space allocated in the TX FIFO defined by DSS.DSI_TX_FIFO_VC_SIZE VCn_FIFO_SIZE (n corresponds to the VC number n) bit fields should be a multiple of the threshold defined in DSS.DSI_VCn_CTRL[19:17] DMA_TX_THRESHOLD bit field. Only the enabled VCs should be taken into account. To change the size of the space of the memory allocated for a specific VC, the VC should be disabled by setting the DSS.DSI_VCn_CTRL[0] VC_EN bit to 0. The whole FIFO may not be used by all the VCs at a given time since a VC can be disabled to change one or multiple parameters. Software users are responsible for correctly configuring the start address and the size for each VC.

Table 12-62 indicates the corresponding values for the size of the space allocated in the FIFO.

Table 12-62. Virtual Channel TX FIFO Size Values

DSI_TX_FIFO_VC_SIZE.VCn_FIFO_SIZE[n = 0, 3]	Space Size (up to the size of the FIFO)
0	0 x 33 bits
1	32 x 33 bits
2	64 x 33 bits
3	96 x 33 bits
4	128 x 33 bits

The total size of TX FIFO is 128*33 bits. Therefore, the sum of all virtual channel FIFO allocation cannot exceed 128*33 bits.

Table 12-63 indicates the start address of the space in the FIFO.

Table 12-63. Virtual Channel TX FIFO Start Address

DSI_TX_FIFO_VC_SIZE.VCx_FIFO_ADD[x = 0, 2]	Start Address
0	0
1	32
2	64
3	96
4	128

CAUTION

There must be no overlap of different VCs spaces.

When the TX FIFO is full:

- the overflow interrupt (FIFO_TX_OVF_IRQ) is generated. To monitor this interrupt request, users can read the DSS.DSI_VCn_IRQSTATUS[3] FIFO_TX_OVF_IRQ status bit.
- there is no L4 interconnect error generated
- the commands are accepted but the data are not written into the FIFO

To ensure that all writes are correctly stored in the TX FIFO, the FIFO should not be full. The software must read the room in the space allocated for the VC in the TX FIFO by reading the DSS.DSI_TX_FIFO_VC_EMPTINESS register. In case there is no space allocated in the TX FIFO for the VC, the DSS.DSI_TX_FIFO_VC_EMPTINESS register indicates a value of 0 for the VC space emptiness.

When waiting to receive the first VSYNC event on the video port to start the video mode on DSI link no command data from TX FIFO should be sent on the interface. It is required to ensure that when receiving the VSYNC event, there is no on-going command mode transfer that could delay the start of video mode on the DSI link.

12.5.4.10.2 Command Mode RX FIFO

The RX FIFO is used to store the data received from the DSI complex I/O. The data are always packed in the RX FIFO (single or multiple packets receiving during a single or multiple BTA periods).

The read requests access to corresponding VC locations to transfer data for a specific VC. The logic managing the FIFO must be able to extract 32-bit values in-order for a specific VC upon read requests. The byte enable of the read access is ignored. Each read returns one 32-bit value from the RX FIFO. If the application accesses the RX FIFO should extract always 32-bit values. Only in the case of 1, 2, or 3 bytes are remaining in the RX FIFO.

The read requests (single or burst) can be less, equal or greater than the packet size. If the packet size is smaller than the read request, the following packet(s) is also transferred. If the packet size is longer than the read request, only part of the packet is transfer. In that case, the logic should keep the VC information to provide the rest of the data during the next read request(s).

The register DSS.DSI_RX_FIFO_VC_SIZE defines the allocated number of 33-bit values for each VC in the RX FIFO and the start address for each VC. Only the enabled VCs should be taken into account. to change the size of the space of the memory allocated for a specific VC, the VC should be disabled by resetting the DSS.DSI_VCn_CTRL[0] VC_EN bit to 0. The whole FIFO may not be used by the entire VC at a given time since a VC can be disabled to change one or multiple parameters. Software users are responsible for correctly configuring the start address and the size for each VC.

Table 12-64 indicates the corresponding values for the size of the space allocated in the FIFO:

Table 12-64. Virtual Channel RX FIFO Size Values

DSI_RX_FIFO_VC_SIZE.VCx_FIFO_SIZE[x = 0, 3]	Space Size (up to the size of the FIFO)
0	0 x 33 bits
1	32 x 33 bits
2	64 x 33 bits
3	96 x 33 bits
4	128 x 33 bits

The total size of RX FIFO is 128*33 bits. Therefore, the sum of all virtual channel FIFO allocation cannot exceed 128*33 bits.

Table 12-65 indicates the start address of the space in the FIFO.

Table 12-65. Virtual Channel RX FIFO Start Address

DSI_RX_FIFO_VC_SIZE.VCx_FIFO_ADD[x = 0, 2]	Start Address
0	0
1	32
2	64
3	96
4	128

CAUTION

There must be no overlap of different VCs spaces.

While reading the received bytes in the RX FIFO, only the DSS.DSI_VCn_SHORT_PACKET_HEADER register is used since the hardware does not keep track of the header position for long packets and start/end of each packet. The software must extract the information from the bytes read from the RX FIFO. There is no specific hardware to track the received bytes in the RX FIFO. The DSS.DSI_VCn_LONG_PACKET_HEADER and DSS.DSI_VCn_LONG_PACKET_PAYLOAD registers are not used.

The ECC is only used by the first header when receiving multiple packets during the same LP RX transfer from the peripheral since the DSI Protocol engine does not parse the header to identify the length of the packets. In case of multiple packets, the Check-sum does not be enabled since the hardware checks the check-sum considering a single packet. The ECC in the first header is used to correct and check the header. For the following headers in the same LP RX transfer, the hardware does not detect any header and can not check or/and detect errors in the headers of the packets except for the first packet.

When the RX FIFO is empty:

- there is no OCP error generated
- the commands are accepted and the data for the responses are 0s

12.5.4.10.3 Command Mode DMA Requests

The DMA requests (DSI_DMA_REQ) are used to allow automatic transfer by the system DMA or MPU (with less efficiency and through-put capability) from the DSI RX FIFO to the system memory and from the system memory to the DSI TX FIFO. Two independent DMA requests for RX FIFO and TX FIFO for the same VC are supported. The read and write accesses can use burst structure. The DSI protocol engine should access each write request in a burst without any IDLE between. For read OCP request in a burst to RX FIFO, the acceptance is sent immediately and the response is delayed by at least four L4 interface clock (DSS_L4_ICLK) cycles. In case of register reads, the response is returned in the first clock cycle.

The thresholds used for requests for the TX FIFO and RX FIFO are programmable by software. Users program the DSS.DSI_VCn_CTRL[19:17] DMA_TX_THRESHOLD and DSS.DSI_VCn_CTRL[26:24] DMA_RX_THRESHOLD bit fields for TX FIFO and RX FIFO respectively. The hardware asserts the DMA request based on the threshold value. The size of the space allocated in TX FIFO for each VC should be a multiple of DSS.DSI_VCn_CTRL[19:17] DMA_TX_THRESHOLD bit field value.

The only exception is in the case of the RX FIFO when the LP data transfer finishes and the threshold value is not reached. In that case the DMA request should be asserted. So the drain of the FIFO is supported in that configuration to empty the FIFO even if the number of data received is not a multiple of the threshold value.

In case of TX FIFO, if all the bytes defined by the word count field in the DSS.DSI_VCn_LONG_PACKET_HEADER header register have been received, the DMA request is not asserted anymore even during the last transfer less than DMA_TX_THRESHOLD number of bytes have been received because of the word count being not a multiple of the DMA_TX_THRESHOLD value.

In case of RX FIFO, while the DMA request is used to transfer the data from the RX FIFO to the system memory, the system DMA should be programmed to read the exact number of received bytes in the FIFO. If users do not know the size of the received bytes, the direct access of the RX FIFO through the DSS.DSI_VCn_SHORT_PACKET_HEADER register is performed until the DSS.DSI_VCn_CTRL[20] RX_FIFO_NOT_EMPTY bit goes to 0.

The use of each DMA request is programmable by software. The DSS.DSI_VCn_CTRL[23:21] DMA_TX_REQ_NB is dedicated to DMA request numbering for the TX FIFO. The DSS.DSI_VCn_CTRL[29:27] DMA_RX_REQ_NB is dedicated to DMA request numbering for the RX FIFO.

When the DMA request is used to indicate the number of 32-bit values ready in the RX FIFO or BTA has been received from peripheral indicating end of the transfer from peripheral to host for a transfer to the system memory, the DMA request corresponding to the VC ID is generated.

The system DMA transfers the number of 32-bit values defined in the threshold register or the exact number of bytes received from the peripheral (user should know the number of expected received bytes to program correctly the system DMA). When the system DMA transfers a multiple number of threshold value, the DSI protocol engine should send 0s for the data when there is no more received data in the RX FIFO for the VC. Software users are responsible for parsing the data and determine the valid bytes.

Software users can decide to determine the number of data received in the RX FIFO to read the information in the DSS.DSI_RX_FIFO_VC_FULLNESS register. Then the system DMA can be programmed to read the exact number of bytes from the RX FIFO. The BTA interrupt (BTA_IRQ) should be used to know when to read the number of received bytes. To monitor the BTA interrupt, the user should read the DSS.DSI_VCn_IRQSTATUS[5] BTA_IRQ status bit. The DMA request should not be selected until the system DMA is programmed with the correct number of data to read from RX FIFO.

If the RX FIFO space for the VC is expected to be overflow because the number of data to be received is greater than the space allocated for the VC, the previous programming model should be used. In place, the DMA request should be asserted as soon as the threshold is reached or when BTA is received.

When the DMA request is used to indicate the number of 33-bit entries empty in the TX FIFO for a transfer from the system memory, the DMA request corresponding to the VC ID is generated.

NOTE: To obtain best efficiency of the transfer the size of the request (read or write, single or burst) should be aligned with the threshold value.

Concurrent access using interlaced requests (read/write) to the TX and RX FIFO is supported for the same VC ID or different VC IDs.

12.5.4.11 Ultra-Low Power State

This section describes how to enter/exit to/from ultralow-power state (ULPS).

NOTE: The DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIGy bits (x range is 1 to 3 corresponding to lane #1 to lane #3 and y range is 1 to 2) must be read back after writing to verify that the write operations are effective before proceeding to the next step. This is to take latency at low TxClkEsc frequencies into account.

12.5.4.11.1 Entering ULPS

To enter into ULPS for a clock lane, the following sequence is required:

1. Wait for DSS.DSI_COMPLEXIO_CFG2[16] HS_BUSY and DSS.DSI_COMPLEXIO_CFG2[17] LP_BUSY bits to be reset to 0 and ensure that the DSS.DSI_CLK_CTRL[13] DDR_CLK_ALWAYS_ON bit is 0.
2. TxUlpsClk state should change from inactive to active by setting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 1.

To enter into ULPS for a data lane, the following sequence is required:

1. Wait for all TX_FIFOs for all VCs working in HS are empty, for video mode is not active, and for DSS.DSI_COMPLEXIO_CFG2[16] HS_BUSY bit is reset to 0 (in addition, for data lane #1, DSS.DSI_COMPLEXIO_CFG2[17] LP_BUSY bit is reset to 0)
2. TxRequestEsc state should change from inactive to active by setting the DSS.DSI_COMPLEXIO_CFG2. LANEx_ULPS_SIG2 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 1.

NOTE: When the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 and DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 bits are written to 0, they can be combined into one write. Both bits must be read back to confirm they are effective before proceeding.

12.5.4.11.2 Exiting ULPS

To exit from ULPS for a clock lane, the following sequence is required:

1. Change the state of TxUlpsExit for each lane to ACTIVE by setting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 1.
2. Wait for the ULPSACTIVENOT_ALL1_IRQ interrupt indicating that all lanes with TxUlpsExit active have acknowledged by asserting UlpsActiveNot. This is performed by monitoring the DSS.DSI_COMPLEXIO_IRQSTATUS[31] ULPSACTIVENOT_ALL1_IRQ status bit.
3. Start the wake-up timer (GPTimer).
4. Wait for the time-out.
5. Change TxUlpsClk signals to INACTIVE state for the clock lane by resetting the

DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0.

6. Reset the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0.

NOTE: When the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 and DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 bits are both being written to 0, they can be combined into one write. Both bits must be read back to confirm they are effective before proceeding.

To exit from ULPS for a clock lane, in case ComplexIO is in OFF state (the DSI protocol engine sends ComplexIO to OFF state by setting DSS.DSI_COMPLEXIO_CFG1[28:27] PWROFF = 0x0), the sequence is:

1. Change TxUlpsClk signals to INACTIVE state for the clock lane by resetting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0.
2. Change the state of TxUlpsExit for clock lane to INACTIVE state by resetting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0. This step is necessary only in case a PWROFF command (the command for power control of the complex I/O) is issued while the sequence for exiting is in progress (TxUlpsExit signal is already in ACTIVE state).

NOTE: When the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 and DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 bits are both being written to 0, they can be combined into one write. Both bits must be read back to confirm they are effective before proceeding.

To exit from ULPS for a data lane, the following sequence is required:

1. Change the state of TxUlpsExit for each lane to ACTIVE by setting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 1.
2. Wait for the ULPSACTIVENOT_ALL1_IRQ interrupt indicating that all lanes with TxUlpsExit active have acknowledged by asserting UlpsActiveNot. This is performed by monitoring the DSS.DSI_COMPLEXIO_IRQSTATUS[31] ULPSACTIVENOT_ALL1_IRQ status bit.
3. Start the application wake-up timer (GPtimer).
4. Wait for the time-out.
5. Change TxRequestEsc signals to INACTIVE state for the data lane by resetting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0.
6. Reset the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0.

NOTE: When the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 and DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 bits are both being written to 0, they can be combined into one write. Both bits must be read back to confirm they are effective before proceeding.

To exit from ULPS for a data lane, in case ComplexIO is in OFF state (the DSI protocol engine sends ComplexIO into OFF state by setting DSS.DSI_COMPLEXIO_CFG1[28:27] PWROFF = 0x0), the sequence is:

1. Change TxRequestEsc signals to INACTIVE state by resetting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0.
2. Change the state of TxUlpsExit to INACTIVE state by resetting the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 (x range is 1 to 3 corresponding to lane #1 to lane #3) bit to 0. This step is necessary only in case a PWROFF command (the command for power control of the complex I/O) is

issued while the sequence for exiting is in progress (TxUlpsExit signal is already in ACTIVE state).

NOTE: When the DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG2 and DSS.DSI_COMPLEXIO_CFG2 LANEx_ULPS_SIG1 bits are both being written to 0, they can be combined into one write. Both bits must be read back to confirm they are effective before proceeding.

When the sequence for entering/exiting into/from ULP state is started for specific lanes, users should wait for the completion of the sequence before changing the state of the same or other lanes.

12.5.4.12 DSI Programming Sequence Example

This section describes distinct configurations of the DSI protocol engine to support different type of traffics.

NOTE: When the VC is used to send video mode data from the video port, the DSS.DSI_VCn_CTRL[9] MODE_SPEED and the DSS.DSI_VCn_CTRL[1] SOURCE bits are ignored.

12.5.4.12.1 Video Mode Transfer

Description: One channel, video mode, no DMA requests, no bus turn-around.

1. Configure the display controller with the timing parameters.
2. Configure the DSS.DSI_VCn_CTRL register as follows:
   - SOURCE bit is ignored by hardware
   - BTA_LONG_EN bit is set to 0: No BTA on long packet
   - BTA_SHORT_EN bit is set to 0: No BTA on short packet
   - MODE bit set to 1: The video mode is selected
   - MODE_SPEED bit is ignored by hardware
3. Configure the DSS.DSI_VM_TIMING1 to DSS.DSI_VM_TIMING7 registers.
4. Set the ForceTxStopMode bit to 1 in the DSS.DSI_TIMING1 register.
5. Enable the channel by setting the DSS.DSI_VCn_CTRL[0] VC_EN bit to 1
6. Enable the module by setting the DSS.DSI_CTRL[0] IF_EN bit to 1.
7. Poll the ForceTxStopMode bit to 0 in the DSS.DSI_TIMING1 register.
8. Enable the LCD video output by setting the DSS.DISPC_CONTROL[0] LCDENABLE bit to 1

CAUTION

The restriction for stopping the video mode is that no frame should be sent by the display controller (DISPC) after disabling video mode in DSI.

12.5.4.12.2 Command Mode Transfer Example 1

CAUTION

In DSI command mode, the display controller must be configured in stall mode by setting the DSS.DISPC_CONTROL[11] STALLMODE bit to 1.

Description: One channel, command mode, no DMA requests, manual bus turn-around

1. Configure the DSS.DSI_VCn_CTRL register as follows:
   - SOURCE bit set to 0: The source is the L4 interconnect port

- BTA_LONG_EN bit is set to 0: No automatic BTA on long packet
- BTA_SHORT_EN bit is set to 0: No automatic BTA on short packet
- MODE bit set to 0: The command mode is selected

2. Enable the packet sent interrupt by setting the DSS.DSI_VCn_IRQENABLE[2] PACKET_SENT_IRQ_EN bit to 1
3. Set the ForceTxStopMode bit to 1 in the DSS.DSI_TIMING1 register.
4. Enable the channel by setting the DSS.DSI_VCn_CTRL[0] VC_EN bit to 1
5. Enable the module by setting the DSS.DSI_CTRL[0] IF_EN bit to 1
6. Poll the ForceTxStopMode bit to 0 in the DSS.DSI_TIMING1 register.
7. For long packet:
   - Write the header value into the DSS.DSI_VCn_LONG_PACKET_HEADER register
   - Write the data into the DSS.DSI_VCn_LONG_PACKET_PAYLOAD register for the full payload. Repeat step until WC is reached (see DSI_VCn_LONG_PACKET_HEADER)

   For the short packet:
   - Send short packets through the L4 interconnect:Write the header value into the DSS.DSI_VCn_SHORT_PACKET_HEADER register. Repeat step for each short packet.
8. Send one or more packets through L4 interconnect:
   - Write the header value into DSS.DSI_VCn_LONG_PACKET_HEADER register
   - Write the data into DSS.DSI_VCn_LONG_PACKET_PAYLOAD register for the full payload.
   - Repeat step 5 for all the long packets
9. Send short packets through L4 interconnect:
   - Write the header value into DSS.DSI_VCn_SHORT_PACKET_HEADER register
   - Repeat step 6 for all the short packets
10. Interrupt routine: Wait for PACKET_SENT_IRQ interrupt generation by polling the DSS.DSI_VCn_IRQSTATUS[2] PACKET_SENT_IRQ status bit and notify the application software when received.
11. The applicative software forces the bus turn-around:
    - Wait until the PACKET_SENT_IRQ has happened as many times as the number of sent packets
    - Set the DSS.DSI_VCn_CTRL[6] BTA_EN bit to 1 to send manually a BTA
    - Wait until the DSS.DSI_VCn_CTRL[6] BTA_EN bit is reset to 0 by hardware
12. Receive the packets from the peripheral
    - Start polling the DSS.DSI_VCn_CTRL[20] RX_FIFO_NOT_EMPTY status bit
    - Whenever the RX_FIFO_NOT_EMPTY bit equals to 1, read one word in the RX FIFO

12.5.4.12.3 Command Mode Transfer Example 2

CAUTION

In DSI command mode, the display controller must be configured in stall mode by setting the DSS.DISPC_CONTROL[11] STALLMODE bit to 1.

Description: One channel, command mode, DMA request, automatic bus turn-around

1. Configure the DSS.DSI_VCn_CTRL register as follows:
   - SOURCE bit set to 0: The source is the L4 interconnect port
   - BTA_LONG_EN bit is set to 1: Automatic BTA on long packet
   - BTA_SHORT_EN bit is set to 1: Automatic BTA on short packet
   - MODE bit set to 0: The command mode is selected
2. Enable the packet sent interrupt by setting the DSS.DSI_VCn_IRQENABLE[2]

PACKET_SENT_IRQ_EN bit to 1

3. Set the ForceTxStopMode bit to 1 in DSS.DSI_TIMING1 register.
4. Enable the channel by setting the DSS.DSI_VCn_CTRL[0] VC_EN bit to 1
5. Configure the TX FIFO threshold and DMA requests parameters:
   - Program the DSS.DSI_VCn_CTRL[19:17] DMA_TX_THRESHOLD bit field
   - Program the DSS.DSI_VCn_CTRL[23:21] DMA_TX_REQ_NB bit field
6. Program the system DMA to be ready to send data to the L4 interconnect port
7. Enable the module by setting the DSS.DSI_CTRL[0] IF_EN bit to 1
8. Poll the ForceTxStopMode bit to 0 in DSS.DSI_TIMING1 register.
9. Write the header value into DSS.DSI_VCn_LONG_PACKET_HEADER register
10. Interrupt routine: Wait for PACKET_SENT_IRQ interrupt generation by polling the DSS.DSI_VCn_IRQSTATUS[2] PACKET_SENT_IRQ status bit and notify the application software when received.
11. Receive the packets from the peripheral
    - Start polling the DSS.DSI_VCn_CTRL[20] RX_FIFO_NOT_EMPTY status bit
    - Whenever the RX_FIFO_NOT_EMPTY bit equals to 1, read one 32-bit word in the RX FIFO
12. Repeat the steps 7 for all long packets.

12.5.5 DSI PLL Controller Basic Programming Model

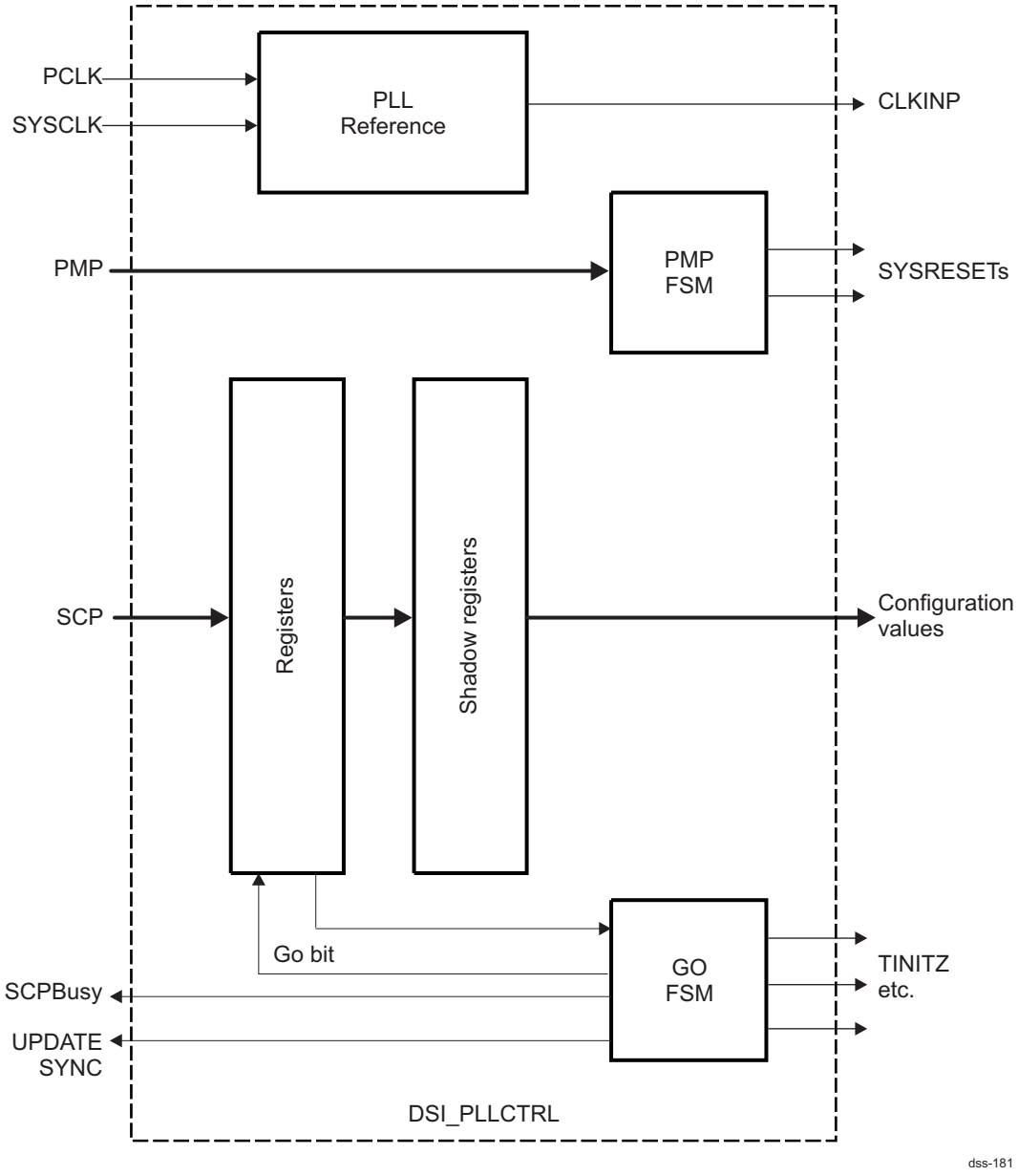
12.5.5.1 Software Reset

The DSI PLL control module does not have its own software reset. It is reset by the DSI protocol engine. Nevertheless, software users can monitor the reset status of the DSI PLL control module by reading the DSS.DSI_PLL_STATUS[0] DSI_PLLCTRL_RESET_DONE status bit.

12.5.5.2 DSI PLL Programming Blocks

Figure 12-139 shows the DSI PLL programming blocks.

Figure 12-139. DSI PLL Programming Blocks



12.5.5.3 DSI PLL Go Sequence

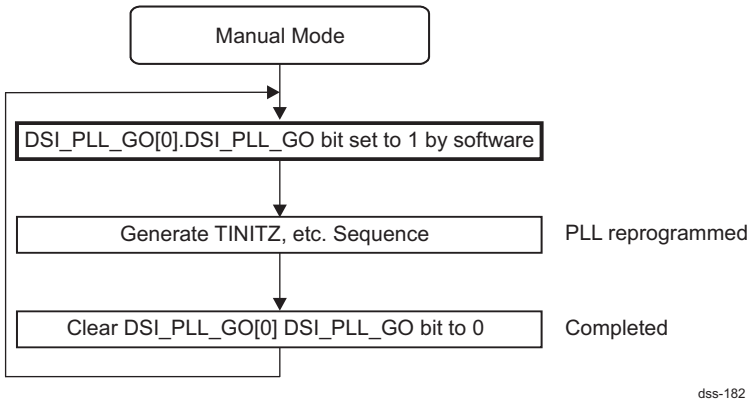
In Manual Mode (DSS.DSI_PLL_CONTROL[0] DSI_PLL_AUTOMODE bit set to 0), the DPLL requires a sequence on TINITZ, TENABLE and TENABLEDIV to update the configuration values and start the locking sequence.

Once all the configuration values have been programmed into the registers, the GO bit should be set. The appropriate sequence should then be sent on the TINITZ, TENABLE & TENABLEDIV pins, respecting the timing requirements of the ADPLLV2. The DSS.DSI_PLL_GO[0] DSI_PLL_GO bit will be cleared to 0 at the end of the sequence.

The TENABLEDIV signal is shared with the HSDIVIDER module, so that will be programmed at the same time. In this mode, the software should deassert CLKINEN by unsetting the DSS.DSI_PLL_CONFIGURATION2[14] DSI_PHY_CLKINEN to 0 and to assert HSDIVBYPASS correctly by setting the DSS.DSI_PLL_CONFIGURATION2[20] DSI_HSDIVBYPASS bit to 1 to prevent uncontrolled frequencies affecting the DSI_PHY and display subsystem during PLL locking. In manual mode the shadow register should be updated anyway so that valid values are present when later selecting automatic mode.

Figure 12-140 shows the DSI PLL Go flowchart in manual mode (DSS.DSI_PLL_CONTROL[0] DSI_PLL_AUTOMODE bit set to 0).

Figure 12-140. DSI PLL Go Sequence (Manual Mode)

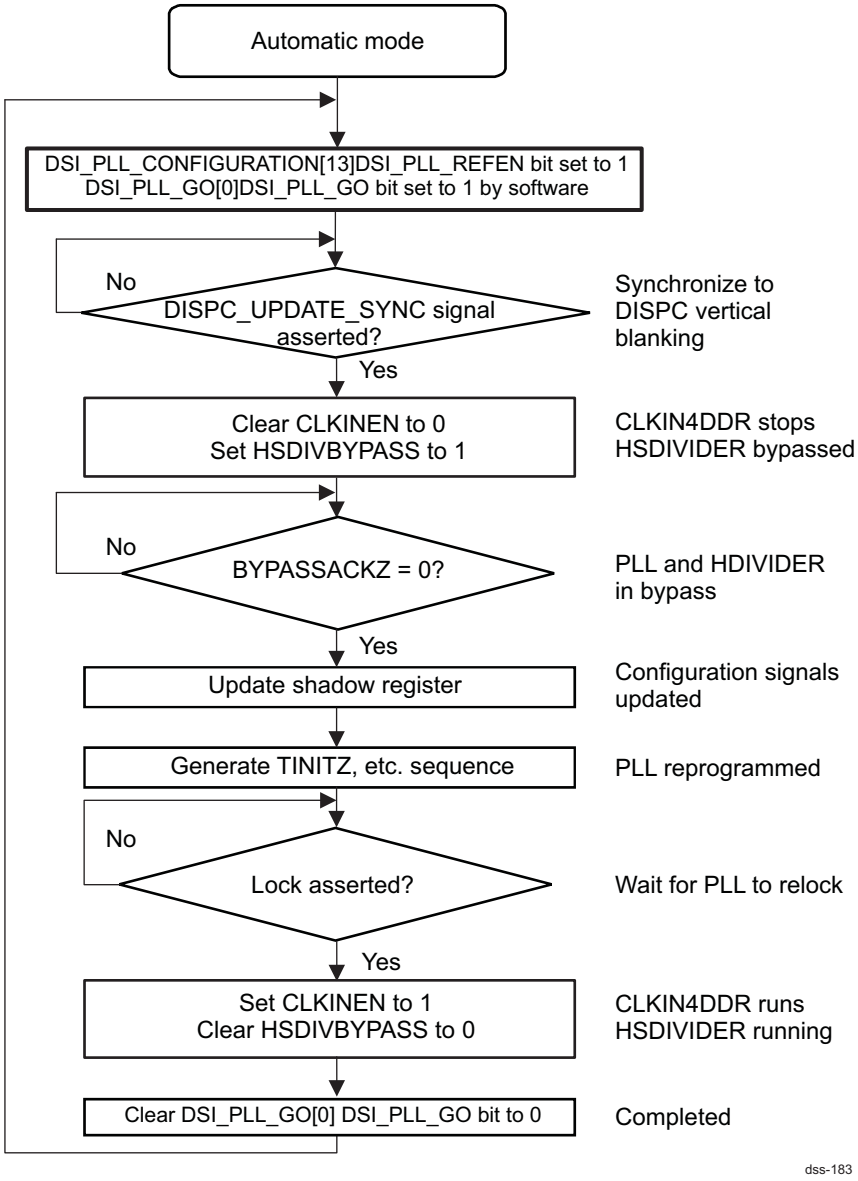


NOTE: All thick-outlined blocks show operations performed by software. Other blocks show operations performed by hardware.

In automatic Mode (DSS.DSI_PLL_CONTROL[0] DSI_PLL_AUTOMODE bit set to 1),the TINITZ, TENABLE and TENABLEDIV sequence and the update of the PLL configuration from the DSI_PLL_CONFIGURATION2 register will be deferred until the time of the front porch time signal sent by the DISPC module. This is intended to simplify the software to implement a configuration change (such as a frequency change to support a different link bandwidth). In this mode CLKINEN, HSDIVBYPASS and REFEN will be controlled automatically and the register value is overridden.

Figure 12-141 shows the DSI PLL Go flowchart in automatic mode (DSS.DSI_PLL_CONTROL[0] DSI_PLL_AUTOMODE bit set to 1).

Figure 12-141. DSI PLL Go Sequence (Automatic Mode)



NOTE: All thick-outlined blocks show operations performed by software. Other blocks show operations performed by hardware.

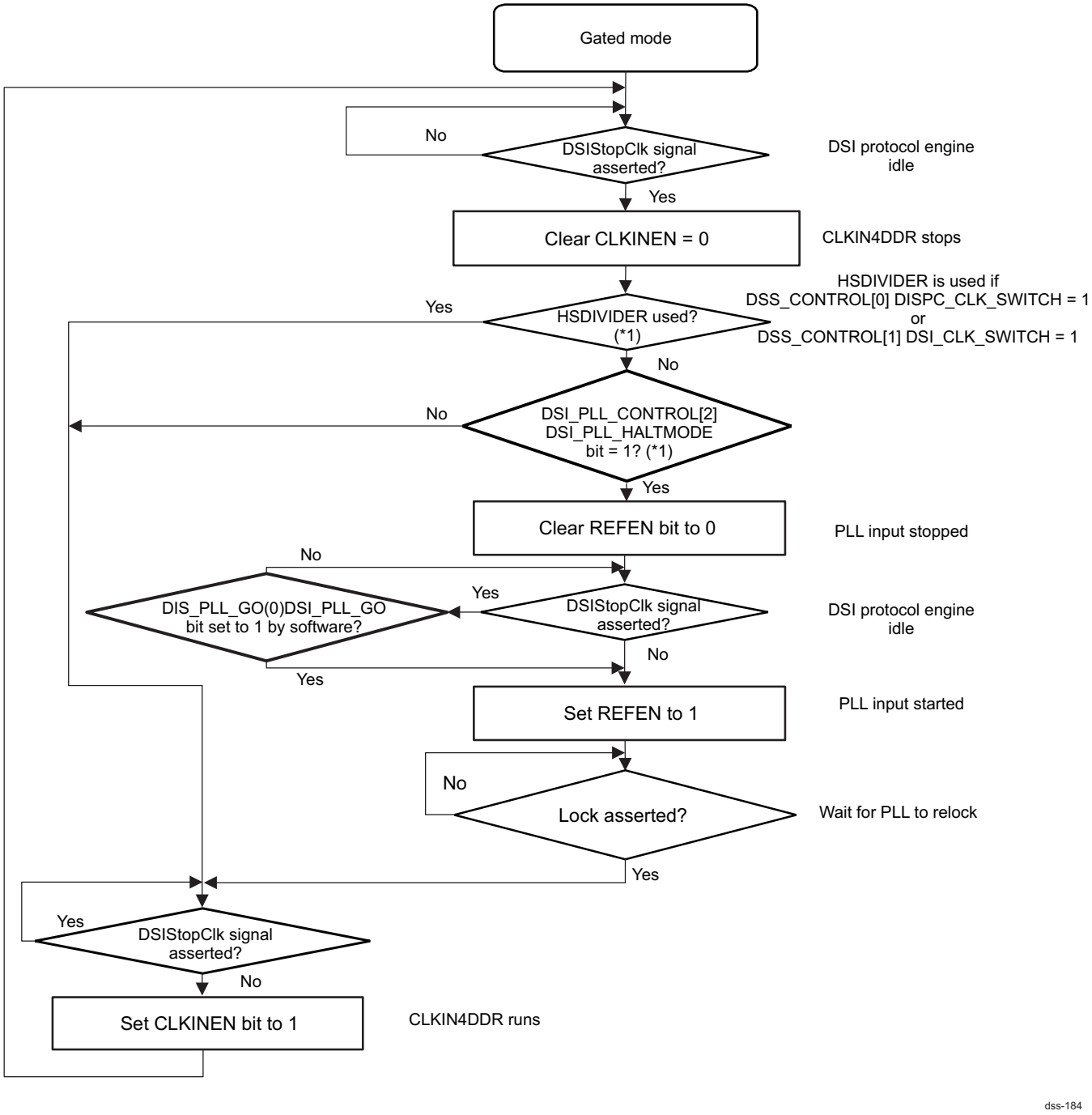
12.5.5.4 DSI PLL Clock Gating Sequence

Clock gating may be used to reduce system power consumption when the DSI protocol engine indicates that it does not need the clock. If the HSDIVIDER is not used, then the PLL can also be stopped (at the cost of additional unstarting latency).

The DSI protocol engine can verify when the PLL has unstarted by inspecting the LOCK signal (DSS.DSI_PLL_STATUS[1] DSI_PLL_LOCK status bit). Since TxByteClkHS is stopped when CLKIN4DDR is stopped, this should obviate the need for any explicit feedback that the clock has been unstarted in the other case. This flow chart should run even if the DSS.DSI_PLL_GO[0] DSI_PLL_GO bit has not been set.

Figure 12-142 shows the DSI PLL gated mode sequence.

Figure 12-142. Gated Mode Sequence



NOTE: All thick-outlined blocks show operations performed by software.

12.5.5.5 DSI PLL Lock Sequence

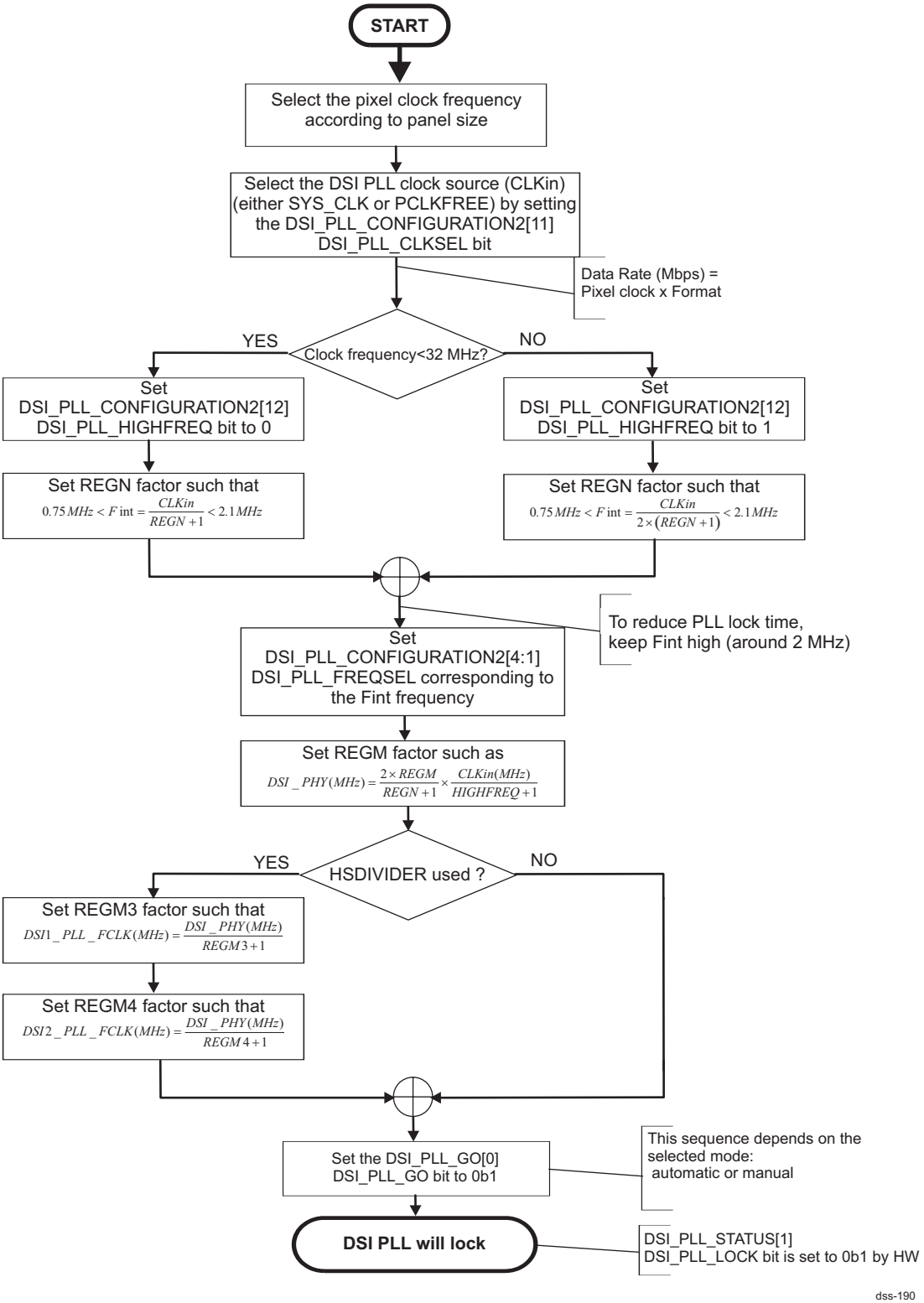
The DSI PLL (ADPLLv2) generates CLKIN4DDR. The HSDIVIDER generates two clocks: DSI1_PLL_FCLK connected to the display controller (DISPC) and the DSI2_PLL_FCLK connected to the DSI protocol engine. If these two clocks are not used, the HSDIVIDER functions are not required.

CLKIN4DDR is twice the data rate and four times the DSI output clock frequency. The DSI PLL factors need to be calculated based on the required input and output frequencies, keeping the PLL internal reference frequency in the appropriate range:

- REGM factor is programed by DSS.DSI_PLL_CONFIGURATION1[18:8] DSI_PLL_REGM bit field
- REGN factor is programed by DSS.DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN bit field
- REGM3 factor is programed by DSS.DSI_PLL_CONFIGURATION1[22:19] DSI_CLOCK_DIV bit field
- REGM4 factor is programed by DSS.DSI_PLL_CONFIGURATION1[26:23] DSIPROTO_CLOCK_DIV bit field

Figure 12-143 shows the programming sequence.

Figure 12-143. DSI PLL Programming Sequence



NOTE:

- The REGM3 and REGM4 factors must according with the following conditions:
  - The DSI1_PLL_FCLK and DSI2_PLL_FCLK frequencies must be a multiple of the PCLK frequency (for proper settings of PCD and LCD factors in the DISPC)
  - The DSI1_PLL_FCLK and DSI2_PLL_FCLK frequencies must be lower than 173 MHz at nominal voltage (OPP3), and lower than 96 MHz at low voltage (OPP2).
- Most of the other DSI PLL programming values are available for software flexibility but it is not recommended to update the values in normal use. See Section 12.5.5.7 for details on DSI PLL recommended values.

DSI PLL programming examples:

- WVGA Display on one data pair: Pixel clock (PCLK) = 30 MHz with 18-BPP pixel format

The data rate is 30 x 18 = 540 Mbps on one data lane. Therefore, the frequency on the data lane is twice the data rate: 1080 MHz.

The frequency on the clock lane is 270 MHz (1080 divided by 4).

The SYS_CLK at 26 MHz is selected as the clock reference by setting the DSS.DSI_PLL_CONFIGURATION2[11] DSI_PLL_CLKSEL to 0b0.

Set the DSS.DSI_PLL_CONFIGURATION2[12] DSI_PLL_HIGHFREQ to 0b0 as PCLK is lower than 32 MHz.

Set Fint to 2 MHz as PLL internal reference frequency: Set REGN to 12 (divide by 13) by setting the DSS.DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN bit field to 0xC and set DSI_PLL_CONFIGURATION2[4:1] DSI_PLL_FREQSEL bit field to 0x7 (1.75 to 2.1 MHz range)

To get CLKIN4DDR to 1080 MHz, set the REGM factor to 270 by setting the DSS.DSI_PLL_CONFIGURATION1[18:8] DSI_PLL_REGM to 0x10E.

CLKIN4DDR = 2 x 270/13 x 26/1 = 1080 MHz

Because DSI1_PLL_FCLK and DSI2_PLL_FCLK (REGM3 and REGM4 factors) must be multiples of PCLK and also lower than 173 MHz at nominal voltage (OPP3), and lower than 96 MHz at low voltage (OPP2), program these frequencies to 90 MHz by setting the REGM3 and REGM4 factors to 11 (divide by 12). This is done by setting the DSS.DSI_PLL_CONFIGURATION1[22:19] DSS_CLOCK_DIV bit field and DSS.DSI_PLL_CONFIGURATION1[26:23] DSIPROTO_CLOCK_DIV to 0xB:

DSI1_PLL_FCLK = DSI2_PLL_FCLK = 1080/12 = 90 MHz

- XGA Display on two data pairs: Pixel clock (PCLK) = 60 MHz with 16-BPP pixel format

  The data rate is (60 x 16)/2 = 480 Mbps on each data lane. Therefore, the frequency on the data lane is twice the data rate: 960 MHz.

  The frequency on the clock lane is 240 MHz (960 divided by 4).

  The SYS_CLK at 26 MHz is selected as the clock reference by setting the DSS.DSI_PLL_CONFIGURATION2[11] DSI_PLL_CLKSEL bit to 0b0.

  Set the DSS.DSI_PLL_CONFIGURATION2[12] DSI_PLL_HIGHFREQ bit to 0b0 as the source clock frequency (SYS_CLK in this example) is lower than 32 MHz.

  Set Fint to 2 MHz as PLL internal reference frequency: Set REGN to 12 (divide by 13) by setting the DSS.DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN bit field to 0xC and set DSI_PLL_CONFIGURATION2[4:1] DSI_PLL_FREQSEL bit field to 0x7 (1.75 to 2.1 MHz range)

  To get CLKIN4DDR to 960 MHz, set the REGM factor to 240 by setting the DSS.DSI_PLL_CONFIGURATION1[18:8] DSI_PLL_REGM to 0x0F0.

  CLKIN4DDR = 2 x 240/13 x 26/1 = 960 MHz

  Because DSI1_PLL_FCLK and DSI2_PLL_FCLK (REGM3 and REGM4 factors) must be multiples of PCLK and also lower than 173 MHz at nominal voltage (OPP3), program these frequencies to 120 MHz by setting the REGM3 and REGM4 factors to 7 (divide by 8). This is done by setting the DSS.DSI_PLL_CONFIGURATION1[22:19] DSS_CLOCK_DIV bit field and

DSS.DSI_PLL_CONFIGURATION1[26:23] DSIPROTO_CLOCK_DIV to 0x7:

DSI1_PLL_FCLK = DSI2_PLL_FCLK = 960/8 = 120 MHz

12.5.5.6 DSI PLL Error Handling

The PLL lock and recalibration signals may be monitored to detect loss of lock or requirement to recalibrate (due to large temperature change since the last lock request):

- The DSS.DSI_PLL_STATUS[1] DSI_PLL_LOCK status bit gives the DSI PLL lock state.
- The DSS.DSI_PLL_STATUS[2] DSI_PLL_RECAL status bit informs if the PLL must be uncalibrated

These signals can also generate interrupts at DSI protocol engine level:

-
  - DSS.DSI_IRQSTATUS[9], PLL_RECAL_IRQ bit
  - DSS.DSI_IRQSTATUS[8] PLL_UNLOCK_IRQ
  - DSS.DSI_IRQSTATUS[7] PLL_LOCK_IRQ
- The PLL_LOCK_IRQ interrupt indicates that the DSI PLL is locked. To monitor this event, read the DSS.DSI_IRQSTATUS[7] PLL_LOCK_IRQ bit. Set this bit to 1 to clear the status bit.
- The PLL_UNLOCK_IRQ interrupt indicates that the DSI PLL is unlocked. To monitor this event, read the DSS.DSI_IRQSTATUS[8] PLL_UNLOCK_IRQ bit. Set this bit to 1 to clear the status bit.
- The PLL_RECAL_IRQ interrupt indicates that the DSI PLL must be recalibrated. To monitor this event, read the DSS.DSI_IRQSTATUS[9] PLL_RECAL_IRQ bit. Set this bit to 1 to clear the status bit.

The PLL reference loss and limp status signals can also be monitored :

- The DSS.DSI_PLL_STATUS[3] DSI_PLL_LOSSREF status bit informs if the DSI PLL has lost the reference.
- The DSS.DSI_PLL_STATUS[4] DSI_PLL_LIMP status bit informs about the DSI PLL limp status.

12.5.5.7 DSI PLL Recommended Values

Table 12-66 shows the DSI PLL recommended values.

Table 12-66. Recommended Programming Values

Field Name	Value	Description
DSI_HSDIV_SYSRESET	0	Allow power FSM to control
DSI_PLL_SYSRESET	0	Allow power FSM to control
DSI_PLL_HALTMODE	-	See Section 12.5.5.4 for details
DSI_PLL_GATEMODE	-	See Section 12.5.5.4 for details
DSI_PLL_AUTOMODE	-	See Section 12.5.5.4 for details
DSI_PLL_GO	1->0	Write a 1 when PLL is to be (re-)locked with new parameters. This bit is cleared by hardware when the PLL request has completed
DSIPROTO_CLOCK_DIV	See (1)	DSI protocol engine clock divider
DSS_CLOCK_DIV	See (1)	DSS clock divider
DSI_PLL_REGM	See (1)	Feedback clock divider
DSI_PLL_REGN	See (1)	Reference clock divider
DSI_PLL_STOPMODE	1	Required to use GATEMODE bit
DSI_HSDIVBYPASS	0	PLL is controlling HSDIVIDER bypass
DSI_PROTO_CLOCK_PWDN	0	If PLL/HSDIVIDER is used as the DSI protocol clock source
DSI_PROTO_CLOCK_EN	1	If PLL/HSDIVIDER is used as the DSI protocol clock source

(1) The bit field value should be set according to the desired clock frequency.

Table 12-66. Recommended Programming Values (continued)

Field Name	Value	Description
DSS_CLOCK_PWDN	0	If PLL/HSDIVIDER is used as the DSS clock source
DSS_CLOCK_EN	1	If PLL/HSDIVIDER is used as the DSS clock source
DSI_BYPASSEN	0	To use PLL as the clock source. For small displays it may be possible to use the DSS functional clock, in which case this bit should be set to 1
DSI_PHY_CLKINEN	1	Enable CLKIN4DDR
DSI_PLL_REFEN	1	Enable PLL reference
DSI_PLL_HIGHFREQ	0/1	Set to 1 if the clock reference is higher than 32 MHz (21 MHz if DSS.DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN = 0)
DSI_PLL_CLKSEL	0/1	Set to 0 to use DSS2_ALWON_FCLK as the PLL reference or set to 1 to use PCLKFREE as the PLL reference, in this case the DISPC must be using DSS1_ALWON_FCLK.
DSI_PLL_LOCKSEL	0x0	Phase lock criteria to lock the PLL
DSI_PLL_DRIFTGUARDEN	0x0	The RECAL status/interrupt should be used to decide when to perform a PLL uncalibration No automatic uncalibration will be performed
DSI_PLL_TIGHTPHASELOCK	0	Normal criteria
DSI_LOWCURRSTDBY	0/1	Set to 0 for fast PLL unlock, but higher standby current Set to 1 for leakage level standby current, but longer unlock time
DSI_PLL_PLLLPMODE	0	Normal operation For smaller display sizes may be possible to set to 1
DSI_PLL_FREQSEL	See (2)	Must be set according to the PLL internal reference frequency (after N divider) See register descriptions for values
DSI_PLL_IDLE	0	PLL active

(2) The bit field value should be set according to the desired clock frequency.

12.5.6 DSI Complex I/O Basic Programming Model

12.5.6.1 Software Reset

The clock domain using the TxByteClkHS from the DSI complex I/O has dedicated reset done information in the DSS.DSI_COMPLEXIO_CFG1[29] RESET_DONE bit. The DSS.DSI_SYSCONFIG[1] SOFT_RESET bit is used to reset the TxByteClkHS power domain. A dummy read using the SCP interface to any DSI_PHY register is required after DSI_PHY reset to complete the reset of the DSI complex I/O.

12.5.6.2 Reset-Done Bits

The DSI complex I/O has several clock domains. The reset status for each clock domain is provided in DSS.DSI_PHY_CFG5 register:

- DSS.DSI_PHY_CFG5[31] RESETDONETXBYTECLK bit : Reset done for the TXBYTECLK domain.
- DSS.DSI_PHY_CFG5[28:26] RESETDONETXCLKESCi bits: Reset done for the TXCLKESC domain for lane i (i between 0 and 2).
- DSS.DSI_PHY_CFG5[30] RESETDONESCPCLK bit: Reset done for the SCP clock domain. Software

users must perform a dummy read on this bit to initiate the reset sequence of the SCP finite state machine. When the reset sequence is complete, the RESETDONESCPCLK signal goes high and software users can read again the DSS.DSI_PHY_CFG5[30] RESETDONESCPCLK bit to ensure that the value is now 1.

NOTE: The software should not write in the DSI_PHY_SCP registers before the DSS.DSI_PHY_CFG5[30] RESETDONESCPCLK bit is set to 1.

- DSS.DSI_PHY_CFG5[29] RESETDONEPWRCLK bit: Reset done for the PWR clock domain. The reset sequence of the PWR finite state machine is complete when the RESETDONEPWRCLK signal goes high.

12.5.6.3 Pad Configuration

The number of lanes is configurable through the DSS.DSI_COMPLEXIO_CFG1 register.

It is not allowed to change on the fly the position (by modifying the DATAi_POSITION with i=1 or 2 and CLOCK_POSITION bit fields), P/N order (Positive/Negative order of the differential pair by modifying the DATAi_POL with i=1 or 2 and CLOCK_POL) or number of active data lanes (by modifying the DSS.DSI_COMPLEXIO_CFG1[10:8] DATA2_POSITION bit). To add or remove the lane #2, it is required to be in OFF mode for the DSI complex I/O.

The minimum requirement for the number of lanes is one clock lane and one data lane. Note that by default, the data lane 2 is not connected (the DSS.DSI_COMPLEXIO_CFG1[10:8] DATA2_POSITION bit reset value is 0).

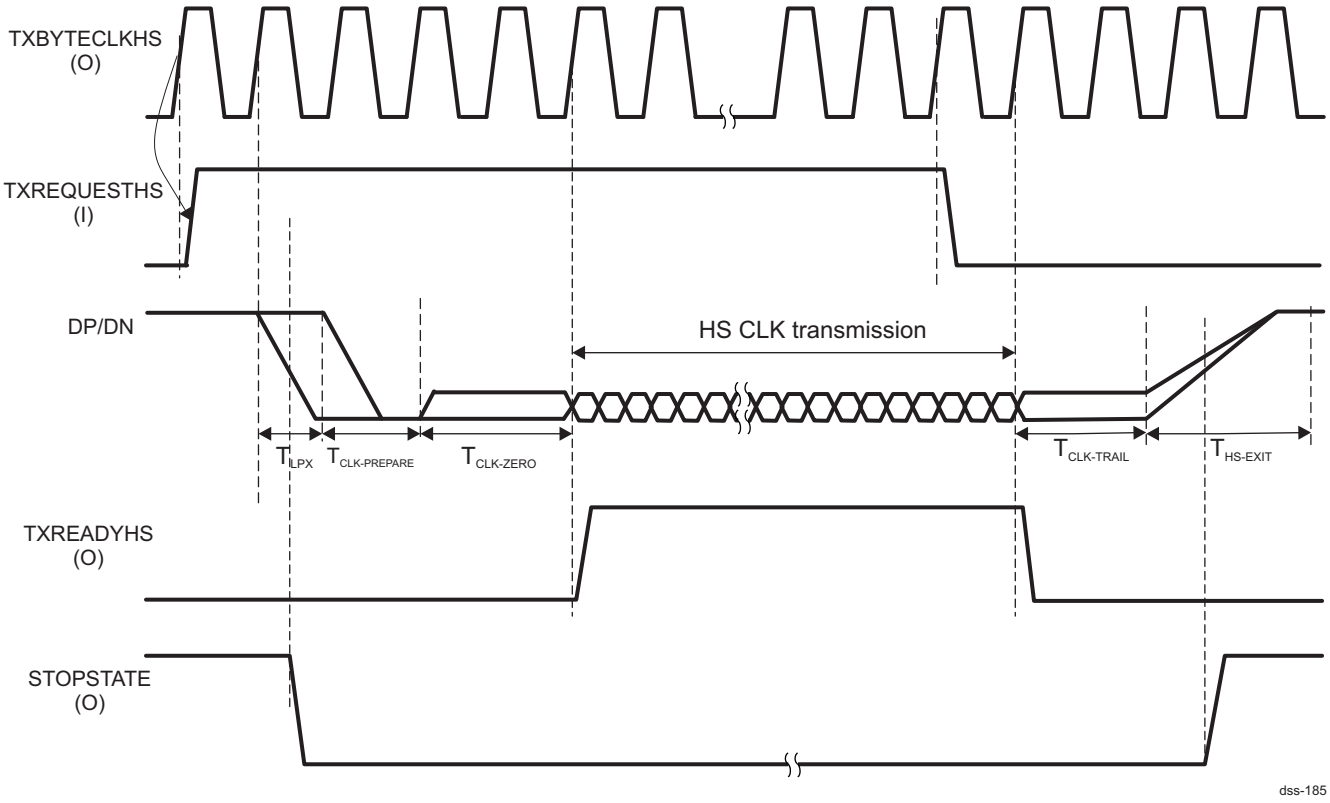
12.5.6.4 Display Timing Configuration

Depending on the CLKIN4DDR frequency settings programmed with the DSI PLL control module, software users must program accordingly the timing parameters in the DSI complex I/O registers.

12.5.6.4.1 High-Speed Clock Transmission

Figure 12-144 shows an example of high-speed Clock Transmission.

Figure 12-144. High-Speed Clock Transmission



TXByteClkHS is an output clock which is derived by dividing the CLKIN4DDR clock by 16.

To begin transmission, the protocol drives TXREQUESTHS high on a rising edge of TXByteClkHS. The PHY detects this signal on the next rising edge, following which it initiates the LP Start of Transmission (SoT) procedure.

During a high-speed Clock Transmission, these parameters are defined in multiples of CLKIN4DDR and programmed by the following register bit fields:

- TLPX timing is programmed by the DSS.DSI_PHY_CFG1[20:16] TLPX_HALF bit field.
- THS-PREPARE timing is programmed by the DSS.DSI_PHY_CFG0[31:24] THS_PREPARE bit field.
- TCLK-ZERO timing is programmed by the DSS.DSI_PHY_CFG1[7:0] TCLK_ZERO bit field.

. TCLK-ZERO is extended, if required, so that the entire LP SoT procedure lasts an integer number of TXByteClkHS cycles.

At the end of the SoT procedure, HS clock transmission begins. At the same time, TXREADYHS is made high.

To stop clock transmission, the protocol drives TXREQUESTHS low on a rising edge of TXByteClkHS. The DSI_PHY detects this change in TXREQUESTHS on the next edge and stops clock transmission. TXREADYHS is made low.

The DSI_PHY then goes through the LP End of Transmission (EoT) procedure. TCLK-TRAIL and THS-EXIT parameters are also multiples of CLKIN4DDR and programmed by the following register fields:

- TCLK-TRAIL timing is programmed by the DSS.DSI_PHY_CFG1[15:8] TCLK_TRAIL bit field.
- THS-EXIT timing is programmed by the DSS.DSI_PHY_CFG0[7:0] THS_EXIT bit field.

The DSI_PHY completes the SoT and EoT procedures, once begun, irrespective of any change in PPI signals. If TXREQUESTHS goes low during the SoT procedure, the PHY start the EoT procedure immediately after finishing the SoT procedure and no clock is transmitted.

STOPSTATE is high whenever the line is in LP-11 state, as determined by the outputs of the Low Power Receivers. This signal is not synchronized with TXByteClkHS.

It is requires that the high speed clock be present for some time before (TCLK-PRE) and some time after (TCLK-POST) high speed data transmission. The protocol must ensure that these timings are met by asserting and deasserting TXREQUESTHS appropriately.

The PHY ensures that the clock signal has a quadrature-phase with respect to a toggling bit sequence on any Data Lane, and a rising edge in the center of the first transmitted bit of every Data byte. These relations are not described in the timing diagram.

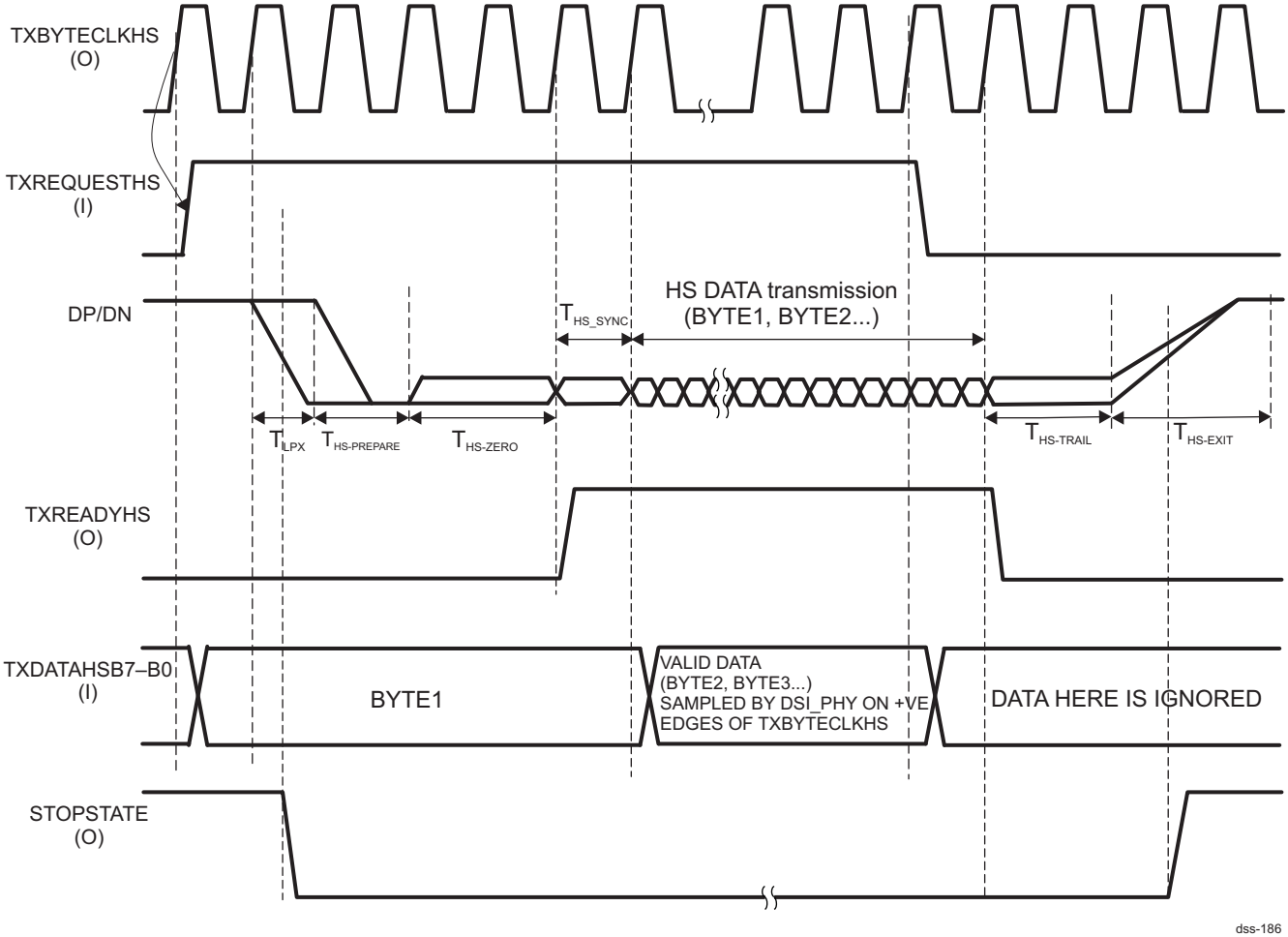
CLKIN4DDR can be shut off 300ns after the clock lane goes to STOPSTATE. Alternatively, CLKIN4DDR can be shut down after TCLK-Trail + THS-Exit + 2 Txbyteclk periods after TxRequestHS falling edge is received by DSI_PHY.

The DSI protocol engine should ensure that TXREQUESTESC, TXULPSCLK and TURNREQUEST are low whenever TXREQUESTHS is asserted.

12.5.6.4.2 High-Speed Data Transmission

Figure 12-145 shows an example of high-speed Data Transmission.

Figure 12-145. High-Speed Data Transmission



TXByteClkHS is an output clock which is derived by dividing CLKIN4DDR by 16.

To begin transmission, the protocol drives TXDATAHS with the first byte of data on a rising edge of TXByteClkHS. It also makes TXREQUESTHS high the same rising edge. The PHY detects TXREQUESTHS going high on the next rising edge of TXByteClkHS, following which it initiates the LP Start of Transmission (SoT) procedure.

During a high-speed Data Transmission, these timings are multiple of CLKIN4DDR and programmed by the following register bit fields:

- TLPX timing is programmed by the DSS.DSI_PHY_CFG1[20:16] TLPX_HALF bit field.
- THS-PREPARE + THS-ZERO timing is programmed by the DSS.DSI_PHY_CFG0[23:16] THS_PREPARE_THS_ZERO bit field.

THS-ZERO will be extended, if required, so that the entire LP SoT procedure lasts an integer number of TXByteClkHS cycles. THS-SYNC corresponds to the length of the sync pattern which is 8 high-speed bits, and can be configured through the tDSI_PHY_CFG2[31:24] HS_SYNC bit field.

Towards the end of the SoT procedure, the PHY makes TXREADYHS high on a positive edge of TXByteClkHS and then start accepting data from TXDATAHS from the next positive edge onwards. The protocol is expected to provide (new) valid data on TXDATAHS on every positive edge of TXByteClkHS if TXREADYHS is high.

At the end of the SoT procedure, HS data transmission begins. HS Data Transmission happens LSB first.

To stop data transmission, the protocol drives TXREQUESTHS low on a rising edge of TXByteClkHS. The PHY detects this change in TXREQUESTHS on the next edge and stops data transmission. TXREADYHS is made low and data on TXDATAHS, from that point, is ignored.

The PHY then goes through the LP End of Transmission (EoT) procedure. THS-TRAIL and THS-EXIT are also multiples of CLKIN4DDR and programmed by the following register bit fields:

- THS-TRAIL timing is programmed by the DSS.DSI_PHY_CFG0[15:8] THS_TRAIL bit field.
- THS-EXIT timing is programmed by the DSS.DSI_PHY_CFG0[7:0] THS_EXIT bit field.

The PHY completes the SoT and EoT procedures, once begun, irrespective of any change in PPI signals. If TXREQUESTHS goes low during the SoT procedure, the PHY start the EoT procedure immediately after finishing the SoT procedure and no data is transmitted.

STOPSTATE is high whenever the line is in LP-11 state, as determined by the outputs of the Low Power Receivers. This signal is not synchronized with TXByteClkHS.

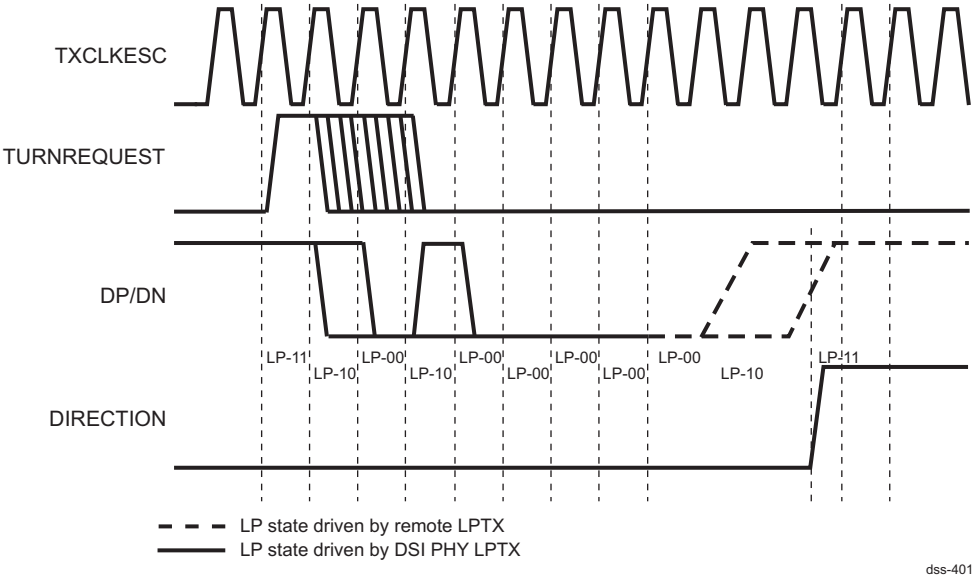
The Protocol should ensure that TXREQUESTESC, TXULPSCLK and TURNREQUEST are low whenever TXREQUESTHS is asserted.

12.5.6.4.3 Turn-Around Request in Transmit Mode

When the DSI PHY is in transmit mode, the DSI protocol engine can request a turnaround by making the TurnRequest signal high for at least one clock cycle of TxClkEsc (see Section 12.4.3.7.3, *TurnRequest FSM*).

The DSI PHY transmits the turn-around request pattern (LP 11-10-00-10-00-00-00-00) (see Figure 12-146). The number of 00 states at the end of the pattern is defined by the $T_{TA\text{-}GO}$ timing parameter and is programmable through the DSS.DSI_PHY_CFG1[31:29] TTA_GO bit field, in number of TxClkEsc clocks. Following the transmission of the pattern, the DSI PHY disables its LP transmitters and waits for an acknowledgement from the remote device. The remote device detects the turn-around request and acknowledges it by driving LP-10, followed by the STOP state. When this acknowledgment is received (BTA_IRQ is asserted, as described in Section 12.4.3.8, *Bus Turnaround*), the DSI PHY switches to receive mode and indicates the completion of the turn-around procedure by changing the direction (BTA_EN = 0).

Figure 12-146. Turn-Around Request in Transmit Mode

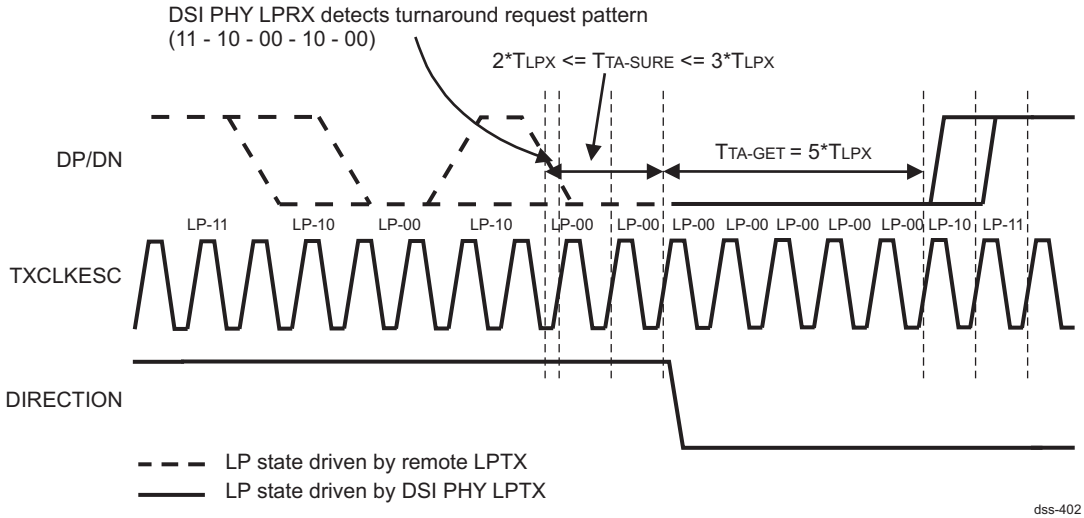


The DSI protocol engine must not stop TxClkEsc after the turn-around process completes (the DSS.DSI_CLK_CTRL[20] LP_CLK_ENABLE bit must be kept at 1), because TxClkEsc is also used in handling a turn-around request transmitted by a remote slave device (see Section 12.5.6.4.4, *Turn-Around Request in Receive Mode*.

12.5.6.4.4 Turn-Around Request in Receive Mode

When the DSI PHY is in the receive mode, an LP pattern of 11-10-00-10-00 on DP/DN lines indicates a turn-around request from the remote device (see Figure 12-147).

Figure 12-147. Turn-Around Request in Receive Mode



If the line stays in LP-00 for a time $T_{TA-SURE}$, the DSI PHY accepts the turn-around request, changes the direction, transmits LP-00 for a time T_{TA-GET}, and then transmits the acknowledgment pattern LP-10, followed by the STOP state.

This completes the turn-around procedure (BTA_IRQ is asserted, as described in Section 12.4.3.8, *Bus Turnaround*). The $T_{TA-SURE}$ and T_{TA-GET} timing parameters are programmable through DSS.DSI_PHY_CFG1 in number of TxClkEsc clocks:

- $T_{TA-SURE}$ can be configured in the [28:27] TTA_SURE bit field.

- T_{TA-GET} can be configured in the [26:24] TTA_GET bit field.

12.5.6.4.5 Other DSI_PHY Transmission and Reception

The timing of the following sequences defined in the DSI_PHY protocol cannot be programed by users:

- Low-power data transmission
- Escape mode trigger command transmission
- ULP state command transmission on data lanes
- ULP state transmission on clock lane
- Low-power data in receive mode
- Low-power trigger in receive mode
- ULP state command on clock lane in receive mode
- ULP state command on data lane in receive mode

12.5.6.5 Error Handling

A dedicated register for the DSI complex I/O, DSS.DSI_COMPLEXIO_IRQSTATUS, indicates the state of each error provided by the DSI complex I/O error signals. The DSI_PHY reports the following errors:

- DSS.DSI_COMPLEXIO_IRQSTATUS[7:5] ERR_ESCi_IRQ: ERRESC is asserted if an unrecognized Escape entry command is received. This remains high until the next change in the line state.
- DSS.DSI_COMPLEXIO_IRQSTATUS[2:0] ERRSYNCESCi_IRQ: If the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends, ERRSYNCESC is made high and remains high until the next change in line state.
- DSS.DSI_COMPLEXIO_IRQSTATUS[12:10] ERRCONTROLi_IRQ: ERRCONTROL is asserted if an incorrect line state sequence is detected. For example, if a turn-around request or escape mode request is immediately followed by a Stop state instead of the required Bridge state, this signal is asserted and remains asserted until the next change in the line state.
- DSS.DSI_COMPLEXIO_IRQSTATUS ERRCONTENTIONLP0_i_IRQ: ERRCONTENTION0LPDX and ERRCONTENTION0LPDY are asserted when the Lane module detects a contention situation on lines DX and DY respectively while trying to drive the lines low. Contention is detected only if it lasts at least 50ns
- DSS.DSI_COMPLEXIO_IRQSTATUS ERRCONTENTIONLP1_i_IRQ: ERRCONTENTION1LPDX and ERRCONTENTION1LPDY are asserted when the Lane module detects a contention situation on lines DX and DY respectively while trying to drive the lines high. Contention is detected only if it lasts at least 50ns

The ULPSACTIVENOT signal goes low which indicates to the protocol that the PHY has entered ULP state. When all the ULPSActiveNot signals are low, the DSS.DSI_COMPLEXIO_IRQSTATUS[30] ULPSACTIVENOT_ALL0_IRQ event is generated. When all the ULPSActiveNot signals are high, the DSS.DSI_COMPLEXIO_IRQSTATUS[31] ULPSACTIVENOT_ALL1_IRQ event is generated.

When any of the events defined in DSS.DSI_COMPLEXIO_IRQSTATUS register happened, the DSS.DSI_IRQSTATUS[10] COMPLEXIO_ERR_IRQ bit is set to 1 at DSI protocol engine level.

The software must take appropriate action when receiving the interrupt indicating the error from the complex I/O. The action can be:

- Reset of the DSI protocol engine module
- Reset of the peripheral though reset trigger or directly driving the hardware reset pin of the display module
- Ignore the error

12.5.7 RFBI Basic Programming Model

The RFBI programming model must be used for LCD display support only.

12.5.7.1 DISPC Control Registers

The following DISPC registers are used in RFBI mode:

- The STALL mode is selected by setting the DSS.DISPC_CONTROL[11] STALLMODE bit. The DSS.DISPC_CONTROL[5] GOLCD bit should not be set to 1 but the display controller configuration (DMA engine, pipelines associated to the LCD output,..) should be set before enabling the LCD output by setting the DSS.DISPC_CONTROL[0] LCDENABLE bit to 1.
- To enable the hardware handcheck to avoid underflow, the DSS.DISPC_CONTROL[16] FIFOHANDCHECK should be set to 1. The reset value of this bit is 0. The handcheck applies to the pipelines connected to the LCD output. It should be disabled before resetting the DSS.DISPC_CONTROL[11] STALLMODE bit to 0. The new setting for the FIFO handcheck is used for the following frames.

NOTE: The LCD output is disabled at the end of the transfer of the frame. The software must reenable the LCD output to generate a new frame by setting the DSS.DISPC_CONTROL[0] LCDENABLE to 1. See Figure 12-148.

12.5.7.2 RFBI Control Registers

The following registers define the RFBI control registers:

- DSS.RFBI_CONTROL
- DSS.RFBI_PIXEL_CNT
- DSS.RFBI_LINE_NUMBER

12.5.7.2.1 High Threshold

The DSS.RFBI_CONTROL[6:5] HIGHTHRESHOLD bit field is used to define the threshold to be used for the generation of the DMA request to receive data into the interconnect FIFO (24 × 32 FIFO depth) through the address of the register RFBI_DATA. It should be the size of the burst. The supported values are 4x32, 8x32 and 16x32. The system DMA receives the DMA request and is in charge of providing the correct number of bytes. If the DSS.RFBI_CONTROL[7] DISABLE_DMA_REQ bit is reset, the DMA request is generated when there is enough room in the interconnect FIFO to accept the full burst. In case the RFBI receives writes L4 requests to the RFBI_DATA location when the interconnect FIFO is full, the request is not accepted. The RFBI waits for a free entry in the interconnect FIFO to accept the L4 request.

If the DSS.RFBI_CONTROL[7] DISABLE_DMA_REQ bit is set, the DMA request is not generated. The threshold value is ignored.

NOTE: Software users can access the RFBI_DATA location without using the DMA request and without programming the high threshold value (backward mode).

12.5.7.2.2 Bypass Mode

Setting the DSS.RFBI_CONTROL[1] BYPASSMODE bit directly outputs the LCD controller output to the LCD panel. Resetting this bit directs the MPU module to send commands/parameters and data from the input video port FIFO.

12.5.7.2.3 Enable

Setting/resetting the DSS.RFBI_CONTROL[0] ENABLE bit enables/disables the RFBI module. The hardware resets the enable bit after all of the pixels are sent to the panel. The DSS.RFBI_PIXEL_CNT[31:0] PIXELCNT bit field value defines the number of pixels to send to the LCD panel. When the transfer is finished, the configuration used can be modified.

Table 12-67. RFBI Behavior

RFBI_CONTROL[1] BYPASSMODE bit value	RFBI_CONTROL[0] ENABLE bit value	RFBI Behavior
0	0	L4 interconnect can write command/param/data and read data/status from the Remote Frame Buffer (RFB). L4 interconnect access can only be done to the CSx actually active
0	1	The DISPC sends pixels to the RFB.

The stall signal is asserted when the module is disabled. Through the L4 port, pixels can be sent to the LCD panel only when the pixel count has reach the value 0x0

NOTE: The LCD output is disabled at the end of the transfer of the frame. The software must reenable the LCD output to generate a new frame by setting the DSS.DISPC_CONTROL[0] LCDENABLE to 1. See Figure 12-148.

12.5.7.2.4 Configuration Selection

Setting the DSS.RFBI_CONTROL[3:2] CONFIGSELECT bit field selects the configuration number (1 or 0 if bits are set or reset). The registers associated with the configuration output the data to the LCD panel.

If both chip-selects are selected, the configuration for the first chip-select is used (except for the polarity of the RFBI_CS1 signal defined by the second configuration) and both devices connected to the CS signals are driven in parallel. In read mode, if both chip-selects are set, only RFBI_CS0 is asserted to read data from the device connected on RFBI_CS0. In write mode with two chip-selects selected, the RFBI can write to the two devices simultaneously.

12.5.7.2.5 ITE Bit

Set the DSS.RFBI_CONTROL[4] ITE bit to start capturing the data from the display controller. This bit has no effect if the trigger mode is set to external. The display controller must be configured in the STALL mode to account for the RFBI_DISPC_STALL signal. Setting the trigger mode to external (DSS.RFBI_CONFIGi[3:2] TRIGGERMODE bit field set to 0x1 or 0x2) causes the DSS.RFBI_CONTROL[4] ITE bit to be ignored. The corresponding chip-select must be selected when this bit is set by users.

The RFBI_DISPC_STALL signal is asserted when at least one of the following cases occur:

- Default status when no data to capture from the display controller
- High FIFO threshold reached
- End of the transfer (number of data to output)
- Reset of the RFBI module
- DSS.RFBI_CONTROL[0] ENABLE bit reset to 0x0

The RFBI_DISPC_STALL signal is deasserted when the DSS.RFBI_CONTROL[0] ENABLE bit is set to 0x1 and at least one of the following cases occur:

- Low FIFO threshold reached
- External TE occurs and the DSS.RFBI_CONFIGi[3:2] TRIGGERMODE bit field is set to 0x1 or 0x2 for automatic external trigger (start of the transfer, the FIFO pointers are reset, the FIFO is empty).
- DSS.RFBI_CONTROL[4] ITE bit set to 0x1 by users (start of the transfer, the FIFO pointers are reset, the FIFO is empty).

12.5.7.2.6 Number of Pixels to Transfer

Setting the DSS.RFBI_PIXEL_CNT[31:0] PIXELCNT bit field value directs the application to indicate the number of pixels to be transferred to the LCD panel. The value can be changed only when the DSS.RFBI_CONTROL[0] ENABLE is reset.

During the transfer, the hardware decrements the register when a pixel is sent to the remote frame buffer. When the DSS.RFBI_CONTROL[0] ENABLE bit is set and a new value is written in the DSS.RFBI_PIXEL_CNT register when the current value in the register is a non-zero (the remaining number of pixels to transfer), the ongoing transfer is aborted.

From the L4 interconnect side, if DSS.RFBI_CONFIGi[10:9] CYCLEFORMAT bit field is equal to 0x3 and DSS.RFBI_CONFIGi[8:7] L4FORMAT is equal to 0x0, an even number of write accesses to the data register should be performed before accessing any other register (CMD/PARAM/STATUS/READ).

When DSS.RFBI_CONFIGi[10:9] CYCLEFORMAT bit field is 0x3 - meaning that 2 pixels are sent over 3 cycles -, the number of pixels to be programmed in the DSS.RFBI_PIXEL_CNT[31:0] PIXELCNT bit field should be a multiple of 2. If another CYCLEFORMAT is used, the value for PIXELCNT can be odd or even. This constraint is valid for data provided on the L4 interconnect port and from the display controller.

If the DSS.RFBI_CONFIGi[10:9] CYCLEFORMAT bit field is equal to 0x3, DSS.RFBI_CONFIGi[8:7] L4FORMAT is equal to 0, and back to back register write is processed, the following registers should be written after the first data: RFBI_CMD, RFBI_PARAM, RFBI_READ, and RFBI_STATUS. The whole data transfer should first be performed before being able to write to any other registers (RFBI_CMD, RFBI_PARAM, RFBI_READ, and RFBI_STATUS).

12.5.7.2.7 Programmable Line Number

When the trigger mode is set to external trigger mode with HSYNC and VSYNC or the TE, the hardware resets the line counter when the VSYNC occurs and, after a programmable number of lines (the HSYNC pulse occurs for every line), the transfer to the LCD panel begins. When the programmable line number is 0, only the VSYNC pulse indicates the beginning of the transfer in both modes: HSYNC/VSYNC and TE (logical OR operation between HSYNC and VSYNC).

12.5.7.3 RFBI Configuration

The following registers define the RFBI configuration:

- DSS.RFBI_SYSCONFIG
- DSS.RFBI_SYSSTATUS
- DSS.RFBI_CONFIG0 (configuration 0) and DSS.RFBI_CONFIG1 (configuration 1)
- DSS.RFBI_VSYNC_WIDTH
- DSS.RFBI_HSYNC_WIDTH

The configuration register for one configuration can be accessed only when the configuration is not in use (based on the value of the RFBI_CONTROL[3:2] CONFIGSELECT bit field).

12.5.7.3.1 Parallel Mode

The DSS.RFBI_CONFIGi[1:0]PARALLELMODE bit field (with i = 0, 1) defines the width of the interface (8-, 9-, 12-, or 16-bit parallel).

12.5.7.3.2 Trigger Mode

Setting the DSS.RFBI_CONFIG[3:2]TRIGGERMODE bit field configures the trigger on the external TE signal (RFBI_TE_VSYNC), or external with VSYNC/HSYNC with the programmable number of HSYNCs to begin the transfer in both cases or the internal programmable DSS.RFBI_CONTROL[4] ITE bit.

12.5.7.3.3 VSYNC Pulse Width (Minimum Value)

The DSS.RFBI_VSYNC_WIDTH[15:0] MINVSYNCPULSEWIDTH bit field defines the minimum number of L4 clock cycles of the VSYNC pulse for detection on VSYNC. It allows differentiation between VSYNC and HSYNC, which are ORed on the same signal and is also used in the VSYNC/HSYNC mode on the two separate input lines.

- The VSYNC pulse width must be at least equal to two L4 cycles when HSYNC is not present.
- The VSYNC pulse width must be at least equal to four L4 cycles when HSYNC is present.

12.5.7.3.4 HSYNC Pulse Width (Minimum Value)

The DSS.RFBI_HSYNC_WIDTH[15:0] MINHSYNCPULSEWIDTH bit field defines the minimum number of L4 clock cycles of the HSYNC pulse for detection on HSYNC. It allows differentiation between VSYNC and HSYNC, which are ORed on the same signal, and is also used in the VSYNC/HSYNC mode on the separate two input lines. The HSYNC pulse width must always be at least equal to two L4 cycles to be detected.

12.5.7.3.5 Cycle Format

Setting the DSS.RFBI_CONFIGi[10:9] CYCLEFORMAT bit field (with i = 0, 1) defines which registers are used to format the data in the interconnect FIFO with the appropriate number of bits (starting from the LSB) and with the alignment on the interface as follows:

- DSS.RFBI_DATA_CYCLE_i (if DSS.RFBI_CONFIG[10:9] CYCLEFORMAT bit field = 00) only

  or
- DSS.RFBI_DATA_CYCLE1_i and DSS.RFBI_DATA_CYCLE2_i (if DSS.RFBI_CONFIG[10:9] CYCLEFORMAT bit field = 01)

  or
- DSS.RFBI_DATA_CYCLE1_i, DSS.RFBI_DATA_CYCLE2_i, and DSS.RFBI_DATA_CYCLE3_i (if DSS.RFBI_CONFIG[10:9] CYCLEFORMAT bit field = 10)

The data from the display controller and from the L4 interconnect are formatted based on the configuration of the DSS.RFBI_DATA_CYCLE_i registers.

12.5.7.3.6 Unused Bits

Based on the configuration, the undefined bits for each cycle are defined with the previous values of the bits at the same position in the previous cycle, 0s, or 1s (the unused bits can be at any position). The DSS.RFBI_CONFIGi[12:11] UNUSEDBITS bit field (with i = 0, 1) is used.

12.5.7.3.7 RFBI Timings

The timing registers for one configuration can be accessed only when the configuration is not in use (based on the value of the DSS.RFBI_CONTROL[3:2] CONFIGSELECT bit field). Granularity is defined using the DSS.RFBI_CONFIGi[4] TIMEGRANULARITY bit. This feature allows the extension of programmable ranges of timing parameters for the RFBI interface. Refer to Table 12-68 for the bits configuration values.

- Chip-select assertion/deassertion time

  RFBI_A0 setup time to chip-select assertion is assured by the programmable chip-select assertion time from the start access time:
  DSS.RFBI_ONOFF_TIMEi[3:0] CSONTIME bit field (with i = 0, 1).

  The chip-select deassertion time from the start access time is programmable:
  DSS.RFBI_ONOFF_TIMEi[9:4] CSOFFTIME bit field (with i = 0, 1)

> **CAUTION**
>
> Configuring DSS.RFBI_ONOFF_TIMEi[3:0] CSONTIME = DSS.RFBI_ONOFF_TIMEi[9:4] CSOFFTIME = 0 (with i = 0, 1) is not supported and must be avoided. This configuration creates contention on the bus and progressively damages the LCD panel.

- Chip-select pulse width

  The total chip-select pulse width is the time when write cycle time or read cycle time has completed and is programmable:
  DSS.RFBI_CYCLE_TIMEi[17:12] CSPULSEWIDTH bit field (with i = 0, 1)

  It applies on the read-to-write, write-to-read, read-to-read, and write-to-write access based on:

  - The DSS.RFBI_CYCLE_TIMEi [19] RRENABLE bit: Read-to-read access
  - The DSS.RFBI_CYCLE_TIMEi [20] WWENABLE bit: Write-to-write access
  - The DSS.RFBI_CYCLE_TIMEi [18] RWENABLE bit: Read-to-write access
  - The DSS.RFBI_CYCLE_TIMEi [21] WRENABLE bit: Write-to-read access

    By default, it applies to any access (read-to-read, read-to-write, write-to-read, write-to-write) when the chip-select changes.

- Access time

  The total access time is the time from when A0 becomes valid until data are sampled before deasserting the RE signal; access time is programmable:
  DSS.RFBI_CYCLE_TIMEi[27:22] ACCESSTIME bit field (with i = 0, 1)

  When reading the data on the bus, the data are sampled at the end of the access time, which occurs before the end of the read off time (DSS.RFBI_ONOFF_TIMEi[29:24] REOFFTIME, with i = 0, 1).

- Write enable cycle time

  The total write enable cycle time is the time from when A0 becomes valid until write cycle completion; the write enable cycle time is programmable:
  The DSS.RFBI_CYCLE_TIMEi[5:0] WECYCLETIME bit field (with i = 0, 1)

- Write enable assertion/deassertion time

  The WE assertion delay time from start access time is programmable:
  DSS.RFBI_ONOFF_TIMEi[13:10] WEONTIME bit field (with i = 0, 1)

  The WE deassertion delay time from the start access time is programmable:
  DSS.RFBI_ONOFF_TIMEi[19:14] WEOFFTIME bit field (with i = 0, 1)

- Read enable cycle

  The total read enable cycle time is the time when A0 becomes valid until read cycle completion; the read enable cycle time is programmable:
  The DSS.RFBI_CYCLE_TIMEi[11:6] RECYCLETIME bit field (with i = 0, 1)

- Read enable assertion/deassertion time

  The RE assertion delay time from the start access time is programmable:
  DSS.RFBI_ONOFF_TIMEi[23:20] REONTIME bit field (with i = 0, 1)

  The RE deassertion delay time from the start access time is programmable:
  DSS.RFBI_ONOFF_TIMEi[29:24] REOFFTIME bit field (with i = 0, 1)

At cycle time completion (read access or write access) all control signals (RFBI_CSi, RFBI_WR, and RFBI_RD, with i = 0, 1) are deasserted regardless of their deassertion time parameter values, if they are not deasserted already.

However, an exception to this forced deassertion exists when a pipelined request to the same chip-select or to a different chip-select is pending. Also, a control signal with deassertion time parameters equal to the cycle time parameter is not necessarily deasserted when a pipelined request to the same chip-select or different chip-select is pending. This prevents any unnecessary glitch transitions.

If no inactive cycles are required between successive accesses to the same chip-select (the DSS.RFBI_CYCLE_TIMEi[17:12] CSPULSEWIDTH bit field = 0, with i = 0, 1), and if assertion time parameters associated with the following access equal 0, the asserted control signals (RFBI_CSi, RFBI_WR, and RFBI_RD, with i = 0, 1) stay asserted. This is applicable to any read/write-to-read/write access combination.

Table 12-68 summarizes the configurations values for each timing bit.

Table 12-68. RFBI Timings Configuration

Configuration bits[1]	Granularity[2]	
	one	two
DSS.RFBI_ONOFF_TIMEi[3:0] CSONTIME	0 to 15	0 to 30

(1) Where i = 0 or 1.
(2) Number of L4Clk cycles. The granularity can be configured using the DSS.RFBI_CONFIGi[4] TIMEGRANULARITY bit.

Table 12-68. RFBI Timings Configuration (continued)

Configuration bits[1]	Granularity[2]	
	one	two
DSS.RFBI_ONOFF_TIMEi[9:4] CSOFFTIME	0 to 63	0 to 126
DSS.RFBI_CYCLE_TIMEi[17:12] CSPULSEWIDTH	0 to 63	0 to 126
DSS.RFBI_CYCLE_TIMEi[27:22] ACCESSTIME	0 to 63	0 to 126
DSS.RFBI_CYCLE_TIMEi[5:0] WECYCLETIME	0 to 63	0 to 126
DSS.RFBI_ONOFF_TIMEi[13:10] WEONTIME	0 to 15	0 to 30
DSS.RFBI_ONOFF_TIMEi[19:14] WEOFFTIME	0 to 63	0 to 126
DSS.RFBI_CYCLE_TIMEi[11:6] RECYCLETIME	0 to 63	0 to 126
DSS.RFBI_ONOFF_TIMEi[23:20] REONTIME	0 to 15	0 to 30
DSS.RFBI_ONOFF_TIMEi[29:24] REOFFTIME	0 to 63	0 to 126

12.5.7.3.8 RFBI State-Machine

Referring to Table 12-37, the signals RFBI_A0, RFBI_RD, and RFBI_WR are asserted/deasserted based on the register accessed (DSS.RFBI_CMD, DSS.RFBI_PARAM, DSS.RFBI_DATA, DSS.RFBI_READ, and DSS.RFBI_STATUS). When the DSS.RFBI_SYSSTATUS[8] BUSY bit is set by hardware, any access to the registers is stalled, except for the RFBI_DATA register.

The DSS.RFBI_SYSSTATUS[9] BUSYRFBIDATA bit indicates whether there are still pending data in the interconnect FIFO associated with the register RFBI_DATA only.

- Command register

  Write a command at a time by writing in the DSS.RFBI_CMD register. If the previous command is not processed, the DSS.RFBI_SYSSTATUS[8] BUSY bit is set by hardware and the access to writing a new command is stalled.

- Parameter register

  Write a parameter at a time by writing in the DSS.RFBI_PARAM register.

  If the previous parameter is not processed, the DSS.RFBI_SYSSTATUS[8] BUSY bit is set by hardware and the access to writing a new parameter is stalled.

- Data register

  Write one or two pixels at a time by writing in the RFBI_DATA register (when DSS.RFBI_CONFIGi[10:9] CYCLEFORMAT = 0x3 with i = 0, 1, two pixels must be written contiguously, no other access to RFBI registers except DSS.RFBI_DATA is allowed).

  The pixels are formatted based on the specified cycle format. If two pixels are written into the 32-data register, the DSS.RFBI_CONFIGi[8:7] L4FORMAT bit field indicates the number of pixels for each L4 access to the register and the order of the pixels

  If the previous data are not processed, the DSS.RFBI_SYSSTATUS[8] BUSY bit is set by hardware and any access for writing new data is stalled. When the DSS.RFBI_SYSSTATUS[8] BUSY bit is reset by hardware, the access is not stalled.

- Read/status register

  Send through the command and parameter registers the correct information to receive data in the data or status register. The read data from the LCD panel is initiated by writing into the DSS.RFBI_READ or DSS.RFBI_STATUS registers. In this case, the DSS.RFBI_SYSSTATUS[8] BUSY bit is set until the data are available in the register.

  When the DSS.RFBI_SYSSTATUS[8] BUSY bit is set by hardware, the read or write access is stalled until the register is updated with a new value from the LCD panel. To avoid the stall, the software can poll the DSS.RFBI_SYSSTATUS[8] BUSY bit until it is reset by hardware. To receive the data, send the appropriate command/parameters.

12.5.7.3.9 RFBI Configuration Flow Charts

The RFBI configuration depends on the trigger mode used by the application. The available trigger modes are:

- Internal trigger mode when setting the DSS.RFBI_CONFIGi[3:2] TRIGGERMODE bit field to 0x0
- External trigger mode:
  - TE external trigger mode when setting the DSS.RFBI_CONFIGi[3:2] TRIGGERMODE bit field to 0x1
  - HSYNC/VSYNC external trigger mode when setting the DSS.RFBI_CONFIGi[3:2] TRIGGERMODE bit field to 0x2

Figure 12-148 gives an example of how to program and use the RFBI module:

Figure 12-148. How to Use RFBI

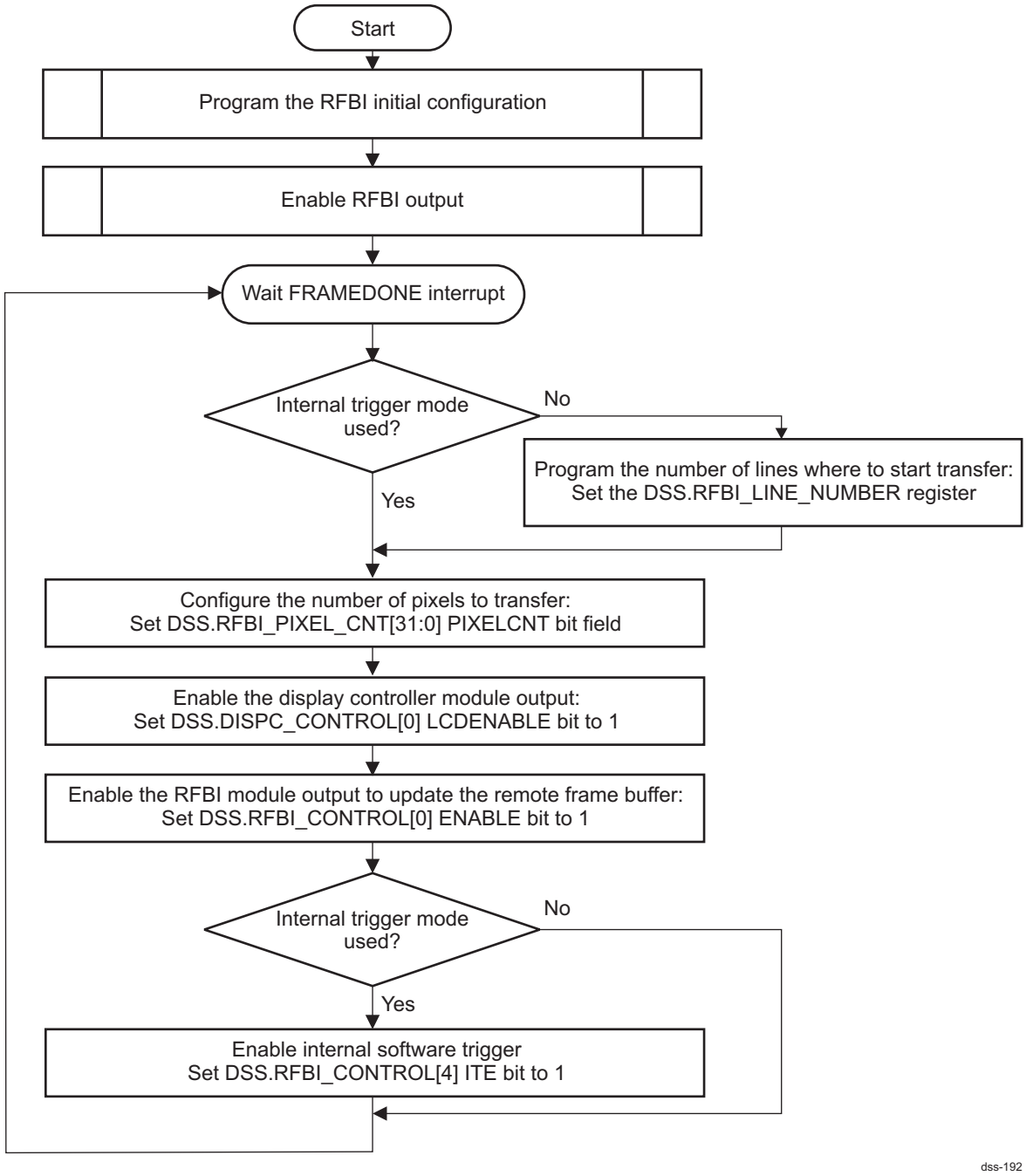


Figure 12-149 details how to configure the RFBI registers:

Figure 12-149. RFBI Initial Configuration

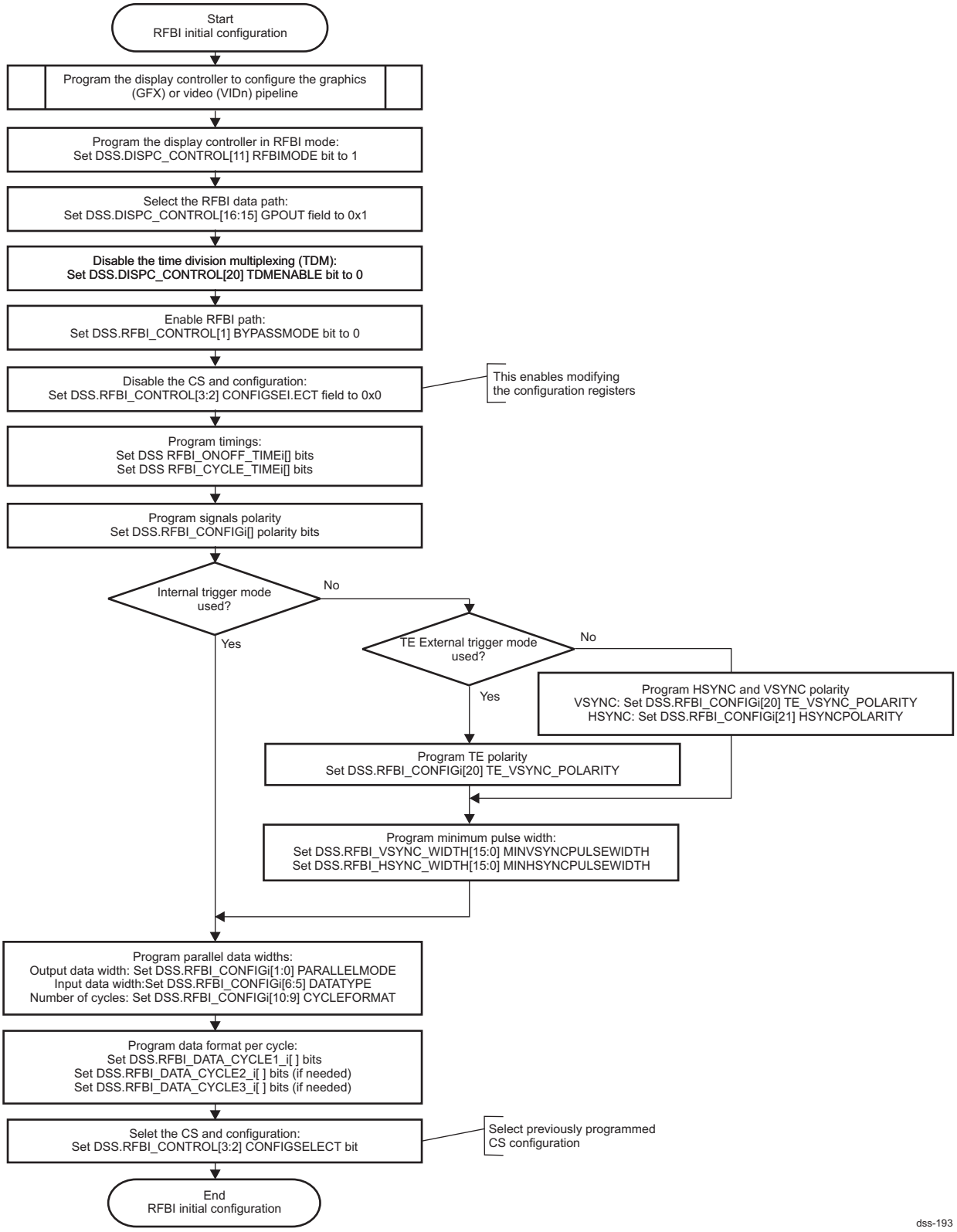
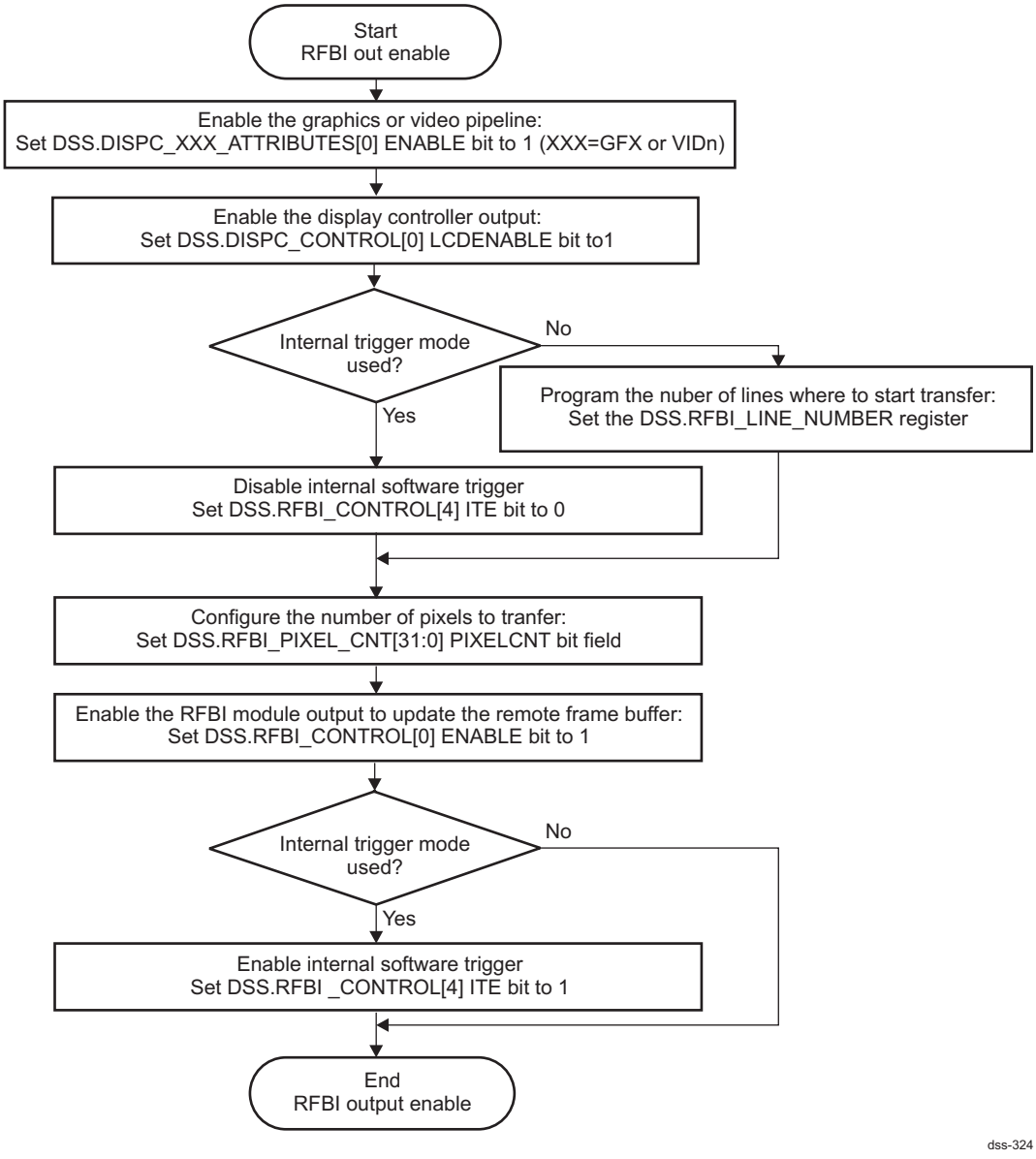


Figure 12-150 describes how to enable the RFBI module.

Figure 12-150. RFBI Output Enable



12.5.8 Video Encoder Basic Programming Model

12.5.8.1 Video Encoder Software Reset

By setting the DSS.VENC_F_CONTROL[8] RESET bit to 1, the video encoder is reset. This bit is automatically cleared by hardware when the reset is done.

NOTE: Before changing the standard (NTSC or PAL) and all the related registers, a software reset is required to properly initialize the VENC module.

12.5.8.2 Video DAC Settings

The video output format can be either:

- one composite video (CVBS) output signal with video DAC1 or

- two separated luma/chroma output signals with both video DAC1(luma analog signal) and DAC2 (chroma analog signal)

  Selection is performed with DSS.DSS_CONTROL[6] VENC_OUT_SEL bit:

  - When VENC_OUT_SEL bit is set to 0 (reset value), video DAC1 is selected for composite video (CVBS) signal
  - When VENC_OUT_SEL bit is set to 1, video DAC1 is selected for luma signal

  One of the first VENC settings is to enable the output of the video DACs. This setting depends on the video output format:

  - CVBS video output format: Enable video DAC1 composite output by setting the DSS.VENC_OUTPUT_CONTROL[1] COMPOSITE_ENABLE bit to 1
  - Separated luma/chroma output format: Enable video DAC1 luma output by setting the DSS.VENC_OUTPUT_CONTROL[0] LUMA_ENABLE bit to 1 and enable video DAC2 chroma output by setting the DSS.VENC_OUTPUT_CONTROL[2] CHROMA_ENABLE bit to 1

12.5.8.3 Video Encoder Programming Sequence

1. Set the DSS.VENC_F_CONTROL[8] RESET bit to 1 to perform a software reset of the VENC module.
2. Before any configuration change, save the DSS.DISPC_IRQENABLE register value (DSS interrupts context), and then disable the display subsystem interrupts by setting the DSS.DISPC_IRQENABLE register to 0x0000.
3. Configure the video encoder registers as described in Table 12-69, depending on the video standard used (PAL or NTSC). The DSS.VENC_F_CONTROL and DSS.VENC_SYNC_CTRL registers must be the last ones to be changed by software.
4. Set the DSS.DISPC_CONTROL[6] GODIGITAL bit and the DSS.DISPC_CONTROL[1] DIGITALENABLE bit to 1.
5. Wait for the first VSYNC pulse signal.
6. Clear the SYNCLOSTDIGITAL interrupt by setting the DSS.DISPC_IRQSTATUS[15] SYNCLOSTDIGITAL bit to 1.
7. Set the DSS.DISPC_IRQENABLE register to the value saved in step 2 (restore the DSS interrupts context).

12.5.8.4 Video Encoder Register Settings

For video encoder programming, see Table 12-69. This table lists the register values to use in standard applications. These values are validated programming values only for the TV display support (NTSC 601 and PAL 601 standards).

Table 12-69. Video Encoder Register Programming Values

Register Name	NTSC 601	PAL 601
VENC_F_CONTROL	0x00000000	0x00000000
VENC_VIDOUT_CTRL	0x00000001	0x00000001
VENC_SYNC_CTRL	0x00008040	0x00000040
VENC_LLEN	0x00000359	0x0000035F
VENC_FLENS	0x0000020C	0x00000270
VENC_HFLTR_CTRL	0x00000000	0x00000000
VENC_CC_CARR_WSS_CARR	0x043F2631	0x2F7225ED
VENC_C_PHASE	0x00000000	0x00000000
VENC_GAIN_U	0x00000102	0x00000111
VENC_GAIN_V	0x0000016C	0x00000181
VENC_GAIN_Y	0x0000012F	0x00000140
VENC_BLACK_LEVEL	0x00000043	0x0000003B
VENC_BLANK_LEVEL	0x00000038	0x0000003B
VENC_X_COLOR	0x00000007	0x00000007

Table 12-69. Video Encoder Register Programming Values (continued)

Register Name	NTSC 601	PAL 601
VENC_M_CONTROL	0x00000001	0x00000002
VENC_BSTAMP_WSS_DATA	0x00000038	0x0000003F
VENC_S_CARR	0x21F07C1F	0x2A098ACB
VENC_LINE21	0x00000000	0x00000000
VENC_LN_SEL	0x01310011	0x01290015
VENC_L21_WC_CTL	0x0000F003	0x0000F603
VENC_HTRIGGER_VTRIGGER	0x00000000	0x00000000
VENC_SAVID_EAVID	0x069300F4	0x06A70108
VENC_FLEN_FAL	0x0016020C	0x00180270
VENC_LAL_PHASE_RESET	0x00060107	0x00040135
VENC_HS_INT_START_STOP_X	0x008E0350	0x00880358
VENC_HS_EXT_START_STOP_X	0x000F0359	0x000F035F
VENC_VS_INT_START_X	0x01A00000	0x01A70000
VENC_VS_INT_STOP_X_VS_INT_START_Y	0x020701A0	0x000001A7
VENC_VS_INT_STOP_Y_VS_EXT_START_X	0x01AC0024	0x01AF0000
VENC_VS_EXT_STOP_X_VS_EXT_START_Y	0x020D01AC	0x000101AF
VENC_VS_EXT_STOP_Y	0x00000006	0x00000025
VENC_AVID_START_STOP_X	0x03480078	0x03530083
VENC_AVID_START_STOP_Y	0x02060024	0x026C002E
VENC_FID_INT_START_X_FID_INT_START_Y	0x0001008A	0x0001008A
VENC_FID_INT_OFFSET_Y_FID_EXT_START_X	0x01AC0106	0x002E0138
VENC_FID_EXT_START_Y_FID_EXT_OFFSET_Y	0x01060006	0x01380001
VENC_TVDETGP_INT_START_STOP_X	0x00140001	0x00140001
VENC_TVDETGP_INT_START_STOP_Y	0x00010001	0x00010001
VENC_GEN_CTRL	0x00F90000	0x00FF0000
VENC_OUTPUT_CONTROL	0x0000000A (composite video CVBS) 0x0000000D (split video S-video)	0x0000000A (composite video CVBS) 0x0000000D (split video S-video)
VENC_OUTPUT_TEST	0x00000000	0x00000000

NOTE: The following display controller registers must be programmed to the NTSC 601 video standard:

DSS.DISPC_SIZE_DIG[10:0] PPL = 720 - 1 = 719 = 0x2CF
DSS.DISPC_SIZE_DIG[26:16] LPP = (482/2) - 1 = 240 = 0xF0
DSS.DISPC_GFX_BA0 or DSS.DISPC_VIDn_BA0 = Base address of even bit field data
DSS.DISPC_GFX_BA1 or DSS.DISPC_VIDn_BA1 = Base address of odd bit field data

12.5.9 SDI Basic Programming Model

This section describes how to configure the SDI module for FlatLink3G-compliant LCD panel support.

Configure the SDI, and then configure the display controller.

12.5.9.1 SDI Configuration

NOTE: To use the SDI with the TI FlatLink3G display panel:

- Set the DSS.DISPC_CONTROL[3] STNTFT bit to 1 (active matrix display operation mode)
- Set the DSS.DISPC_CONTROL[9:8] TFTDATALINES bit field to 0x3 (24-bit data output)
- Set the DSS.DSS_SDI_CONTROL[1:0] SDI_BWSEL bit field to 0x2 (color depth is 24 bits)

12.5.9.1.1 SDI PLL Configuration

The programming of the PLL divisor must be consistent with the choice of the number of data pairs. The PLL divisor determines the multiplication factor from the pixel clock frequency to data rate.

Table 12-70 lists some example values of the PLL settings for TI FlatLink3G.

Table 12-70. PLL Divisor Example Values for TI FlatLink3G

		24-BPP data		
		1 data pair	2 data pairs	3 data pairs
Recommended values	Pixel clock frequency (MHz)	15	30	65
	DSS.DSS_PLL_CONTROL[16:11]			
	SDI_PLL_REGN bit field (N factor = SDI_PLL_REGN bit field value + 1)	0x7	0xE	0xF
	DSS.DSS_PLL_CONTROL[10:1]			
	SDI_PLL_REGM bit field (M factor)	0xF0	0xE1	0x140
	DDS.DSS_PLL_CONTROL[19]			
	SDI_PLL_HIGHFREQ	0	0	1
	Effective PLL divisor value (M/N)	30	15	10

For more details on SDI PLL programming settings, see Section 12.6.3.

12.5.9.1.2 Signal Features Configuration

For proper SDI operation, configure the following features in the display controller:

- Set the DSS.DISPC_POL_FREQ[16] RF bit to drive HSYNC and VSYNC on the rising edge of PCLK.
- Set the DSS.DISPC_POL_FREQ[17] ONOFF bit.
- Clear the DSS.DISPC_POL_FREQ[14] IPC bit to drive the pixel data on the rising edge of PCLK.
- Set the DSS.DISPC_CONTROL[29] LCDENABLEPOL bit to set the LCD-enable signal active high.

The following settings are not required by the SDI2 interface, but these polarities settings are the common polarities used by FlatLink3G displays:

- Set the DSS.DISPC_POL_FREQ[13] IHS bit to 1 to set HSYNC active low.
- Set the DSS.DISPC_POL_FREQ[12] IVS bit to 1 to set VSYNC active low.
- Clear the DSS.DISPC_POL_FREQ[15] IEO bit to set the data enable active high.

12.5.9.1.3 Number of Data Pairs

To select the number of data pairs used to transmit display data between the device and the display panel, set the DSS.DSS_SDI_CONTROL[3:2] SDI_PRSEL field.

Table 12-71 lists the different values for these bits based on the number of data pairs.

Table 12-71. SDI Pixel Data Format

	DSS.DSS_SDI_CONTROL[3:2] SDI_PRSEL
1 data pair (DATA1 pair)	00
2 data pairs (DATA1 and DATA2 pair)	01
3 data pairs (DATA1, DATA2, and DATA3 pair)	10

12.5.9.2 SDI Power-Management Programming Sequence

This section describes the software sequence for powering on and powering down the SDI module.

12.5.9.2.1 SDI Reset State

After DSS power domain reset or power-on reset, the SDI module is in low-power mode by default with the following configuration:

- Display SS pads multiplexing is in parallel mode and not in SDI mode:
  - CONTROL.CONTROL_PADCONF_DSS_DATA10[2:0]MUXMODE0 and CONTROL.CONTROL_PADCONF_DSS_DATA10[18:16]MUXMODE1 field values is 0x0: SDI_DATA1N/SDI_DATA1P signals are not mapped on dss_data10/dss_data11 pins.
  - CONTROL.CONTROL_PADCONF_DSS_DATA12[2:0]MUXMODE0 and CONTROL.CONTROL_PADCONF_DSS_DATA12[18:16]MUXMODE1 bit field values is 0x0: SDI_DATA2N/SDI_DATA2P signals are not mapped on dss_data12/dss_data13 pins.
  - CONTROL.CONTROL_PADCONF_DSS_DATA14[2:0]MUXMODE0 and CONTROL.CONTROL_PADCONF_DSS_DATA14[18:16]MUXMODE1 bit field values is 0x0: SDI_DATA3N/SDI_DATA3P signals are not mapped on dss_data14/dss_data15 pins.
  - CONTROL.CONTROL_PADCONF_DSS_DATA18[2:0]MUXMODE0 and CONTROL.CONTROL_PADCONF_DSS_DATA18[18:16]MUXMODE1 bit field values is 0x0: SDI_VSYNC/SDI_HSYNC signals are not mapped on dss_data18/dss_data19 pins.
  - CONTROL.CONTROL_PADCONF_DSS_DATA20[2:0]MUXMODE0 and CONTROL.CONTROL_PADCONF_DSS_DATA20[18:16]MUXMODE1 bit field values is 0x0: SDI_DEN/SDI_STP signals are not mapped on dss_data20/dss_data21 pins.
  - CONTROL.CONTROL_PADCONF_DSS_DATA22[2:0]MUXMODE0 and CONTROL.CONTROL_PADCONF_DSS_DATA22[18:16]MUXMODE1 bit field values is 0x0: SDI_CLKP/SDI_CLKN signals are not mapped on dss_data22/dss_data23 pins.
- SDI PLL is in reset mode: the DSS.DSS_PLL_CONTROL[18]SDI_PLL_SYSRESET bit value is 0x0.

12.5.9.2.2 SDI Power_On Sequence

When exiting OFF mode, to power-on the SDI module, software users must program the following sequence:

1. Program the SDI pads multiplexing configuration:
   (a) Set the CONTROL.CONTROL_PADCONF_DSS_DATA10[2:0]MUXMODE0 bit field to 0x1 to select SDI_DATA1N signal on dss_data10 pin and CONTROL.CONTROL_PADCONF_DSS_DATA10[18:16]MUXMODE1 bit field to 0x1 to select SDI_DATA1P signal on dss_data11 pin.
   (b) For 2-data and 3-data pair mode: set the CONTROL.CONTROL_PADCONF_DSS_DATA12[2:0]MUXMODE0 bit field to 0x1 to select SDI_DATA2N on dss_data12 pin and CONTROL.CONTROL_PADCONF_DSS_DATA12[18:16]MUXMODE1 bit field to 0x1 to select SDI_DATA2P signal on dss_data13 pin.
   (c) For 3-data pair mode: set the CONTROL.CONTROL_PADCONF_DSS_DATA14[2:0]MUXMODE0 bit field to 0x1 to select SDI_DATA3N on dss_data14 pin and CONTROL.CONTROL_PADCONF_DSS_DATA14[18:16]MUXMODE1 bit field to 0x1 to select SDI_DATA3P signal on dss_data15 pin.

(d) Set the CONTROL.CONTROL_PADCONF_DSS_DATA18[2:0]MUXMODE0 bit field to 0x1 to select SDI_VSYNC signal on dss_data18 pin and CONTROL.CONTROL_PADCONF_DSS_DATA18[18:16]MUXMODE1 bit field to 0x1 to select SDI_HSYNC signal on dss_data19 pin.

(e) Set the CONTROL.CONTROL_PADCONF_DSS_DATA20[2:0]MUXMODE0 bit field to 0x1 to select SDI_DEN signal on dss_data20 pin and CONTROL.CONTROL_PADCONF_DSS_DATA20[18:16]MUXMODE1 bit field to 0x1 to select SDI_STP signal on dss_data21 pin.

(f) Set the CONTROL.CONTROL_PADCONF_DSS_DATA22[2:0]MUXMODE0 bit field to 0x1 to select SDI_CLKP signal on dss_data22 pin and CONTROL.CONTROL_PADCONF_DSS_DATA22[18:16]MUXMODE1 bit field to 0x1 to select SDI_CLKN signal on dss_data23 pin.

2. Wait 1 ms.
3. Release the SDI PLL reset signal by setting the DSS.DSS_PLL_CONTROL[18]SDI_PLL_SYSRESET bit to 0x1.

NOTE: The same software sequence is needed when the DSS power domain is set to ON.

12.5.9.2.3 SDI Power-Down Sequence

To power down the SDI module, software users must program the following sequence:

1. Power down the SDI PLL by setting the DSS.DSS_PLL_CONTROL[18]SDI_PLL_SYSRESET bit to 0x0.
2. Power-down the SDI pads:

(a) Set the CONTROL.CONTROL_PADCONF_DSS_DATA10[2:0]MUXMODE0 bit field to 0x0 to un-select SDI_DATA1N signal on dss_data10 pin and CONTROL.CONTROL_PADCONF_DSS_DATA10[18:16]MUXMODE1 bit field to 0x0 to un-select SDI_DATA1P signal on dss_data11 pin.

(b) For 2-data and 3-data pair mode: set the CONTROL.CONTROL_PADCONF_DSS_DATA12[2:0]MUXMODE0 bit field to 0x0 to un-select SDI_DATA2N on dss_data12 pin and CONTROL.CONTROL_PADCONF_DSS_DATA12[18:16]MUXMODE1 bit field to 0x0 to un-select SDI_DATA2P signal on dss_data13 pin.

(c) For 3-data pair mode: set the CONTROL.CONTROL_PADCONF_DSS_DATA14[2:0]MUXMODE0 bit field to 0x0 to un-select SDI_DATA3N on dss_data14 pin and CONTROL.CONTROL_PADCONF_DSS_DATA14[18:16]MUXMODE1 bit field to 0x0 to un-select SDI_DATA3P signal on dss_data15 pin.

(d) Set the CONTROL.CONTROL_PADCONF_DSS_DATA18[2:0]MUXMODE0 bit field to 0x0 to un-select SDI_VSYNC signal on dss_data18 pin and CONTROL.CONTROL_PADCONF_DSS_DATA18[18:16]MUXMODE1 bit field to 0x0 to un-select SDI_HSYNC signal on dss_data19 pin.

(e) Set the CONTROL.CONTROL_PADCONF_DSS_DATA20[2:0]MUXMODE0 field to 0x0 to un-select SDI_DEN signal on dss_data20 pin and CONTROL.CONTROL_PADCONF_DSS_DATA20[18:16]MUXMODE1 bit field to 0x0 to un-select SDI_STP signal on dss_data21 pin.

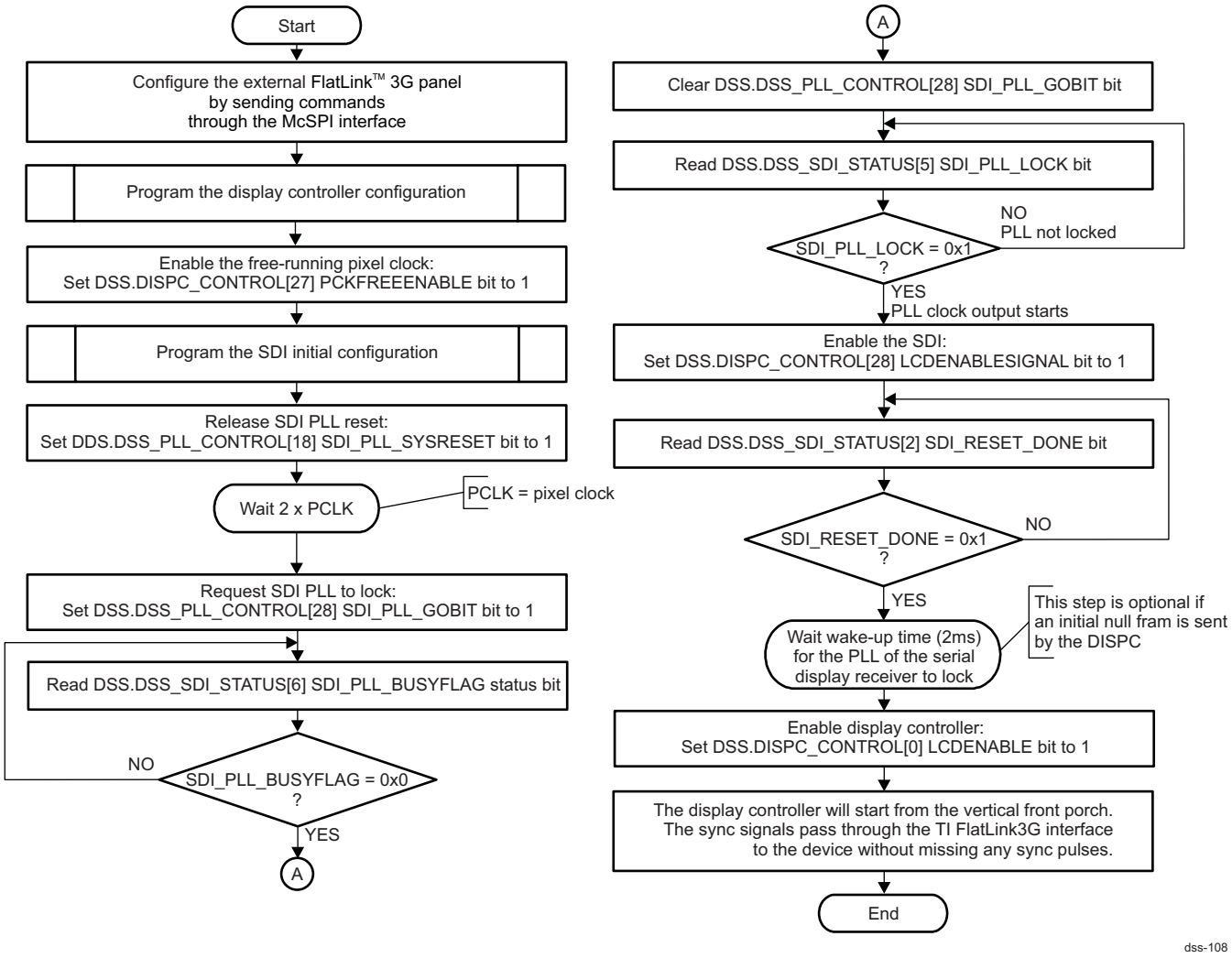
(f) Set the CONTROL.CONTROL_PADCONF_DSS_DATA22[2:0]MUXMODE0 bit field to 0x0 to un-select SDI_CLKP signal on dss_data22 pin and CONTROL.CONTROL_PADCONF_DSS_DATA22[18:16]MUXMODE1 bit field to 0x0 to un-select SDI_CLKN signal on dss_data23 pin.

NOTE: The same software sequence is needed to shut-down the DSS power domain.

12.5.9.3 SDI Start Sequence

Figure 12-151 describes the SDI start sequence.

Figure 12-151. SDI Start Sequence



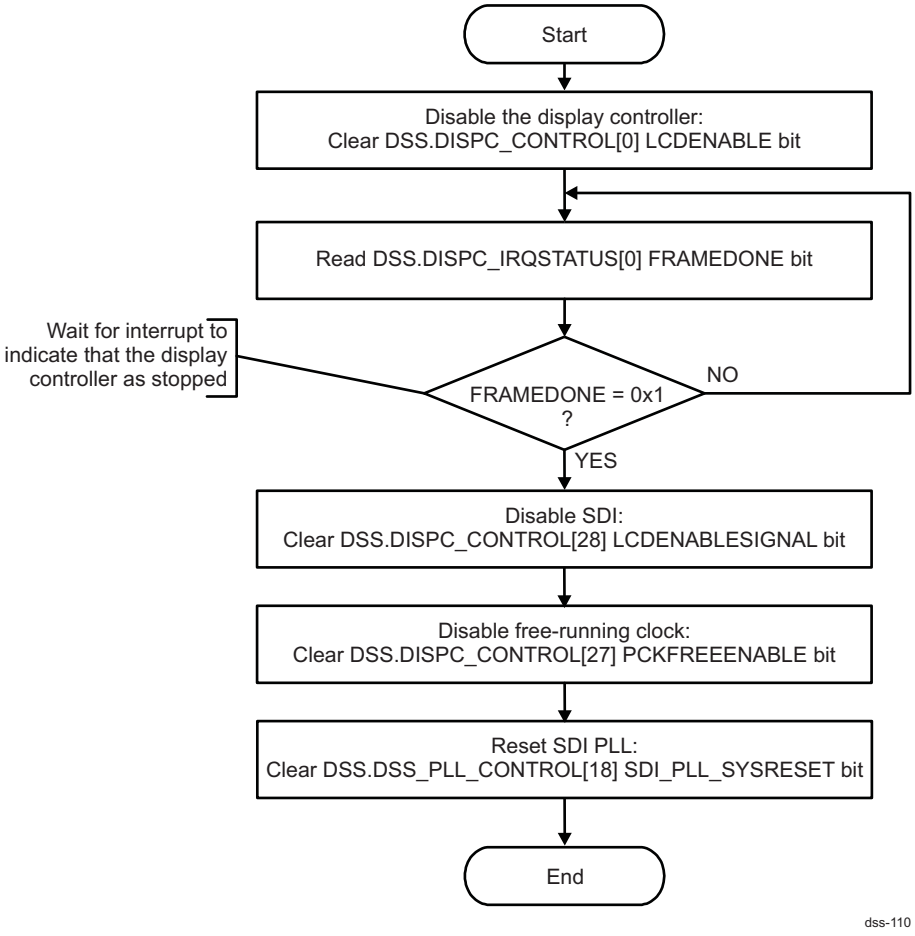
NOTE: The lock status of the SDI PLL is connected to the GPIO_81 signal of the GPIO3 module. To avoid polling, configure the GPIO3 module to generate an interrupt when the SDI PLL is locked (lock status is 1). For more information, see Chapter 21, *General-Purpose Interface*.

NOTE: TI provides a global solution with OMAP device associated to a programmable 27-bit serial display interface receiver, the SN65LVDS302 receiver. For more details on SN65LVDS302 device, see the TI public web at www.ti.com.

12.5.9.4 SDI Stop Sequence

Figure 12-152 describes the SDI stop sequence.

Figure 12-152. SDI Stop Sequence



12.5.9.5 Clock Source/Frequency Change Sequence

To change clock source or frequency, the simplest procedure is to follow the stop and start sequences documented above. If it is desired to change these without stopping the display a more rigorous procedure must be used as follows. The intent of this procedure is to perform the changes as quickly as possible during the vertical blanking interval and hence to minimize any visible display effect due to the increased blanking time.

12.5.9.5.1 Complete Sequence

Figure 12-153 and Figure 12-154 describe the clock source and frequency change sequence of the SDI.

Figure 12-153. SDI Clock Source/Frequency Change Sequence Part A

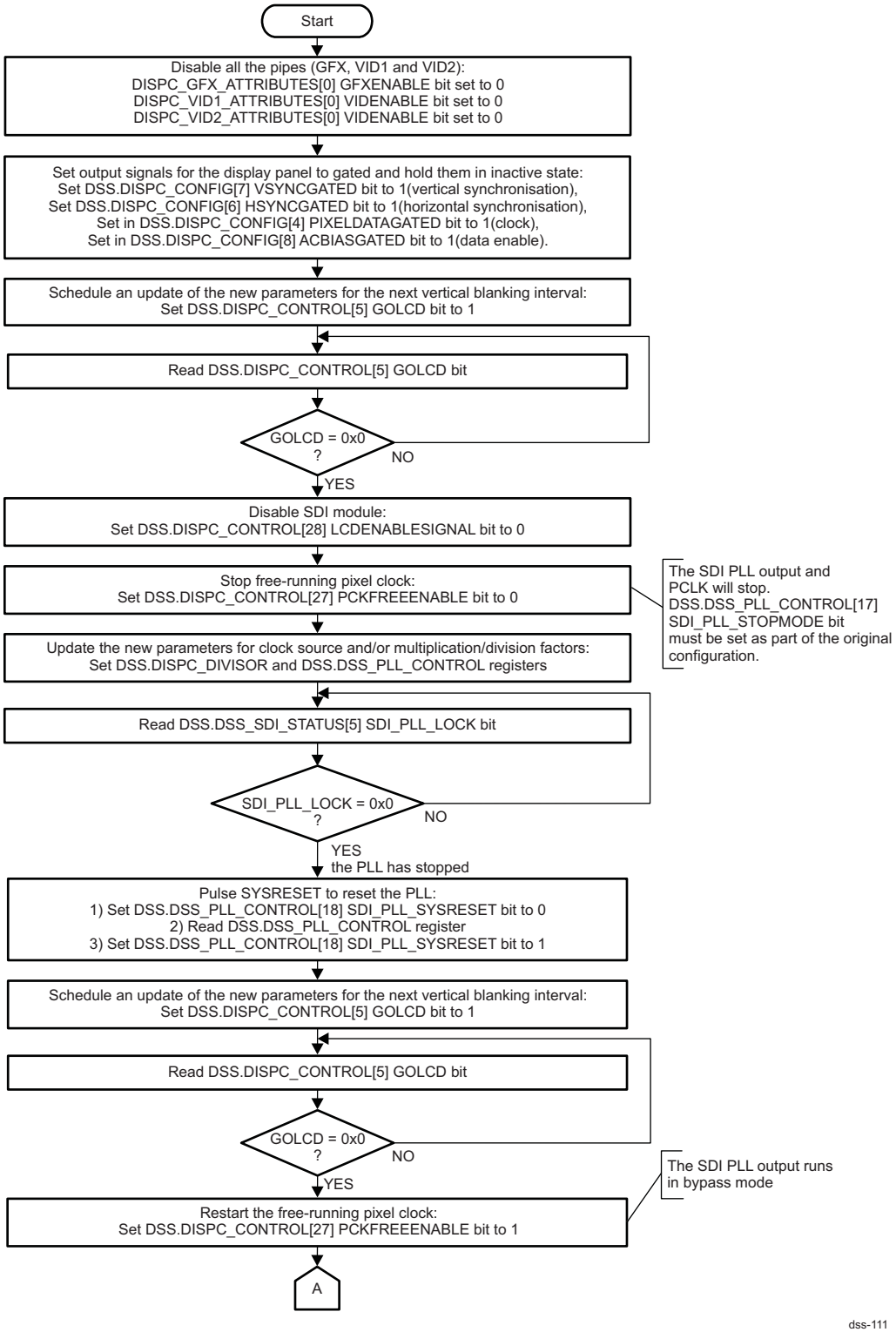
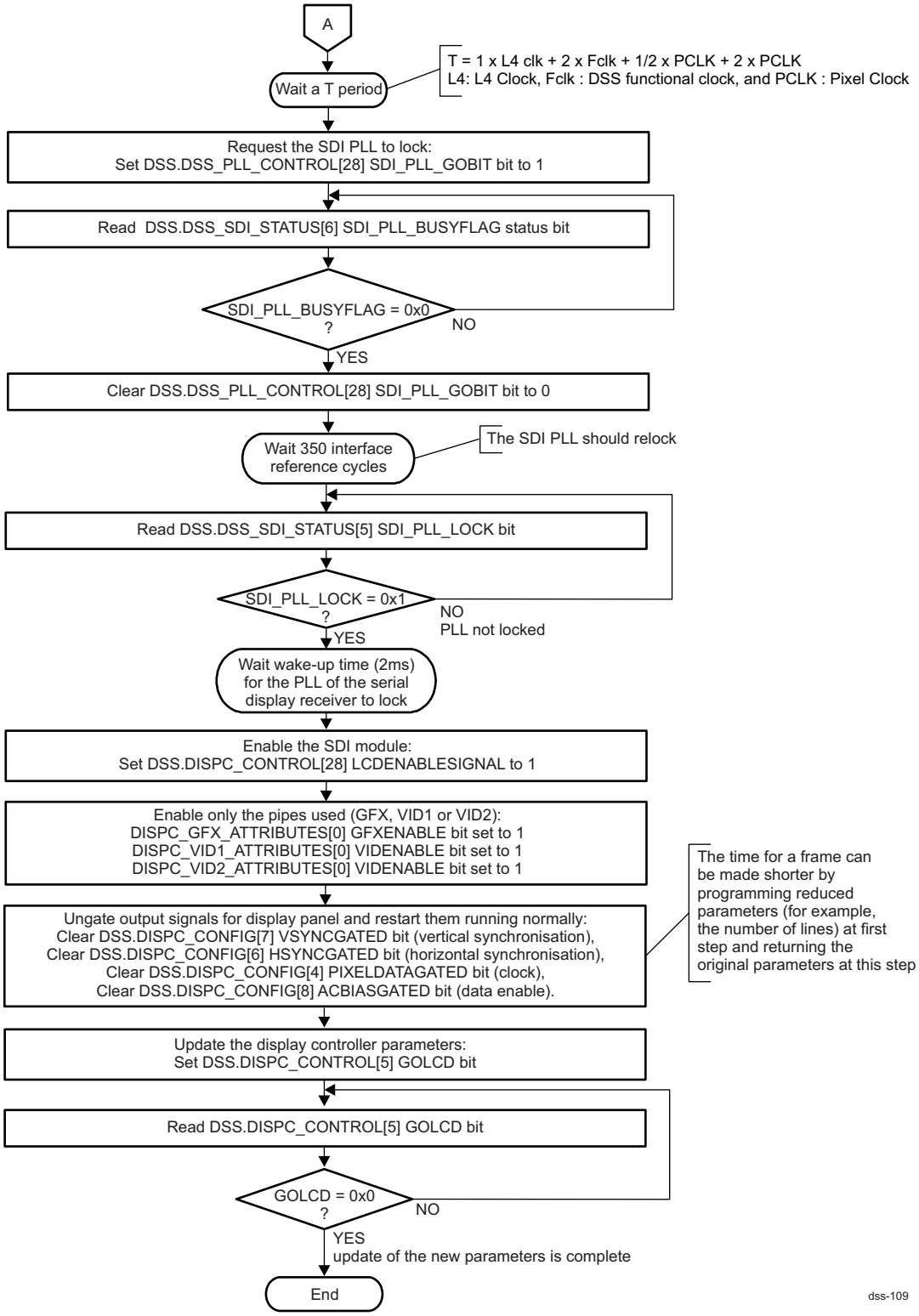


Figure 12-154. SDI Clock Source/Frequency Change Sequence Part B



NOTE: One interface reference cycle is defined as follows:

Pixel Clock Frequency (PCLK)/SDI_PLL_REGN[5:0]/SDI_PLL_HIGHFREQ

Assuming the DSS.DSS_PLL_CONTROL[25:22] SDI_PLL_FREQSEL bit field value is between 0xB and 0xF and the DSS.DSS_PLL_CONTROL[27:26] SDI_PLL_LOCKSEL bit field is set to 0x0.

NOTE: The lock status of the SDI PLL is connected to the GPIO_81 signal of the GPIO3 module. To avoid polling, configure the GPIO3 module to generate an interrupt when the SDI PLL is locked (lock status is 1). For more information, see Chapter 21, *General-Purpose Interface*.

NOTE: TI provides a global solution with OMAP device associated to a programmable 27-bit serial display interface receiver, the SN65LVDS302 receiver. For more details on SN65LVDS302 device, see the TI public web at www.ti.com.

12.5.9.5.2 Simplified Sequence When LCD and PCD Are Swapped

If the values in the DSS.DISPC_DIVISOR[23:16] LCD and DSS.DISPC_DIVISOR[7:0] PCD are exchanged (for example to reduce the internal display controller functional clock frequency while keeping the pixel clock frequency unchanged) with no other changes in the clock path, then there will be no discontinuity in the pixel clock. As a consequence the change will be transparent to the SDI module.

12.5.9.6 SDI Error Management

The DSS.DSS_SDI_STATUS[3] SDI_ERROR status bit indicates an internal buffer underflow/overflow. If such an error occurs, the SDI module must be disabled and enabled again to realign the buffer. This is done by setting the DSS.DISPC_CONTROL[28] LCDENABLESIGNAL bit to 0 (disabling) and to 1 (enabling). Ensure that the DSS.DSS_SDI_STATUS[2] SDI_RESET_DONE bit is set to 1. If this SDI error occurs during the debug phase, it is often due to programming incompatible values for MDIV, NDIV, and PDIV parameters for SDI PLL settings.

The SDI_ERROR internal signal is connected to the GPIO_82 internal signal of the GPIO3 module. Using the GPIO_82 signal, the GPIO3 bank generates two interrupts (GPIO3_MPU_IRQ and GPIO3_IVA2_IRQ) to alert the MPU or IVA2.2 subsystems, respectively, about any SDI error. For additional information, see Chapter 21, *General-Purpose Interface*.

12.6 Display Subsystem Use Cases and Tips

This section gives some generic use cases and tips for setting the modules of the display subsystem.

12.6.1 How to Configure the Scaling Unit in the DISPC Module

This section describes the scaling capability of the display controller (DISPC). The scaling unit is a part of the video pipeline is used when transferring pixels from the system memory (SDRAM or on-chip SRAM) to the LCD panel or the TV set. The scaling unit consists of two scaling blocks: The vertical scaling block followed by the horizontal scaling block. The input pixel format is RGB24. In case the pixel format in system memory is not RGB, the color space conversion unit in front of the scaling unit converts the YUV pixels into RGB pixels. The two scaling units are independent: Neither of them, only one, or both can be used simultaneously

12.6.1.1 Filtering

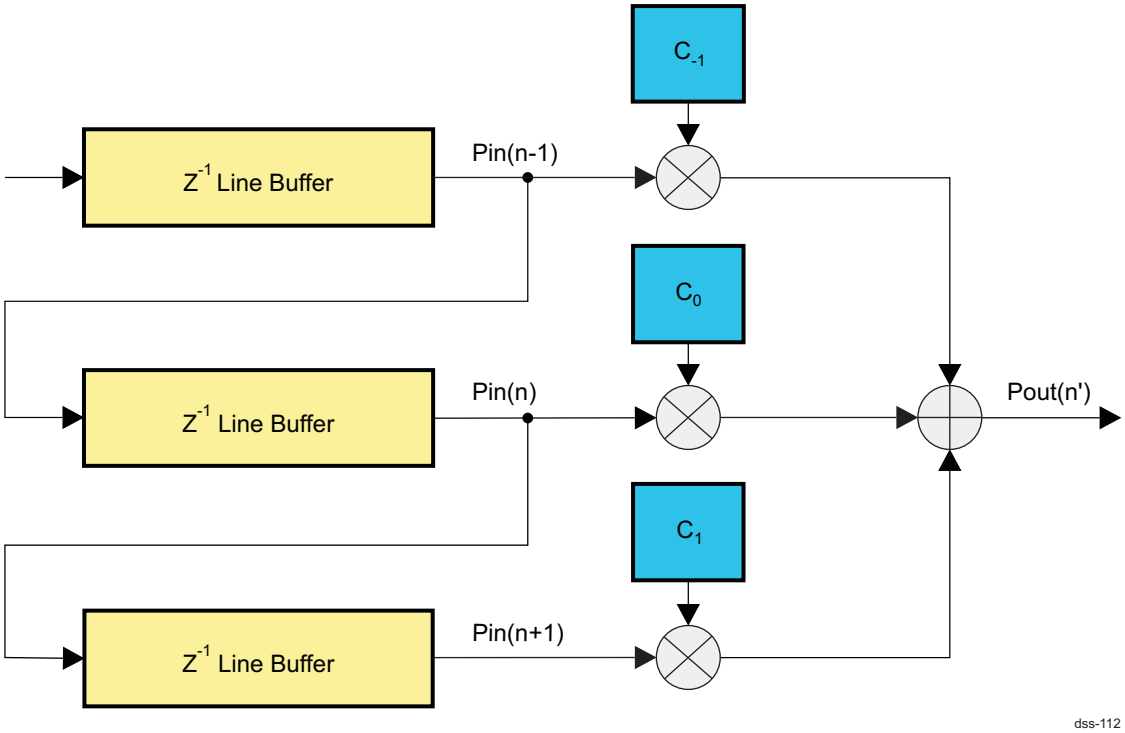
The scaling is used to down-scale, up-scale, or process the image while keeping the same size. It is applied independently horizontally and vertically. The same filtering applies for each color component (R, G, or B).

12.6.1.1.1 Vertical Filtering

The vertical filtering unit is based on a poly-phase rotation architecture with eight phases and three taps. That means that 24 coefficients are programmable

The vertical 3-tap filtering macro architecture is shown in Figure 12-155.

Figure 12-155. Vertical Filtering Macro Architecture (Three Taps)



For the 3-tap vertical up-/downsampling the equation is (with the example of R component):

$$Rout(n) = \left(\sum_{i=-1}^{i=1} Ci(\Phi) \times Rin(n+i) \right) >> 7 \quad \text{dss-E067} \tag{9}$$

Legend:

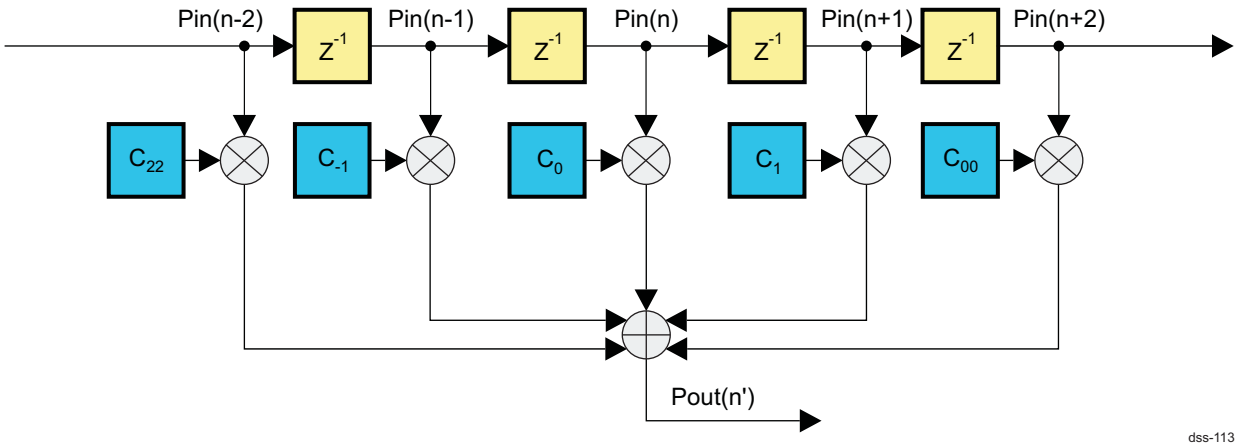
Rout: R component output
Ci():Vertical FIR coefficients
Rin: R component input
The line (n+1) is older than line (n).

The programmable three coefficients of the poly-phase filters are signed 8-bit values (except for the central coefficient C_0(), which is unsigned).

The vertical filtering unit can be configured to support five taps.

The vertical 5-tap filtering macro architecture is shown in Figure 12-156.

Figure 12-156. Vertical Filtering Macro Architecture (Five Taps)



For the 5-tap vertical up/downsampling the equation is (with the example of R component):

$$Rout(n) = \left(\sum_{i=-2}^{i=2} Ci(\Phi) \times Rin(n+i) \right) >> 7 \quad \text{dss-E066} \qquad (10)$$

Legend:

Rout: R component output
Ci():Vertical FIR coefficients with $C_{+2}()=C_{00}()$ and $C_{-2}()=C_{22}()$
Rin: R component input
The line (n+1) is older than line (n).

NOTE: If the 5-tap resizer is used for RGB16 and YUV422 picture formats, the width of the input picture msut be a multiple of 2 pixels and more than 5 pixels. This leads to the following register configuration:

DISPC_VIDn_ATTRIBUTES[21] VIDVERTICALTAPS == 1
DISPC_VIDn_PICTURE_SIZE[10:0] VIDORGSIZEX > 4 and even

The programmable five coefficients of the poly-phase filters are signed 8-bit values (except for the central coefficient C0(), which is unsigned).

In case of three taps, the memory lines are merged into three lines instead of six lines (one line is used as a cache line).

The first line is duplicated to fill up the two first lines (3-tap configuration) and the three first lines (5-tap configuration).

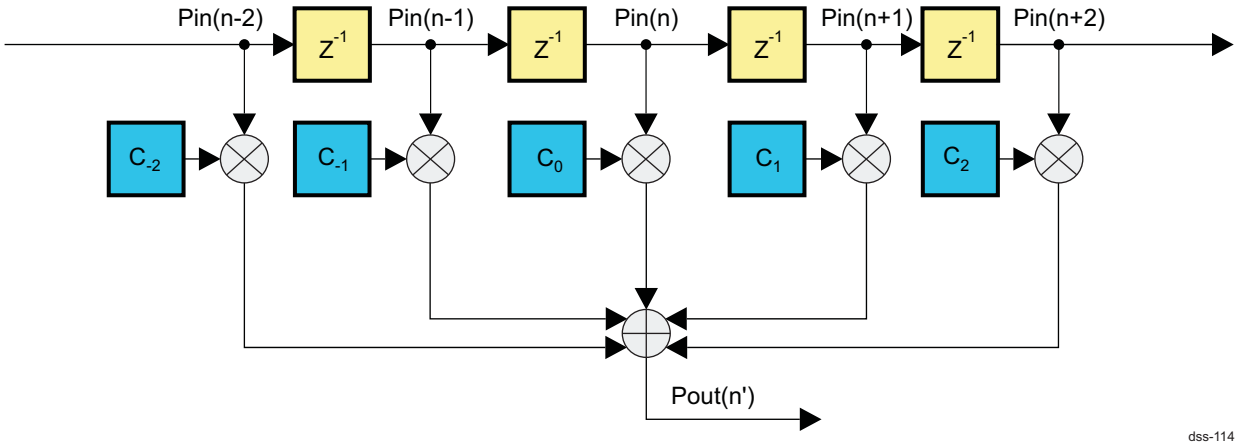
The last line is duplicated if the scaling logic requires loading of more lines and the last line has been reached

12.6.1.1.2 Horizontal Filtering

The horizontal filtering unit is based on a poly-phase rotation architecture with eight phases and five taps. That means that 40 coefficients are programmable.

The horizontal filtering macro architecture is shown in Figure 12-157.

Figure 12-157. Horizontal Filtering Macro Architecture (Five Taps)



For the 5-tap horizontal up-/downsampling, the equation is (with the example of R component):

$$Rout(n) = \left(\sum_{i=-3}^{i=3} Ci(\Phi) \times Rin(n+i) \right) >> 7 \quad (11)$$

dss-E115

Legend:

Rout: R component output

Ci():Vertical FIR coefficients

Rin: R component input

The line (n+1) is older than line (n).

To horizontally and vertically filter the video layer, the phase is calculated separately. The programmable coefficients of the poly-phase filters are signed 8-bit values (except for the central coefficient C_0(), which is unsigned).

The first pixel is duplicated to fill up the three first pixel-buffers (5-tap configuration). The last pixel is duplicated if the scaling logic requires loading of more pixels and the last pixel has been reached

12.6.1.2 Scaling Algorithms

The up-/downsampling finite state machines (FSM) below are detailed in this section.

Figure 12-158 presents the vertical up-/downsampling FSM.

Figure 12-158. Vertical Up-/Down-Sampling Algorithm

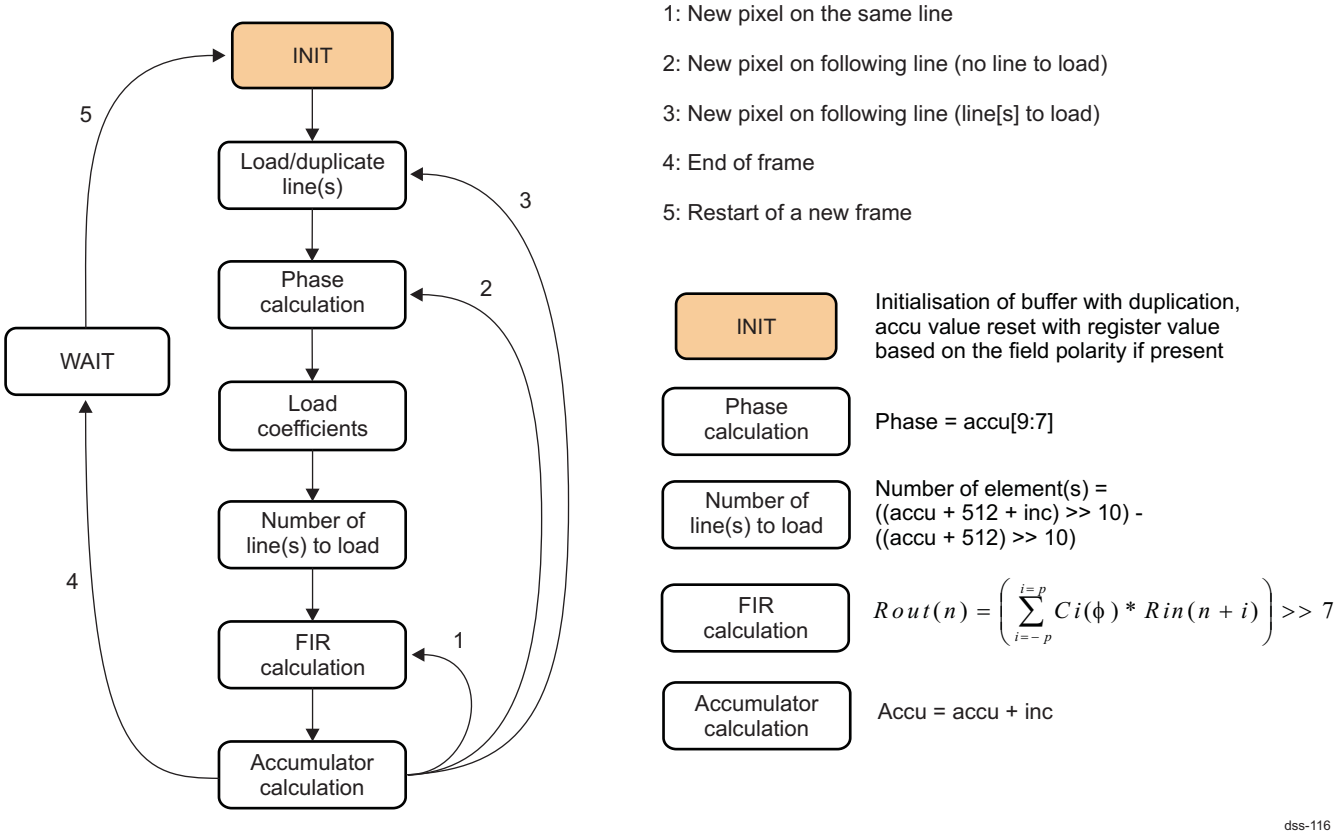
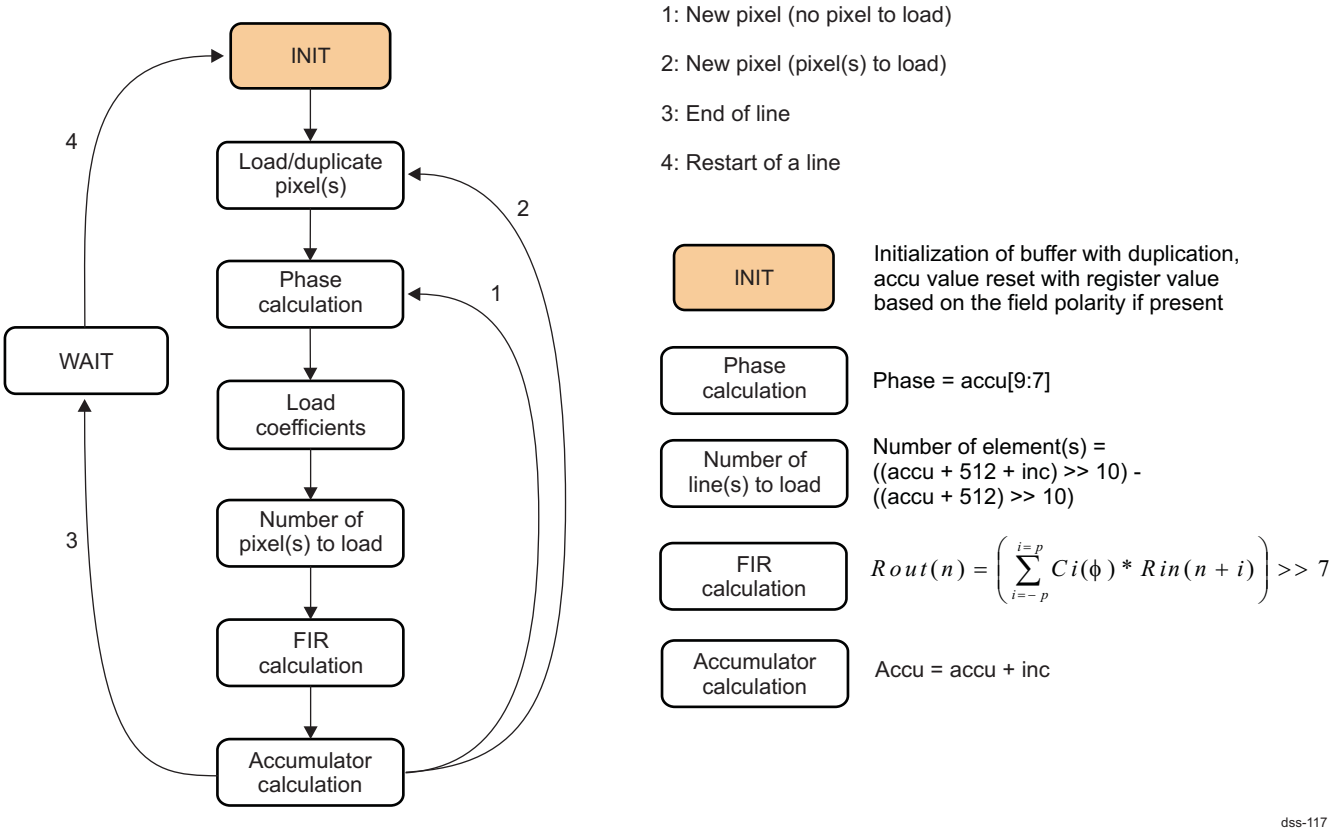


Figure 12-159 presents the horizontal up-/downsampling FSM.

Figure 12-159. Horizontal Up-/Down-Sampling Algorithm



12.6.1.3 Scaling Settings

NOTE:

- In this section, the screen word refers to LCD panel or TV set.
- n indicates pipeline 0 or 1 because there are two video pipelines in the DISPC.

12.6.1.3.1 Register List

The following registers define the scaling registers for the video layer n configuration:

- DSS.DISPC_VIDn_BAj
- DSS.DISPC_VIDn_ATTRIBUTES
- DSS.DISPC_VIDn_FIR
- DSS.DISPC_VIDn_ACCUI
- DSS.DISPC_VIDn_FIR_COEF_Hi
- DSS.DISPC_VIDn_FIR_COEF_HVi
- DSS.DISPC_VIDn_FIR_COEF_Vi

Table 12-72 lists the registers for programming the vertical FIR coefficients (3-tap configuration).

Table 12-72. Vertical FIR Coefficients Corresponding Table (3-Tap Configuration)

C_x()	VidFIRVC$_x$()
C_{-1}()	VidFIRVC$_2$()
C_0()	VidFIRVC$_1$()

Table 12-72. Vertical FIR Coefficients Corresponding Table (3-Tap Configuration) (continued)

$C_x()$	$VidFIRVC_x()$
$C_1()$	$VidFIRVC_0()$

The corresponding registers for programming the vertical FIR coefficients (3-tap configuration) are:

- $VidFIRVC_2()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[31:24] VIDFIRVC2
- $VidFIRVC_1()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[23:16] VIDFIRVC1
- $VidFIRVC_0()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[15:8] VIDFIRVC0

Table 12-73 lists the registers for programming the vertical FIR coefficients (5-tap configuration).

Table 12-73. Vertical FIR Coefficients Corresponding Table (5-Tap Configuration)

$C_x()$	$VidFIRVC_x()$
$C_{22}()$	$VidFIRVC_{22}()$
$C_{-1}()$	$VidFIRVC_2()$
$C_0()$	$VidFIRVC_1()$
$C_1()$	$VidFIRVC_0()$
$C_{00}()$	$VidFIRVC_{00}()$

The corresponding registers for programming the vertical FIR coefficients (5-tap configuration) are:

- $VidFIRVC_{22}()$ = DSS.DISPC_VIDn_FIR_COEF_Vi[15:8] VIDFIRVC22
- $VidFIRVC_2()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[31:24] VIDFIRVC2
- $VidFIRVC_1()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[23:16] VIDFIRVC1
- $VidFIRVC_0()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[15:8] VIDFIRVC0
- $VidFIRVC_{00}()$ = DSS.DISPC_VIDn_FIR_COEF_Vi[7:0] VIDFIRVC00

Table 12-74 lists the registers for programming the horizontal FIR coefficients (5-tap configuration).

Table 12-74. Horizontal FIR Coefficients Corresponding Table (5-Tap Configuration)

$C_x()$	$VidFIRHC_x()$
$C_{-2}()$	$VidFIRHC_4()$
$C_{-1}()$	$VidFIRHC_3()$
$C_0()$	$VidFIRHC_2()$
$C_1()$	$VidFIRHC_1()$
$C_2()$	$VidFIRHC_0()$

The corresponding registers for programming the vertical FIR coefficients (3-tap configuration) are:

- $VidFIRHC_4()$ = DSS.DISPC_VIDn_FIR_COEF_HVi[7:0] VIDFIRHC4
- $VidFIRHC_3()$ = DSS.DISPC_VIDn_FIR_COEF_Hi[31:24] VIDFIRHC3
- $VidFIRHC_2()$ = DSS.DISPC_VIDn_FIR_COEF_Hi[23:16] VIDFIRHC2
- $VidFIRHC_1()$ = DSS.DISPC_VIDn_FIR_COEF_Hi[15:8] VIDFIRHC1
- $VidFIRHC_0()$ = DSS.DISPC_VIDn_FIR_COEF_Hi[7:0] VIDFIRHC0

12.6.1.3.2 Enabling

The video pipeline #n is enabled/disabled by setting/resetting the DSS.DISPC_VIDn_ATTRIBUTES[0].VIDENABLE bit. While the video pipeline is enabled/disabled, the video layer is visible/not visible on the screen (LCD panel or TV set).

The video up-/downsampling block for the video pipeline #n is programmed by setting the DSS.DISPC_VIDn_ATTRIBUTES[6:5] VIDRESIZEENABLE bit field:

- When the VIDRESIZEENABLE[1] bit is set to 1, the video vertical up-/downsampling block is enabled. When set to 0, the vertical resize processing is disabled.
- When the VIDRESIZEENABLE[0] bit is set to 1, the video horizontal up-/downsampling block is enabled. When set to 0, the horizontal resize processing is disabled.
- When the VIDRESIZEENABLE[1:0] is set to 0x3, both horizontal and vertical resize processing are enabled.

NOTE:

- Set a valid configuration before enabling the video up-/downsampling block.
- Vertical and horizontal downsampling are limited to a 0.25 resize factor. When processing a down-scaling with a vertical factor between 0.5 and 0.25, a 5-tap filter configuration must be used. See Section 12.6.1.3.5 for more information concerning the filter coefficients.

12.6.1.3.3 Factor

The following register bit fields define the increment value of the video up-/downsampling block for video pipeline n:

- Vertical up-/downsampling increment value (DSS.DISPC_VIDn_FIR[27:16] VIDFIRVINC bit field, with n = 1 or 2): The unsigned integer value range is [1:4096]. The software calculates the value using the following equation:

$$VIDFIRVINC[11:0] = 1024 \times \frac{VIDORGSIZEY[10:0]}{VIDSIZEY[10:0]} \quad \text{dss-E118} \tag{12}$$

NOTE:

- If the VIDFIRVINC[11:0] bit field value is greater than 4096, it is clipped to 4096. If VIDSIZEY[10:0] equals 0x1, VIDSIZEY[10:0] is replaced by 0x2 in the previous equation.
- The VIDORGSIZEY[10:0] and VIDSIZEY[10:0] bit field values must be programmed with the value desired minus 1.

- Horizontal up-/downsampling increment value (the DSS.DISPC_VIDn_FIR[11:0] VIDFIRHINC bit field, with n = 1 or 2): The unsigned integer value range is [1:4096]. The software calculates the value using the following equation:

$$VIDFIRHINC[11:0] = 1024 \times \frac{VIDORGSIZEX[10:0]}{VIDSIZEX[10:0]} \quad \text{dss-E119} \tag{13}$$

NOTE:

- If the VIDFIRHINC[11:0] bit field value is greater than 4096, it is clipped to 4096. If VIDSIZEX[10:0] equals 1, VIDSIZEX[10:0] is replaced by 2 in the previous equation.
- The VIDORGSIZEX[10:0] and VIDSIZEX[10:0] bit field values must be programmed with the value desired minus 1.

12.6.1.3.4 Initial Phase

- Vertical up-/downsampling accumulator value DSS.DISPC_VIDn_ACCUI[25:16] VIDVERTICALACCU bit fields

  The unsigned integer value range is [0:1023]. The accumulator value indicates on which phase the vertical filtering starts. The value 0 indicates that the phase 0 is the first phase used by the hardware to generate the first data.
- Vertical up-/downsampling accumulator value DSS.DISPC_VIDn_ACCUI[9:0] VIDHORIZONTALACCU

bit fields

The unsigned integer value range is [0:1023]. The accumulator value indicates on which phase the horizontal filtering starts. The value 0 indicates that the phase 0 is the first phase used by the hardware to generate the first data

Table 12-75 lists the vertical/horizontal accumulator values and phases

Table 12-75. Vertical/Horizontal Accumulator Phase

Accumulator Value	Phases
0	0
128	1
256	2
384	3
512	4
640	5
768	6
896	7

NOTE: For LCD output, the initial phase is always 0 (horizontal and vertical.) For TV output, the vertical phases (odd and even) can be nonzero values.

12.6.1.3.5 Coefficients

- **Vertical up-/downsampling coefficients (DSS.DISPC_VIDn_FIR_COEF_HVi and DSS.DISPC_VIDn_FIR_COEF_V)**

  The 3-tap vertical up-/downsampling coefficients are defined in DSS.DISPC_VIDn_FIR_COEF_HVi registers. There are eight registers for the eight phases with three coefficients for each of them so a total of 24 programmable coefficients for the vertical up-/downsampling block. Each register contains two 8-bit signed coefficients and one 8-bit unsigned coefficient (central one).

  In addition, there are 2-tap vertical up-/downsampling coefficients defined in DSS.DISPC_VIDn_FIR_COEF_Vi registers. There are eight registers for the eight phases with two coefficients for each of them so a total of 16 programmable coefficients for the vertical up-/downsampling block used in addition of the 3-tap registers defined above. Each register contains two 8-bit signed coefficients (C_{22}() and C_{00}()).

  In case of 5-tap configuration, both sets of registers, DSS.DISPC_VIDn_FIR_COEF_HVi and DSS.DISPC_VIDn_FIR_COEF_V, are used. In case of 3-tap configuration, only one set of registers, DSS.DISPC_VIDn_FIR_COEF_HV, is used.

- **Horizontal up-/downsampling coefficients (DSS.DISPC_VIDn_FIR_COEF_Hi and DSS.DISPC_VIDn_FIR_COEF_HV)**

  The 5-tap horizontal up-/downsampling coefficients are defined in DSS.DISPC_VIDn_FIR_COEF_Hi and DSS.DISPC_VIDn_FIR_COEF_HVi registers. There are eight registers for the eight phases with five coefficients for each register, for a total of 40 programmable coefficients for the horizontal up-/downsampling block.

  Each DSS.DISPC_VIDn_FIR_COEF_Hi register contains three 8-bit signed coefficients and one 8-bit unsigned coefficient (central one), and each DSS.DISPC_VIDn_FIR_COEF_HVi contains one 8-bit signed coefficient.

Table 12-76 through Table 12-81 give the programmable coefficients for the FIR up/downsampling filters (Max-Fauque-Berthier method).

12.6.1.3.5.1 Up-Sampling

Table 12-76 gives the 24 coefficients to program the vertical upsampling (3-tap configuration).

Table 12-76. Up-Sampling Vertical Filter Coefficients (Three Taps)

Phases	VidFIRVC$_2$()	VidFIRVC$_1$()	VidFIRVC$_0$()
0	0	128	0
1	3	123	2
2	12	111	5
3	32	89	7
4	0	64	64
5	7	89	32
6	5	111	12
7	2	123	3

Table 12-77 gives the 40 coefficients to program the vertical upsampling (5-tap configuration).

Table 12-77. Up-Sampling Vertical Filter Coefficients (Five Taps)

Phases	VidFIRVC$_{22}$()	VidFIRVC$_2$()	VidFIRVC$_1$()	VidFIRVC$_0$()	VidFIRVC$_{00}$()
0	0	0	128	0	0
1	-1	13	124	-8	0
2	-2	30	112	-11	-1
3	-5	51	95	-11	-2
4	0	-9	73	73	-9
5	-2	-11	95	51	-5
6	-1	-11	112	30	-2
7	0	-8	124	13	-1

Table 12-78 gives the 40 coefficients to program the horizontal upsampling (5-tap configuration).

Table 12-78. Up-Sampling Horizontal Filter Coefficients (Five Taps)

Phases	VidFIRHC$_4$()	VidFIRHC$_3$( )	VidFIRHC$_2$()	VidFIRHC$_1$()	VidFIRHC$_0$()
0	0	0	128	0	0
1	-1	13	124	-8	0
2	-2	30	112	-11	-1
3	-5	51	95	-11	-2
4	0	-9	73	73	-9
5	-2	-11	95	51	-5
6	-1	-11	112	30	-2
7	0	-8	124	13	-1

The upsampling coefficients register configuration (vertical three taps and horizontal five taps) is the following:

- DSS.DISPC_VIDn_FIR_COEF_H0 = 0x00800000
- DSS.DISPC_VIDn_FIR_COEF_HV0 = 0x00800000
- DSS.DISPC_VIDn_FIR_COEF_H1 = 0x0D7CF800
- DSS.DISPC_VIDn_FIR_COEF_HV1 = 0x037B02FF
- DSS.DISPC_VIDn_FIR_COEF_H2 = 0x1E70F5FF
- DSS.DISPC_VIDn_FIR_COEF_HV2 = 0x0C6F05FE
- DSS.DISPC_VIDn_FIR_COEF_H3 = 0x335FF5FE
- DSS.DISPC_VIDn_FIR_COEF_HV3 = 0x205907FB
- DSS.DISPC_VIDn_FIR_COEF_H4 = 0xF74949F7
- DSS.DISPC_VIDn_FIR_COEF_HV4 = 0x00404000

- DSS.DISPC_VIDn_FIR_COEF_H5 = 0xF55F33FB
- DSS.DISPC_VIDn_FIR_COEF_HV5 = 0x075920FE
- DSS.DISPC_VIDn_FIR_COEF_H6 = 0xF5701EFE
- DSS.DISPC_VIDn_FIR_COEF_HV6 = 0x056F0CFF
- DSS.DISPC_VIDn_FIR_COEF_H7 = 0xF87C0DFF
- DSS.DISPC_VIDn_FIR_COEF_HV7 = 0x027B0300

NOTE: In this case, the DSS.DISPC_VIDn_FIR_COEF_Vi registers are not used.

The upsampling coefficients register configuration (both vertical and horizontal five taps) is the following:

- DSS.DISPC_VIDn_FIR_COEF_H0 = 0x00800000
- DSS.DISPC_VIDn_FIR_COEF_HV0 = 0x00800000
- DSS.DISPC_VIDn_FIR_COEF_V0 = 0x00000000
- DSS.DISPC_VIDn_FIR_COEF_H1 = 0x0D7CF800
- DSS.DISPC_VIDn_FIR_COEF_HV1 = 0x0D7CF8FF
- DSS.DISPC_VIDn_FIR_COEF_V1 = 0x0000FF00
- DSS.DISPC_VIDn_FIR_COEF_H2 = 0x1E70F5FF
- DSS.DISPC_VIDn_FIR_COEF_HV2 = 0x1E70F5FE
- DSS.DISPC_VIDn_FIR_COEF_V2 = 0x0000FEFF
- DSS.DISPC_VIDn_FIR_COEF_H3 = 0x335FF5FE
- DSS.DISPC_VIDn_FIR_COEF_HV3 = 0x335FF5FB
- DSS.DISPC_VIDn_FIR_COEF_V3 = 0x0000FBFE
- DSS.DISPC_VIDn_FIR_COEF_H4 = 0xF74949F7
- DSS.DISPC_VIDn_FIR_COEF_HV4 = 0xF7404000
- DSS.DISPC_VIDn_FIR_COEF_V04 = 0x000000F7
- DSS.DISPC_VIDn_FIR_COEF_H5 = 0xF55F33FB
- DSS.DISPC_VIDn_FIR_COEF_HV5 = 0xF55F33FE
- DSS.DISPC_VIDn_FIR_COEF_V5 = 0x0000FEFB
- DSS.DISPC_VIDn_FIR_COEF_H6 = 0xF5701EFE
- DSS.DISPC_VIDn_FIR_COEF_HV6 = 0xF5701EFF
- DSS.DISPC_VIDn_FIR_COEF_V6 = 0x0000FFFE
- DSS.DISPC_VIDn_FIR_COEF_H7 = 0xF87C0DFF
- DSS.DISPC_VIDn_FIR_COEF_HV7 = 0xF87C0D00
- DSS.DISPC_VIDn_FIR_COEF_V7 = 0x000000FF

12.6.1.3.5.2 Down-Sampling

Table 12-79 gives the 24 coefficients to program the vertical downsampling (3-tap configuration).

Table 12-79. Down-Sampling Vertical Filter Coefficients (Three Taps)

Phases	$VidFIRVC_2()$	$VidFIRVC_1()$	$VidFIRVC_0()$
0	36	56	36
1	40	57	31
2	45	56	27
3	50	55	23
4	18	55	55
5	23	55	50
6	27	56	45

Table 12-79. Down-Sampling Vertical Filter Coefficients (Three Taps) (continued)

Phases	VidFIRVC$_2$()	VidFIRVC$_1$()	VidFIRVC$_0$()
7	31	57	40

Table 12-80 gives the 40 coefficients to program the vertical downsampling (5-tap configuration).

Table 12-80. Down-Sampling Vertical Filter Coefficients (Five Taps)

Phases	VidFIRVC$_{22}$()	VidFIRVC$_2$()	VidFIRVC$_1$()	VidFIRVC$_0$()	VidFIRVC$_{00}$()
0	0	36	56	36	0
1	4	40	55	31	-2
2	8	44	54	27	-5
3	-12	48	53	22	-7
4	-9	17	52	51	17
5	-7	22	53	48	12
6	-5	27	54	44	8
7	-2	31	55	40	4

Table 12-81 gives the 40 coefficients to program the horizontal downsampling (5-tap configuration).

Table 12-81. Down-Sampling Horizontal Filter Coefficients (Five Taps)

Phases	VidFIRHC$_4$()	VidFIRHC$_3$( )	VidFIRHC$_{2(}$)	VidFIRHC$_{1(}$)	VidFIRHC$_0$()
0	0	36	56	36	0
1	4	40	55	31	-2
2	8	44	54	27	-5
3	-12	48	53	22	-7
4	-9	17	52	51	17
5	-7	22	53	48	12
6	-5	27	54	44	8
7	-2	31	55	40	4

The downsampling coefficients register configuration (vertical three taps and horizontal five taps) is the following:

- DSS.DISPC_VIDn_FIR_COEF_H0 = 0x24382400
- DSS.DISPC_VIDn_FIR_COEF_HV0 = 0x24382400
- DSS.DISPC_VIDn_FIR_COEF_H1 = 0x28371FFE
- DSS.DISPC_VIDn_FIR_COEF_HV1 = 0x28391F04
- DSS.DISPC_VIDn_FIR_COEF_H2 = 0x2C361BFB
- DSS.DISPC_VIDn_FIR_COEF_HV2 = 0x2D381B08
- DSS.DISPC_VIDn_FIR_COEF_H3 = 0x303516F9
- DSS.DISPC_VIDn_FIR_COEF_HV3 = 0x3237170C
- DSS.DISPC_VIDn_FIR_COEF_H4 = 0x11343311
- DSS.DISPC_VIDn_FIR_COEF_HV4 = 0x123737F7
- DSS.DISPC_VIDn_FIR_COEF_H5 = 0x1635300C
- DSS.DISPC_VIDn_FIR_COEF_HV5 = 0x173732F9
- DSS.DISPC_VIDn_FIR_COEF_H6 = 0x1B362C08
- DSS.DISPC_VIDn_FIR_COEF_HV6 = 0x1B382DFB
- DSS.DISPC_VIDn_FIR_COEF_H7 = 0x1F372804

- DSS.DISPC_VIDn_FIR_COEF_HV7 = 0x1F3928FE

NOTE:
- In this case, the DSS.DISPC_VIDn_FIR_COEF_Vi registers are not used.
- In this case, the downsampling factor must be higher than 1/2.

The downsampling coefficients register configuration (both the vertical and the horizontal five taps) is the following:

- DSS.DISPC_VIDn_FIR_COEF_H0 = 0x24382400
- DSS.DISPC_VIDn_FIR_COEF_HV0 = 0x24382400
- DSS.DISPC_VIDn_FIR_COEF_V0 = 0x00000000
- DSS.DISPC_VIDn_FIR_COEF_H1 = 0x28371FFE
- DSS.DISPC_VIDn_FIR_COEF_HV1 = 0x28371F04
- DSS.DISPC_VIDn_FIR_COEF_V1 = 0x000004FE
- DSS.DISPC_VIDn_FIR_COEF_H2 = 0x2C361BFB
- DSS.DISPC_VIDn_FIR_COEF_HV2 = 0x2C361B08
- DSS.DISPC_VIDn_FIR_COEF_V2 = 0x000008FB
- DSS.DISPC_VIDn_FIR_COEF_H3 = 0x303516F9
- DSS.DISPC_VIDn_FIR_COEF_HV3 = 0x3035160C
- DSS.DISPC_VIDn_FIR_COEF_V3 = 0x00000CF9
- DSS.DISPC_VIDn_FIR_COEF_H4 = 0x11343311
- DSS.DISPC_VIDn_FIR_COEF_HV4 = 0x113433F7
- DSS.DISPC_VIDn_FIR_COEF_V4 = 0x0000F711
- DSS.DISPC_VIDn_FIR_COEF_H5 = 0x1635300C
- DSS.DISPC_VIDn_FIR_COEF_HV5 = 0x163530F9
- DSS.DISPC_VIDn_FIR_COEF_V5 = 0x0000F90C
- DSS.DISPC_VIDn_FIR_COEF_H6 = 0x1B362C08
- DSS.DISPC_VIDn_FIR_COEF_HV6 = 0x1B362CFB
- DSS.DISPC_VIDn_FIR_COEF_V6 = 0x0000FB08
- DSS.DISPC_VIDn_FIR_COEF_H7 = 0x1F372804
- DSS.DISPC_VIDn_FIR_COEF_HV7 = 0x1F3728FE
- DSS.DISPC_VIDn_FIR_COEF_V7 = 0x0000FE04

NOTE: This configuration must be used for vertical downsampling factors between 1/2 and 1/4

12.6.2 Display Low-Power Refresh Settings

This section describes the display low-power refresh application on the device. The display subsystem remains active while saving power by putting unused power domains and unused modules into idle mode. This process can be expanded to include the screen saver mode in which the MPU subsystem wakes up to update the frame buffer and then returns to idle mode. On an OMAP platform, where power consumption is of high importance, the display modes must be configured properly to achieve optimal power savings

The display low-power refresh mode can be used in the following scenarios:

- During the period of time when there is no application running and the backlight turns off.
- Once the backlight turns off, the LCD display can be shut off or can be refreshed showing the time and date. The screen saver mode can be used to update the time every minute.

This section discusses the methodology for finding optimal power savings. These settings are detailed for a 16-bit, 240 x 320 pixel QVGA LCD.

12.6.2.1 Display Low-Power Refresh Overview

When the device is not in idle mode, meaning all clocks are on and the power is applied to all power domains, the following activity typically occurs with respect to the display subsystem:

- The MPU subsystem is processing
- The display subsystem DMA controller is moving data from the SDRAM frame buffer location to the display subsystem internal FIFO.
- The LCD data is being sent from the internal FIFO to the display panel.

When the MPU goes into idle mode, the following activity occurs:

- The display subsystem DMA controller remains active, moving data from the SDRAM frame buffer to the internal FIFO.
- The SDRAM will go in and out of self-refresh between transfers.
- The display subsystem internal FIFO will continue to send LCD data to the display panel.

This procedure is named as the display low-power refresh scenario.

NOTE: In the device, the display subsystem has its own power domain (the DSS power domain).

12.6.2.2 Display Subsystem Clock

12.6.2.2.1 Display Subsystem Clock Configuration

The display subsystem contains two possible functional clock sources, DSS functional clock 1 (DSS1_ALWON_FCLK) and DSI PLL functional clock 1 (DSI1_PLL_FCLK):

- DSS1_ALWON_FCLK is sourced from DPLL4 (DPLL4_ALWON_FCLK), with several multipliers available and is configured in the PRCM.CM_CLKSEL_DSS[4:0] CLKSEL_DSS1 bit field.
- DSI1_PLL_FCLK is one of the two output clocks from the DSI PLL.

The pixel clock is set as either DSS1_ALWON_FCLK or DSI1_PLL_FCLK by configuring the DSS.DSS_CONTROL[0] DISPC_CLK_SWITCH bit

NOTE: When the DSI PLL is in bypass mode, the DSI1_PLL_FCLK clock is the DSS2_ALWON_FCLK clock. This ensures the backward compatibility with OMAP2 devices.

The LCD logic clock is determined by the DSS.DISPC_DIVISOR[23:16] LCD bit field. This divisor is used on the DSS functional clock that is selected in the DSS_CONTROL register (either DSS1_ALWON_FCLK or DSS2_ALWON_FCLK). This LCD divisor selects the logical clock frequency which is used to clock the logic in the display subsystem. For some applications there is a required minimum logical clock frequency. The lower the logical clock frequency then the lower the power consumption.

The pixel clock is determined by setting the DSS.DISPC_DIVISOR[7:0] PCD bit field. This divisor is used on the LCD logic clock.

In the following example, the DPLL4 clock (DPLL4_ALWON_FCLK) is enabled and running at 266 MHz:

The PRCM.CM_CLKSEL2_PLL[18:8] PERIPH_DPLL_MULT bit field is set to 0x4 and the PRCM.CM_CLKSEL2_PLL[6:0] PERIPH_DPLL_DIV bit field is set to 0x1 (DPLL4 x 4/(1+1)):

DPLL4_ALWON_FCLKOUTX2= DPLL4_ALWON_FCLK(266MHz) x 4/2 = 532 MHz

NOTE: The DPLL4_ALWON_FCLOUTX2 clock is an internal clock in DPLL4 module after the DPLL_MULT and DPLL_DIV stages. The DPLL4_M4X2_CLK clock is one of the DPLL4 output clocks and is the clock source for DSS1_ALWON_FCLK.

The DSS.DSS_CONTROL[0] DSS_CLK_SWITCH bit is set to 0x0 to select DSS1_ALWON_FCLK as the display subsystem functional clock:

DSS1_ALWON_FCLK = DPLL4_M4X2_CLK

The PRCM.CM_CLKSEL_DSS[4:0] CLKSEL_DSS1 bit field is set to 0x08:

$$\text{DSS1\_ALWON\_FCLK} = \frac{\text{DPLL4\_ALWON\_FCLKOUTX2}(532\text{MHz})}{8} = 66.5\ \text{MHz} \quad \text{dss-E120} \tag{14}$$

The DSS.DISPC_DIVISOR[23:16] LCD bit field is set to 0x01:

$$\text{LogicClock} = \frac{\text{DSS1\_ALWON\_FCLK}(66.5\text{MHz})}{1} = 66.5\ \text{Mhz} \quad \text{dss-E121} \tag{15}$$

The DSS.DISPC_DIVISOR[6:0] PCD bit field is set to 0x0C:

$$\text{PixelClock} = \frac{\text{LogicClock}(66.5\text{MHz})}{12} = 5.54\ \text{Mhz} \quad \text{dss-E122} \tag{16}$$

12.6.2.2.1.1 Pixel Clock Frequency Settings to Reduce Power Consumption

Power consumption is reduced when a low pixel clock frequency is used. If the clock frequency is set too low, however, the frames-per-second (FPS) are reduced. This can result in visible flickering on the screen each time the screen is refreshed. To avoid electrical polarization problems, refer to the appropriate LCD panel datasheet to determine the maximum range of pixel clock frequency variation. To save power, therefore, the pixel clock frequency during low-power mode must be set as low as possible, but high enough to eliminate visible flickering

12.6.2.2.1.2 Display Subsystem Divider Settings to Reduce Power Consumption

The pixel clock is determined by the DSS.DISPC_DIVISOR[7:0] PCD and DSS.DISPC_DIVISOR[23:16] LCD settings. In most cases, the LCD[7:0] bit field is set to 0x1 and the PCD[7:0] bit field is used as the main divider. To reduce power consumption, software users should investigate if the DSS.DISPC_DIVISOR[23:16] LCD bit field can be set to a value other than 0x1 and then decrease DSS.DISPC_DIVISOR[7:0] PCD bit field value. For example, if the desired pixel clock is 1.625 MHz with a 13-MHz functional clock, then this pixel clock can be achieved by setting DSS.DISPC_DIVISOR[23:16] LCD to 0x1 and DSS.DISPC_DIVISOR[7:0] PCD to 0x8. The same pixel clock can be achieved by setting DSS.DISPC_DIVISOR[23:16] LCD to 0x2 and DSS.DISPC_DIVISOR[7:0] PCD to 0x4.

12.6.2.2.2 Display Subsystem Clock Enable

To take the DSS out of reset, all DSS-related clocks must be enabled, and the DPLL4 clock must be enabled. After taking the DSS out of reset, these clocks can be disabled if they are not used. The following clocks must be enabled before the DSS can come out of reset:

- PRCM.CM_FCLKEN_DSS[0] EN_DSS1 = 0x1
- PRCM.CM_FCLKEN_DSS[1] EN_DSS2 = 0x1
- PRCM.CM_FCLKEN_DSS[2] EN_TV = 0x1
- PRCM.CM_ICLKEN_DSS[0] EN_DSS = 0x1
- PRCM.CM_CLKEN_PLL[18:16] EN_PERIPH_DPLL = 0x7

Once these clocks are enabled, the display subsystem can be taken out of reset.

The following sections explain the display low-power mode configuration options, which are determined by product requirements (LCD panel type).

12.6.2.3 DPLL4 in Low-Power Mode

For optimal power savings in low-power mode, DPLL4 can be put into low-power stop mode. Thus the DSS1_ALWON_FCLK clock cut when DPLL4 is in low-power stop mode. Software users must switch from DSS1_ALWON_FCLK to DSI1_PLL_FCLK by setting the DSS.DSS_CONTROL[0] DISPC_CLK_SWITCH bit to 0x1.

NOTE: Before switching to DSI1_PLL_FCLK, the DSI PLL and the HS divider must be programmed and the DSI PLL must be locked.

DPLL4 can be put in low-power stop mode in either of the following methods:

- Manually: When setting the PRCM.CM_CLKEN_PLL[18:16] EN_PERIPH_DPLL bit to 0x1.
- Automatically: When setting the PRCM.CM_AUTOIDLE_PLL[5:3] AUTO_PERIPH_DPLL bit field to 0x1. DPLL4 is automatically put in low-power stop mode when none of the 96- MHz and 54- MHz clocks are required anymore. DPLL4 is also restarted automatically.

Software users must remember to change the clock configuration after enabling DPLL4 when leaving low-power mode by setting the DSS.DSS_CONTROL[0] DISPC_CLK_SWITCH bit to 0x0. The DSS1_ALWON_FCLK clock will be selected.

Lock time must be considered before disabling the DSS1_ALWON_FCLK clock.

12.6.2.4 Autoidle and Smart Idle

12.6.2.4.1 Autoidle

To further save power consumption, the autoidle feature at the module can be enabled for the active modules. For example, the PRCM and the system control modules are active during this mode. By enabling the autoidle feature, the clocks at the module level are gated when they are not needed.

The RFBI, display controller, and L4 interfaces can internally gate their clocks to decrease power consumption if no transaction is present on the related bus. The following bits must be set to enable this functionality:

- DSS.DSS_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the display subsystem
- DSS.RFBI_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the RFBI
- DSS.DISPC_SYSCONFIG[0] AUTOIDLE bit (1: Autoidle; 0: Clock free-running) for the display controller
- DSS.DISPC_CONFIG[9] FUNCGATED bit (1: Functional clocks gated enabled, 0: Functional clocks gated disabled) for the display controller

12.6.2.4.2 Smart-Idle

The smart-idle feature can be enabled to allow the module to enter idle when the clocks are not needed. The smart-idle feature can be enabled for the display subsystem submodules to further save power consumption:

- Display subsystem: DSS.DSS_SYSCONFIG[4:3] SIDLEMODE
- Display controller: DSS.DISPC_SYSCONFIG[4:3] SIDLEMODE
- RFBI: DSS.RFBI_SYSCONFIG[4:3] SIDLEMODE

12.6.2.5 FIFO Thresholds

The display subsystem internal FIFO is used to move data to the LCD panel. This FIFO is filled by the display subsystem DMA controller. The DMA controller is triggered to start and stop based on two thresholds:

- DSS.DISPC_GFX_FIFO_THRESHOLD[11:0] GFXFIFOLOWTHRESHOLD
- DSS.DISPC_GFX_FIFO_THRESHOLD[27:16] GFXFIFOHIGHTHRESHOLD

When the level of the FIFO reaches the low threshold, the internal DMA controller begins to fill the FIFO With the data in the frame buffer. Once the amount of pixel data reaches the high threshold, the internal DMA controller stops.

12.6.2.5.1 FIFO Threshold Settings to Reduce Power Consumption

Power consumption is reduced by increasing the difference between the high and low FIFO threshold levels, thereby leaving the SDRAM in self-refresh for a longer period of time. To perform this reduction, consider the following:

- The low FIFO threshold level must be as low as possible, but not low enough to cause any underflow.
- The high FIFO threshold level must not exceed the FIFO size minus one burst. A value above this limit results in the DMA controller trying to fill the FIFO to a level that cannot be reached, which will increase power consumption.
- The difference between high and low FIFO threshold levels must not be less than one burst size. These settings do not reduce power consumption because the SDRAM never goes into self-refresh, but they will avoid underflow.

12.6.2.6 Vertical and Horizontal Timings

The vertical and horizontal timings and the pixel clock speed determine the number of frames updated per second. Figure 12-160 shows the timings for a 240 x 320 pixel QVGA LCD panel. If the pulse width (also called blanking) and the front porch parameters are increased, more setup time is added before the data is transferred. This additional time is beneficial for delaying the data transfer if the data is not ready because of bandwidth limitations. Care must be taken to determine the fps when modifying these parameters.

Use the following formula to determine the fps for a 240 x 320 QVGA LCD:

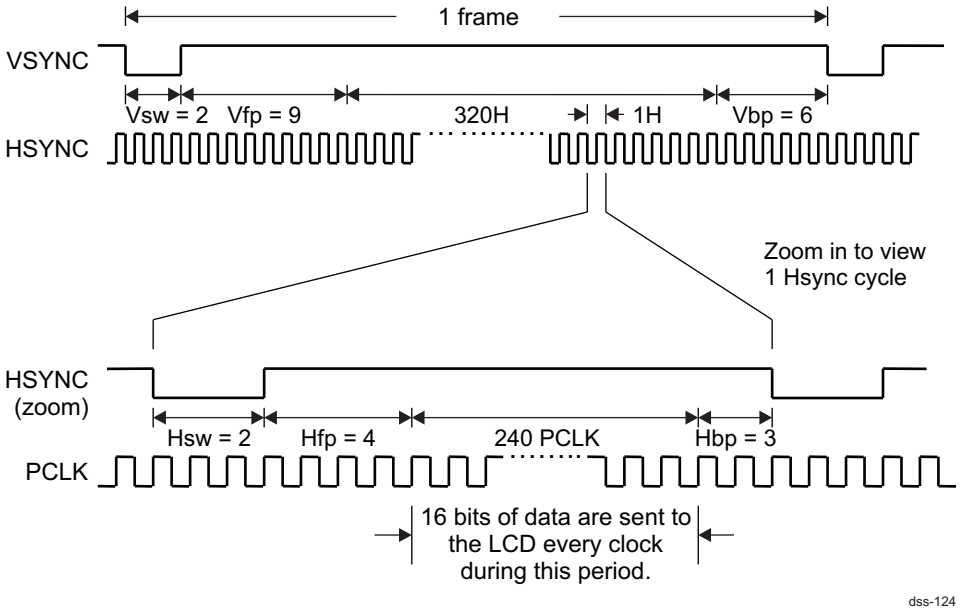
$$fps = \frac{1}{[(Hsw + 1) + (Hrp + 1) + 240 + (Hbp + 1)] \times [(Vsw + 1) + Vrp + 320 + Vbp)] \times (PCLK)} \quad \text{dss-E123} \qquad (17)$$

With:

- Hsw: DSS.DISPC_TIMING_H[7:0] HSW bit field value
- Hp: DSS.DISPC_TIMING_H[19:8] HFP bit field value
- Hbp: DSS.DISPC_TIMING_H[31:20] HBP bit field value
- Vsw: DSS.DISPC_TIMING_V[7:0] VSW bit field value
- Vp: DSS.DISPC_TIMING_V[19:8] VFP bit field value
- Vpb: DSS.DISPC_TIMING_V[31:20] VBP bit field value
- PCLK: Pixel clock period

The horizontal (Hsw) and vertical (Vsw) pulse widths and the horizontal front (Hp) and back (Hbp) porches are increased by 1 because the value is programmed as the desired value minus 1.

Figure 12-160. QVGA LCD Timings



The fps for the example of 6-MHz pixel clock with the setting shown in Figure 12-160 is as follows:

$$fps = \frac{1}{[(2 + 1) + 4 + 240 + 3] \times [(2 + 1) + 9 + 320 + 6] \times 166.67 \times 10^{-9}} \quad \text{dss-E125} \tag{18}$$

$$fps = 71Hz$$

12.6.2.6.1 Horizontal and Vertical Timing Settings to Reduce Power Consumption

The number of fps that the screen is refreshed is also determined by the vertical and horizontal timings. Consequently, longer timings between frames (blanking periods) reduce the fps and reduce average power consumption. Shorter blanking periods increase fps and increases power consumption. If the blanking between frames is too small, a FIFO underflow may occur.

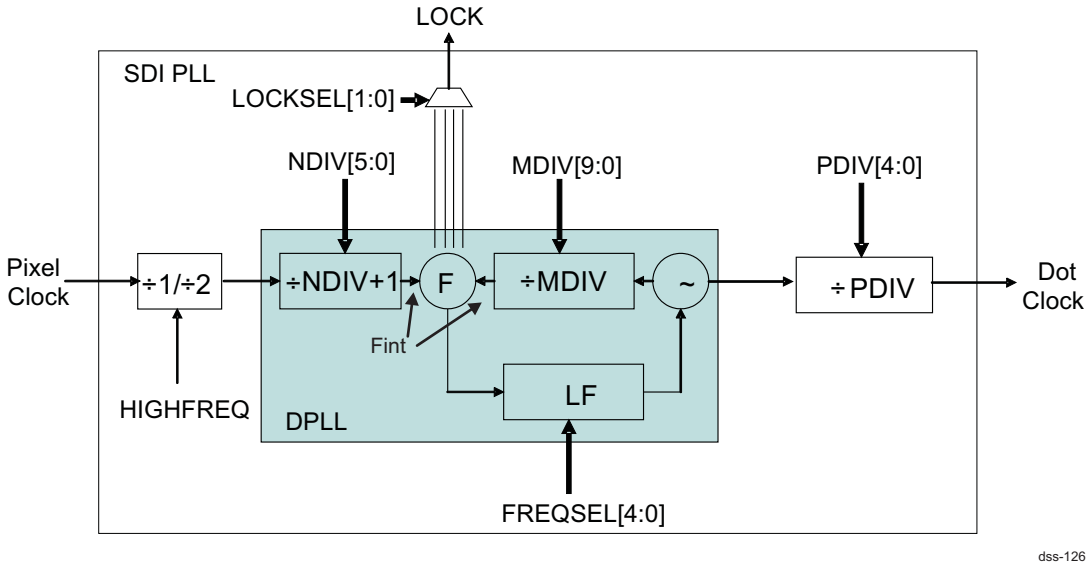
12.6.3 How to Configure the Serial Display Interface Module With FlatLink3G Protocol

This section describes the basic settings of Serial Display Interface (SDI) module and related Phase-Locked Loop (PLL).

12.6.3.1 SDI PLL Architecture

The SDI PLL architecture is based on the Digital Phase-Locked Loop (DPLL) that is described in , *Power, Reset, and Clock Management*. Figure 12-161 gives a functional description of the SDI PLL.

Figure 12-161. SDI PLL Architecture



NOTE:

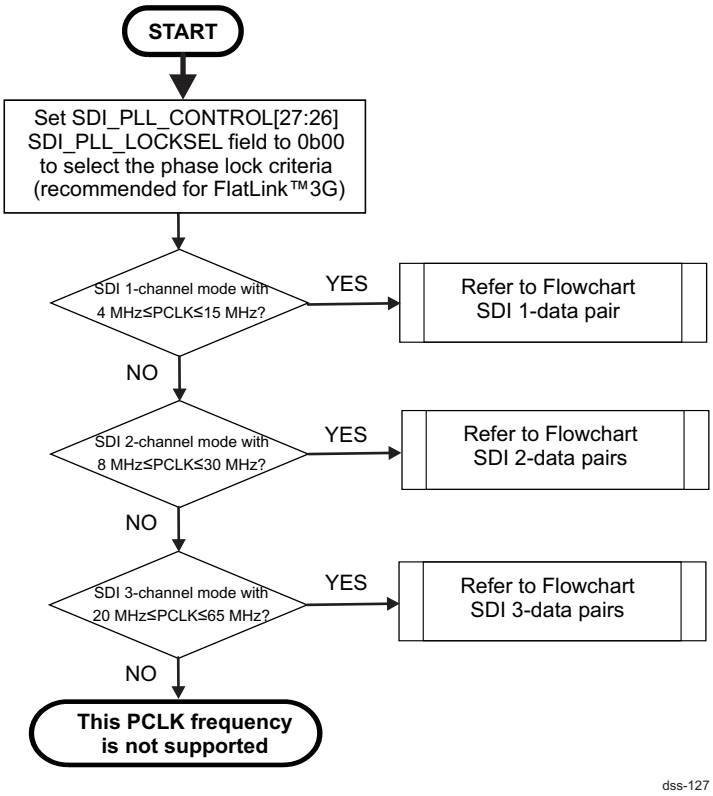
- FlatLink3G data rate is always at SCLK frequency.
- Software programmers must ensure that Pixel Clock = Dot Clock. If it is not the case the DSS.DSS_SDI_STATUS[3] SDI_ERROR status bit will be set indicating the pixel buffer overflow/underflow.

12.6.3.2 SDI PLL Configuration

The SDI PLL configuration depends on the pixel clock (PCLK), and on the number of data pairs in the FlatLink 3G protocol.

The following flowcharts detail the SDI PLL configuration from the PCLK frequency point of view. Figure 12-162 is the main flowchart. See Figure 12-163, Figure 12-164, and Figure 12-165 for details in PLL setting depending the PCLK input frequency.

Figure 12-162. Main Flowchart



NOTE: Compliance with the above ranges is required to meet the SN65LVDS302 receiver specifications.

Figure 12-163. Flowchart SDI 1-Data Pair

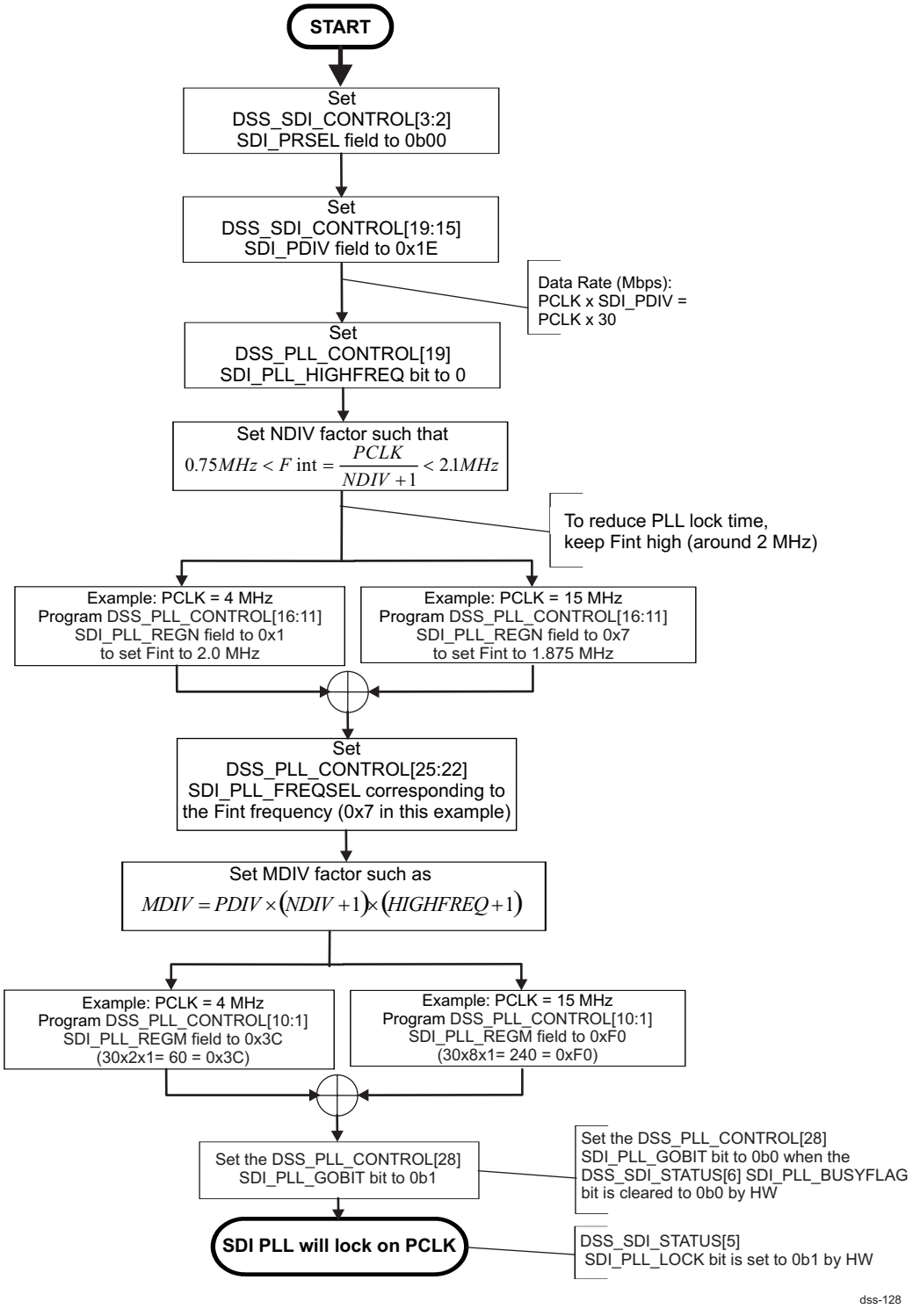


Figure 12-164. Flowchart SDI 2-Data Pairs

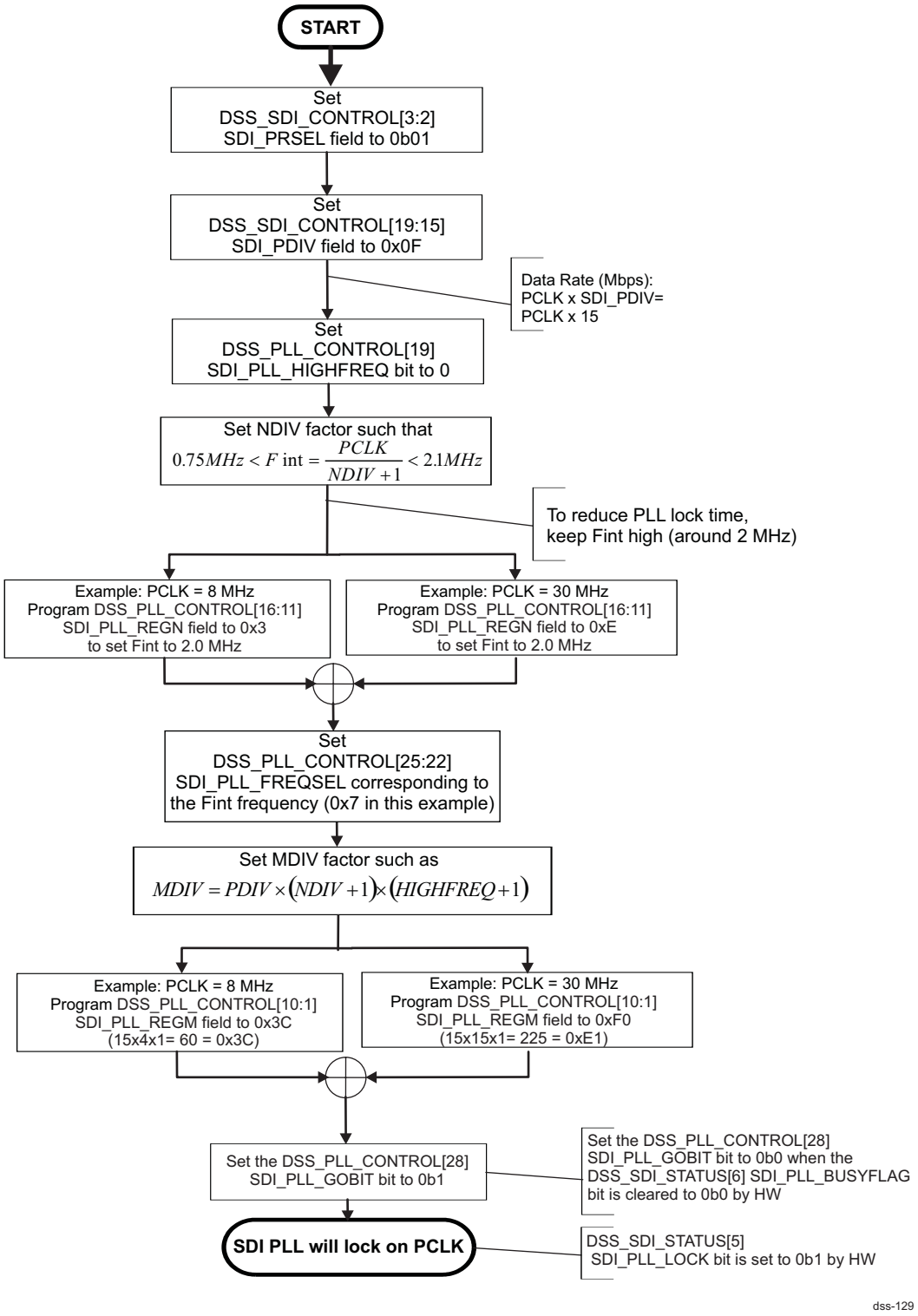
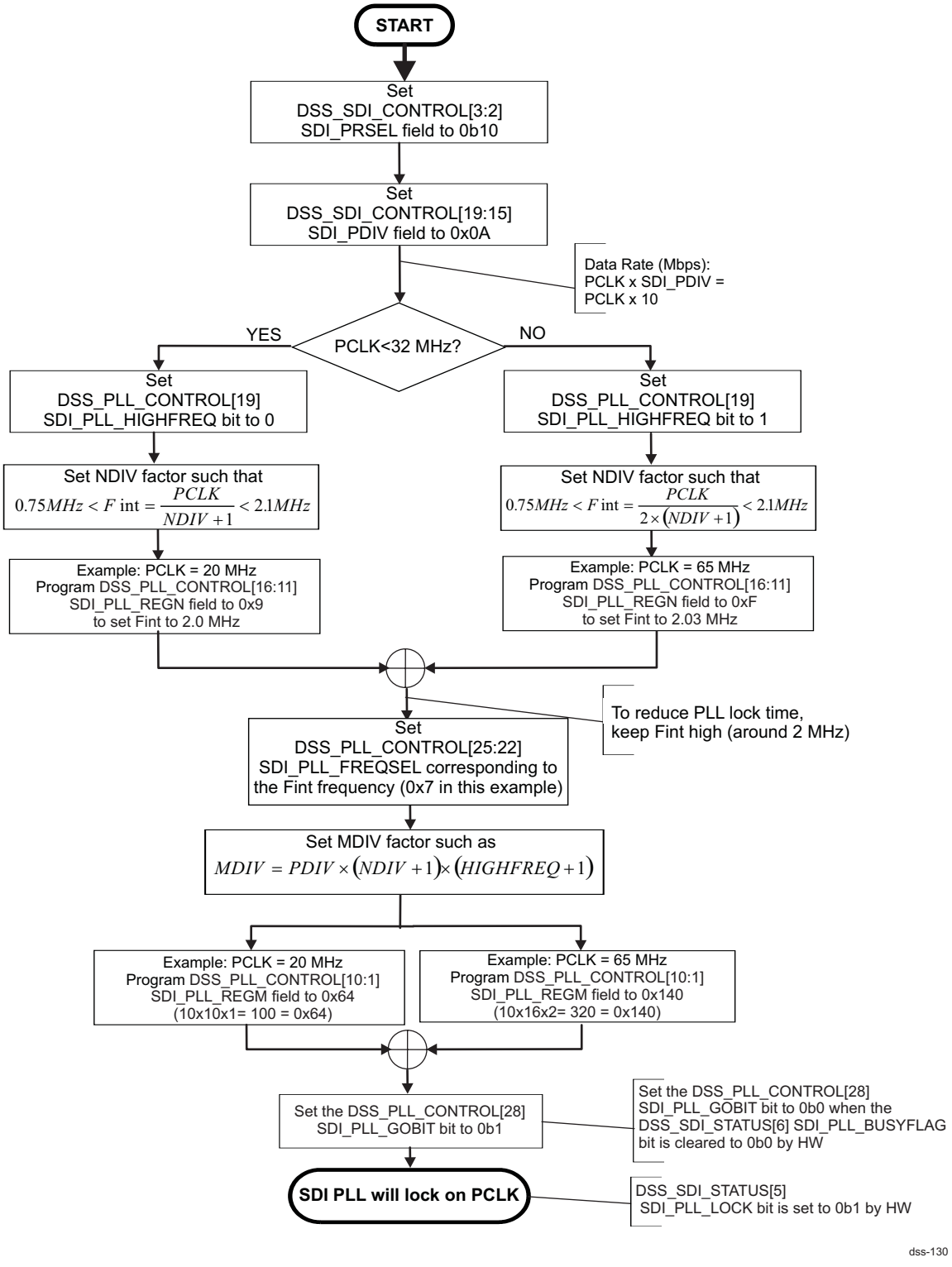


Figure 12-165. Flowchart SDI 3-Data Pairs



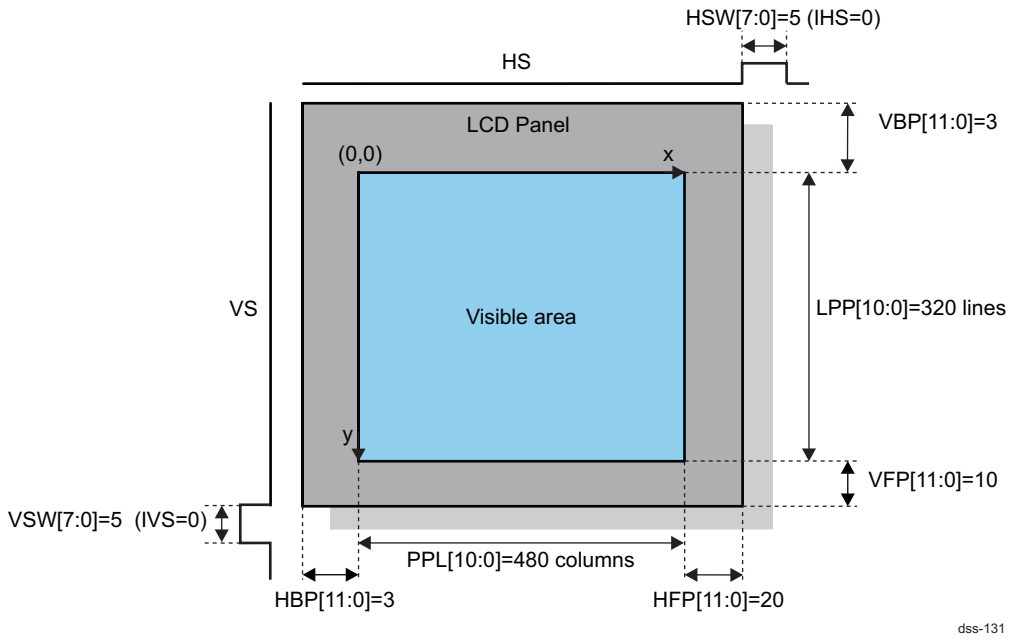
CAUTION

The PLL low-power mode (LP mode) can be activated with DSS.DSS_PLL_CONTROL[21] SDI_PLL_PLLLPMODE bit. By setting this bit to 1, software users reduce the power and increases the jitter. In LP mode, there is no effect on lock time; but in this mode, the SDI PLL allows only 0.75-2.1MHz range (DSS.DSS_PLL_CONTROL[25:22] SDI_PLL_FREQSEL bit field value between 0x3 and 0x7). If software users are using the range 7.5 MHz to 21 MHz (DSS.DSS_PLL_CONTROL[25:22] SDI_PLL_FREQSEL bit field value between 0xB and 0xF) in non-LP mode, and then switching to LP mode and setting the PLL in the range of 0.75 MHz to 2.1 MHz, there will be an effect on the lock time.

12.6.3.3 Application Example: HVGA Display

The following example details the SDI PLL settings for a half-VGA display with the following parameters shown in Figure 12-166.

Figure 12-166. HVGA Display



12.6.3.3.1 HVGA Display

- Display resolution: 320 x 480
- Frame refresh rate: 58.4 Hz
- Vertical visible pixel: 480 lines
- Vertical front porch: 20 lines
- Vertical sync: 5 lines
- Vertical back porch: 3 lines
- Horizontal visible pixel: 320 columns
- Horizontal front porch: 10 columns
- Horizontal sync: 5 columns
- Horizontal back porch: 3 columns

The calculation of the total number of pixel and Blanking overhead is as follows:

- Visible area pixel count: 480 x 320 = 153600 pixel
- Total frame pixel count: (480 + 20 + 5 + 3) x (320 + 10 + 5 + 3) = 171704 pixel
- Blanking overhead: (171704 153600)/153600 = 11.8%

This application requires the following SDI parameters:

- Pixel clock frequency: 171704 x 58.4 Hz = 10.03 MHz
- Serial data rate:
  - 1-channel mode: 10.03 MHz x 30 bit/channel = 300.9 Mbps
  - 2-channel mode: 10.03 MHz x 15 bit/channel = 150.45 Mbps

12.6.3.3.2 SDI PLL Settings for 1-Channel Mode:

- 1-data pair: Set DSS_SDI_CONTROL[3:2] SDI_PRSEL bit field to 0b00.
- Set the DSS_SDI_CONTROL[19:15] SDI_PDIV bit field to 0x1E.
- Set the DSS_PLL_CONTROL[19] SDI_PLL_HIGHFREQ bit to 0.
- Set the DSS_PLL_CONTROL[16:11] SDI_PLL_REGN bit field to 0x4. Thus Fint will be set to 2 MHz.
- Set the DSS_PLL_CONTROL[25:22] SDI-PLL_FREQSEL bit field to 0x7.
- Set DSS_PLL_CONTROL[10:1] SDI_PLL_REGM bit field to 0x96 (30x5x1 = 150 = 0x96).

12.6.3.3.3 SDI PLL Settings for 2-Channel Mode:

- 2-data pair: Set DSS_SDI_CONTROL[3:2] SDI_PRSEL bit field to 0b01.
- Set the DSS_SDI_CONTROL[19:15] SDI_PDIV bit field to 0x0F.
- Set the DSS_PLL_CONTROL[19] SDI_PLL_HIGHFREQ bit to 0.
- Set the DSS_PLL_CONTROL[16:11] SDI_PLL_REGN bit field to 0x4. Thus Fint will be set to 2 MHz.
- Set the DSS_PLL_CONTROL[25:22] SDI-PLL_FREQSEL bit field to 0x7.
- Set DSS_PLL_CONTROL[10:1] SDI_PLL_REGM bit field to 0x4B (15x5x1 = 75 = 0x4B).

12.6.4 How to Interface OMAP Device With SN65LVDS302 Receiver for an XGA Display Application

Texas Instruments Inc. provides a full FlatLink3G solution with the device associated to a SN65LVDS302 receiver. This section will also detail the hardware connections between these two devices and the basic software settings

This use case is based on the usage of a large (XGA) 16 Mega colors display requiring the three data channels and the complete output data bus to be used. Some adaptations may be adopted if the device is used in a less demanding use case

12.6.4.1 Hardware Connections

Figure 12-167 details the hardware connections with the device, the SN65LVDS302 receiver, the display panel and the power IC. The power IC can be the companion chip

Figure 12-167. Hardware Connections for FlatLink3G Application

NOTE: The SN65LVDS302 device was design-optimized for robustness to operate reliably in a dense environment with other digital switching ICs. In cell phone designs, the SN65LVDS302 often shares a power supply with the LCD display driver provided through the FPC interconnect. To minimize the power supply noise floor it is suggested to provide good decoupling near the SN65LVDS302 power pins. Using four capacitors (2x0.01F and 2x0.1F) was found to provide good performance. At the very least, it is recommended to install one 0.1F and one 0.01F capacitor near the SN65LVDS302. To avoid large current loops and trace inductance, the trace length between decoupling cap and IC power inputs pins must be minimized. Placing the capacitor underneath the SN65LVDS302 device on the bottom of the PCB is often a good choice

12.6.4.2 SN65LVDS302 Receiver Description

SN65LVDS302 is a serial interface receiver which converts the FlatlinkTM3G signals (3 sub-LVDS data pairs) into 27 signals (24 RGB bits, HS, VS and DE signals). It also provides a pixel clock (4 MHz to 65 MHz) which is formed from the sub-LVDS input clock pair, internally multiplied for data deserialization and then divided by same factor.

The device may operate in different modes, depending on the input data format. One, two or all of the three data pairs will be used, following the LS0 and LS1 configuration. The clock division rate will automatically be selected from 30 to 10. Pixel clock polarity can be chosen using the CPOL signal

The output bus order is configurable by SWAP pin configuration (R[7:0], G[7:0], B[7:0] or B[0:7], G[0:7], R[0:7]). Furthermore, the CMOS outputs rising/falling edges will be selected between slow or fast depending on F/S input to fit display timing constraints with regards to EMI and consumption consequences.

In the XGA display application detailed in Figure 7, the configuration is the following:

- LS1 = 1 and LS0 = 0: 3-data pairs mode is selected
- SWAP = 0: The data output format is R[7:0], G[7:0], B[7:0]
- F/S = 1: The CMOS bus time rise is selected in fast output
- CPOL = 0: The output clock polarity is rising edge

For more details on SN65LVDS302 receiver, please see the SN65LVDS302 data sheet (SLLS733)

The RXEN signal will enable the OMAP device to control SN65LVDS302 activity which will signal channel parity error through CPE signal. These two previous signals are connected to OMAP GPIO2 module

12.6.4.3 SDI Software Settings

12.6.4.3.1 SDI Configuration

To use the SDI with the TI FlatLink3G display panel:

- Set the DSS.DISPC_CONTROL[3] STNTFT bit to 0b1 (active matrix display operation mode)
- Set the DSS.DISPC_CONTROL[9:8] TFTDATALINES bit field to 0x3 (24-bit data output)
- Set the DSS.DSS_SDI_CONTROL[1:0] SDI_BWSEL bit field to 0x2 (color depth is 24 bits)

12.6.4.3.2 Signal Features Configuration

For proper SDI operation, configure the following features in the display controller:

- Set the DSS.DISPC_POL_FREQ[16] RF bit to 0b1 for driving HSYNC and VSYNC on the rising edge of PCLK.
- Set the DSS.DISPC_POL_FREQ[17] ONOFF bit to 0b1.
- Clear the DSS.DISPC_POL_FREQ[14] IPC bit to 0b0 for driving the pixel data on the rising edge of PCLK
- Set the DSS.DISPC_CONTROL[29] LCDENABLEPOL bit to 0b1 for setting the LCD-enable signal

active high.

The following settings are not required by the SDI2 interface, but these polarities settings are the common polarities used by FlatLink3G displays:

- Clear the DSS.DISPC_POL_FREQ[13] IHS bit to 0b0 for setting HSYNC active high.
- Clear the DSS.DISPC_POL_FREQ[12] IVS bit to 0b0 for setting VSYNC active high.
- Clear the DSS.DISPC_POL_FREQ[15] IEO bit to 0b0 for setting the data enable active high.

12.6.4.3.3 SDI PLL Configuration

The SDI module settings for an XGA display application with 20% blanking overhead are detailed in this section.

- XGA display: 1024 x 768 = 786432 pixels
- 20% blanking overhead: 157286 pixels
- Visible are pixel count: 786432 + 157286 = 943718
- Frame Refresh Rate: 60 Hz
- Pixel Clock Frequency: 943718 x 60 = 56.6 MHz (in 20-65 MHz range for 3-data pairs).
- Serial Data rate: 3-channel mode : 56.6 MHz x 10 bit/channel = 566 Mbps

12.6.4.3.3.1 SDI PLL settings for 3-Channel Mode:

- 3-data pair: Set DSS_SDI_CONTROL[3:2] SDI_PRSEL bit field to 0b10
- Set the DSS_SDI_CONTROL[19:15] SDI_PDIV bit field to 0x0A
- Set the DSS_PLL_CONTROL[19] SDI_PLL_HIGHFREQ bit to 1 as PCLK>32 MHz
- Set the DSS_PLL_CONTROL[16:11] SDI_PLL_REGN bit field to 0xE. Thus Fint will be set to 2.02 MHz
- Set the DSS_PLL_CONTROL[25:22] SDI_PLL_FREQSEL bit field to 0x7
- Set DSS_PLL_CONTROL[10:1] SDI_PLL_REGM bit field to 0x12C (10x(14+1)x2 = 300 = 0x12C)

Set the SDI_PLL_CONTROL[27:26] SDI_PLL_LOCKSEL bit field to 0b00 to select the phase lock criteria to 0b00 (recommended for FlatLink3G mode)

12.6.4.4 SN65LVDS302 Receiver Settings

12.6.4.4.1 Receiver Power-Up

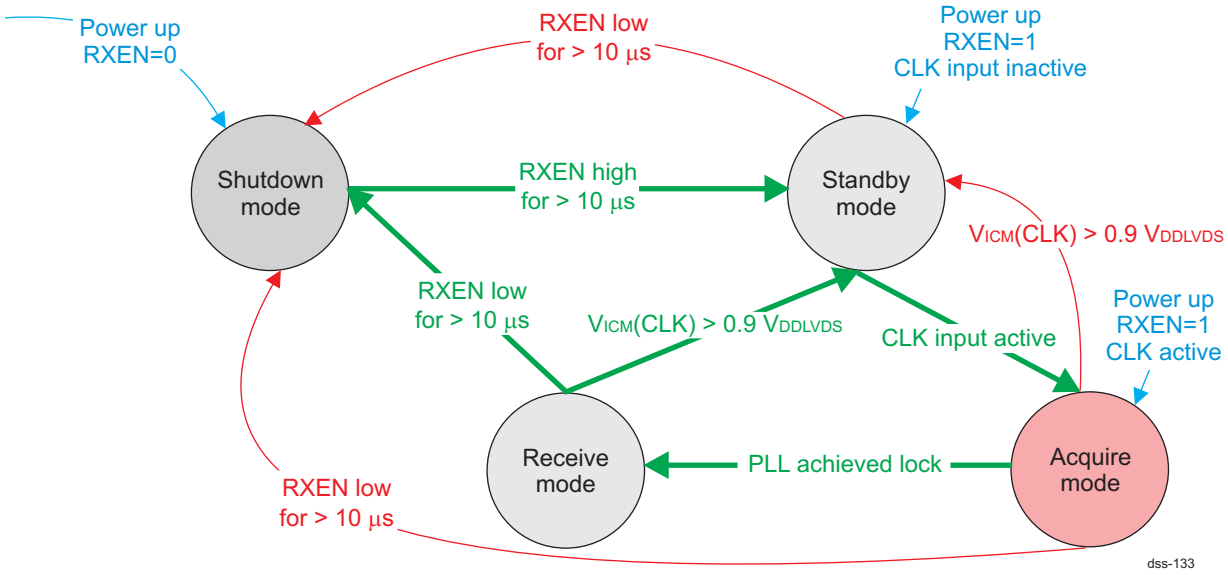
At power up, the receiver is enabled immediately if RXEN=1 (enters Acquire mode) and disabled if RXEN=0 (enters shutdown mode). The receiver may be turned off at any time by pulling RXEN low for at least 10 µs. This disables the CMOS drivers and turns off the receiver PLL, putting the device in shutdown mode. The RXEN signal is connected to the gpio_34 pin of the device. Thus the GPIO2 module can control the receiver power-up.

NOTE: The RXEN input incorporates glitch suppression logic to avoid unwanted switching. The input must be pulled continuously low for at least 10 µs to force the receiver to enter shutdown. The gpio_34 signal must be set continuously high by the OMAP device for at least 10 µs to activate the receiver. An input pulse shorter than 5 µs is interpreted as a glitch and is ignored.

12.6.4.4.2 Receiver Modes and Transitions

Figure 12-168 details the SN65LVDS302 receiver modes and transitions.

Figure 12-168. SN65LVDS302 Receiver Modes and Transitions



12.6.4.4.2.1 Modes Description

- Shutdown mode

  The SN65LVDS302 receiver may be put in Shutdown mode by holding the RXEN pin low. This turns off most of the receiver circuitry including the Sub-LVDS receivers, PLL, and deserializers. The Sub-LVDS input impedance remains 100- while any input signal becomes ignored. All outputs will hold a static output pattern: R[0:7]=G[0:7]=B[0:7]=VS=HS=high; DE=PCLK=low. The current draw in Shutdown mode will be nearly zero

- Standby mode

  The SN65LVDS302 receiver will enter the Standby mode when the SN65LVDS302 receiver is not in Shutdown mode but the clock input common mode range on the Sub-LVDS clock input shifts to VDDLVDS. The CLK input incorporates a pull-up circuitry. This circuit will shift the Sub-LVDS clock input common mode voltage to VDDLVDS in the absence of any input signal. All circuitry, except for the Sub-LVDS clock input Standby monitor will be shutdown. The Sub-LVDS input impedance remains 100- while any input signal on the data inputs D0, D1, and D2 becomes ignored. All outputs will hold a static output pattern: R[0:7]=G[0:7]=B[0:7]=VS=HS=high; DE=PCLK=low. The current draw in Standby mode will be very low.

- Active Modes

  The SN65LVDS302 receiver will be in active mode when the device is enabled by pulling RXEN high and VICM on the CLK input is driven below 0.9xVDD by an external input signal. Current draw in active mode will be dependent on operating frequency and the number of data transitions in the data payload

- Acquire Mode (PLL approaches lock)

  While the SN65LVDS302 receiver is enabled and a Sub-LVDS clock input present, the receiver PLL will pursue lock to the input clock. As long as the PLL is not locked the output data bus will hold a static output pattern: R[0:7]=G[0:7]=B[0:7]=VS=HS=high; DE=PCLK=low.

- Receive Mode

  When the receiver PLL achieves lock, the receiver enters the normal receive mode with the output bus presenting the deserialized data.

12.6.4.4.2.2 Mode Transitions

Table 12-82. SN65LVDS302 Receiver Mode Transitions

Mode Transition	Use Case	Transition Specifics
Shutdown to Standby	Drive RXEN high to enable receiver	• 1. RXEN high for longer than 10 µs • 2. Receiver enters Standby mode – a. R[0:7]=G[0:7]=B[0:7]=VS=HS remain high and DE=PCLK low – b. receiver turns on clock input monitor
Standby to acquire	Transmitter activity detected	• 1. CLK input monitor detects clock input activity • 2. Outputs remain static • 3. the receiver PLL becomes enabled
Acquire to Receive	Link is ready to receive data from OMAP SDI module.	• 1. PLL is active and achieves lock within 2 µs • 2. D1, D2, and D3 become active • 3. first Data word was recovered • 4. parallel output bus switches from static output pattern to output first valid data word
Receive to Standby	Receiver requested to enter Standby mode by input common mode voltage V_{ICM}>0.9 V_{DDLVDS} (that is, transmitter output clock turns high-impedance)	• 1. Transmitter disables outputs • 2. RX Input monitor detects V_{ICM}>0.9 V_{DDLVDS} • 3. R[0:7]=G[0:7]=B[0:7]=VS=HS transition to high and DE=PCLK to low on next falling PLL clock edge • 4. PLL shuts down • 5. Clock activity input monitor remains active
Receive/Standby to Shutdown	Turn off receiver	• 1. RXEN pulled low for longer than 10 µs • 2. Receiver immediately puts all outputs into high impedance • 3. Most receiver blocks are shut down for least power consumption

12.6.4.4.3 Parity Error Detection and Handling

The SN65LVDS302 receiver performs an error checking on the basis of a parity bit that is transmitted across the LVDS interface from the transmitting device. Once the SN65LVDS302 detects the presence of the clock and the receiver PLL has locked onto PCLK, then the parity is checked. The parity bit covers the 27-bit data payload (reserved bits are not covered in the parity calculations). Odd Parity bit signaling is used. The parity error is output on the output CPE pin:

- If the sum of the 27 data bits and the parity bit result in an odd number, the receive data are assumed to be valid and CPE output will be held low.
- If the sum equals an even number, parity error is declared and CPE output will indicate high for half a PCLK period.

The CPE output will be set with the data bit transition and cleared after 1/2 the data bit time. The receiver CPE output pin is connected to gpio_35 pin. The OMAP device can use the gpio_35 input pin to generate an IRQ, respectively GPIO2_MPU_IRQ for MPU subsystem and GPIO2_IVA2_IRQ for IVA2.2 subsystem. This will allow the OMAP device to count every detected parity error with a simple counter triggered by GPIO2 module interrupt request.

When a parity error is detected, the data on that PCLK cycle is not output. Instead, the last valid data from a previous PCLK cycle is repeated on the output bus. This is to prevent any bit error that may occur on the LVDS link from causing perturbations in VS, HS or DE that may be visually disruptive to the display panel.

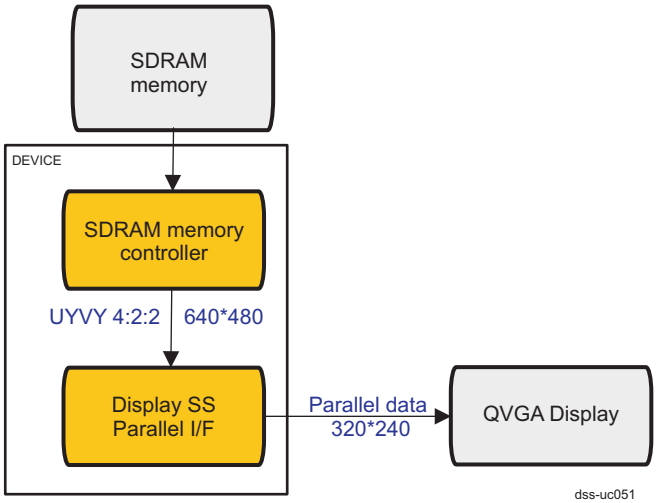
12.6.5 Camcorder Use Case: How to Configure the Display Subsystem When Connected With a QVGA LCD Panel

12.6.5.1 Overview

This section describes the configuration and used of a QVGA LCD panel through the parallel data interface (in RFBI by-pass mode).

The display subsystem reads each image from the SDRAM memory in the UYVY 4:2:2 format and converts it to a RGB18-666 pixel format to send to the QVGA LCD panel (see Figure 12-169).

Figure 12-169. Overview



The camcorder use case is configured as follows:

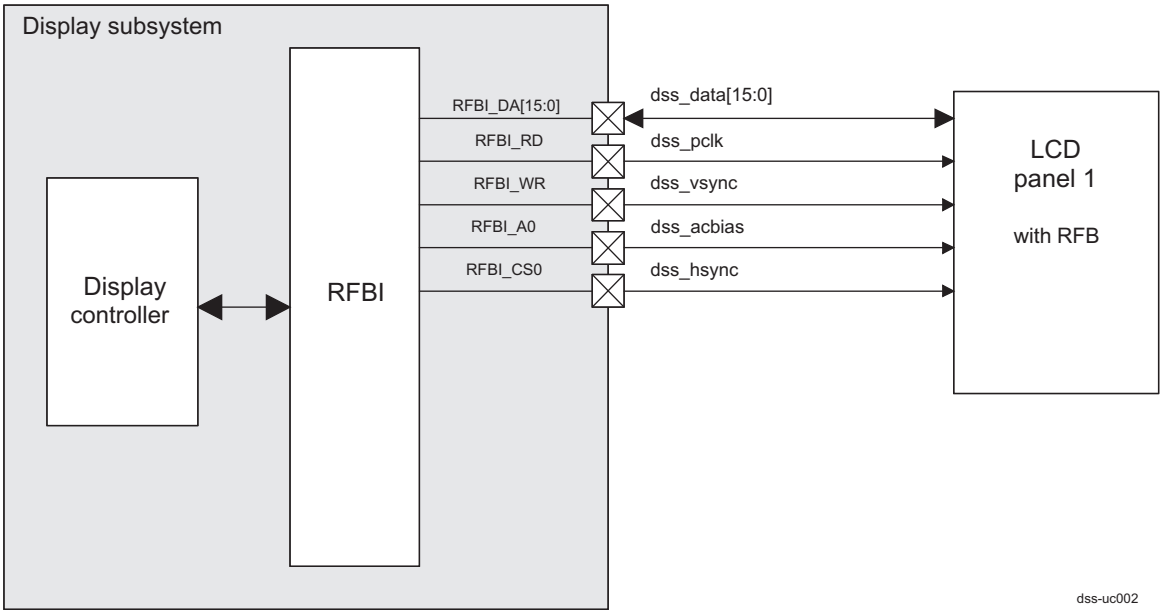
- Input Image: VGA 640*480 pixels , UYVY 4:2:2 format
- Output Image Pixel: QVGA 320*240 pixels , RGB18-666
- Pixel Clock (PCLK): 6 MHz to allow a video display of 60 frames per second
- No DMA requests to the system DMA, but the display subsystem uses its DMA controller to access the data into SDRAM
- VSYNC, SYNCLOST, VID1FIFOUNDERFLOW, and VID1ENDWINDOW interrupts used to monitor the video1 channel

12.6.5.2 Environment

In this use case, the display subsystem is connected to a QVGA LCD panel by the parallel interface.

Figure 12-170 details the OMAP device connections with the LCD panel.

Figure 12-170. Environment



12.6.5.2.1 LCD panel Features

The LCD panel includes the following features:

- Display format 320(H) x 240(V)
- Color mode: 18 bit/pixel R:5 G:6 B:5 (65 536 colors)

12.6.5.3 Data Path

In this use case, the display subsystem manages the following:

- Video stream (UYVY 4:2:2) between the SDRAM controller (SDRC) and the display controller through the internal DMA controller (L3 interconnect)
- Video stream (RGB16) between the display controller and the parallel module
- Video stream (parallel data) between the parallel module and the QVGA LCD panel

Figure 12-171 shows the data flow inside the display subsystem highlighted in red.

Figure 12-171. Display Subsystem Data Flow

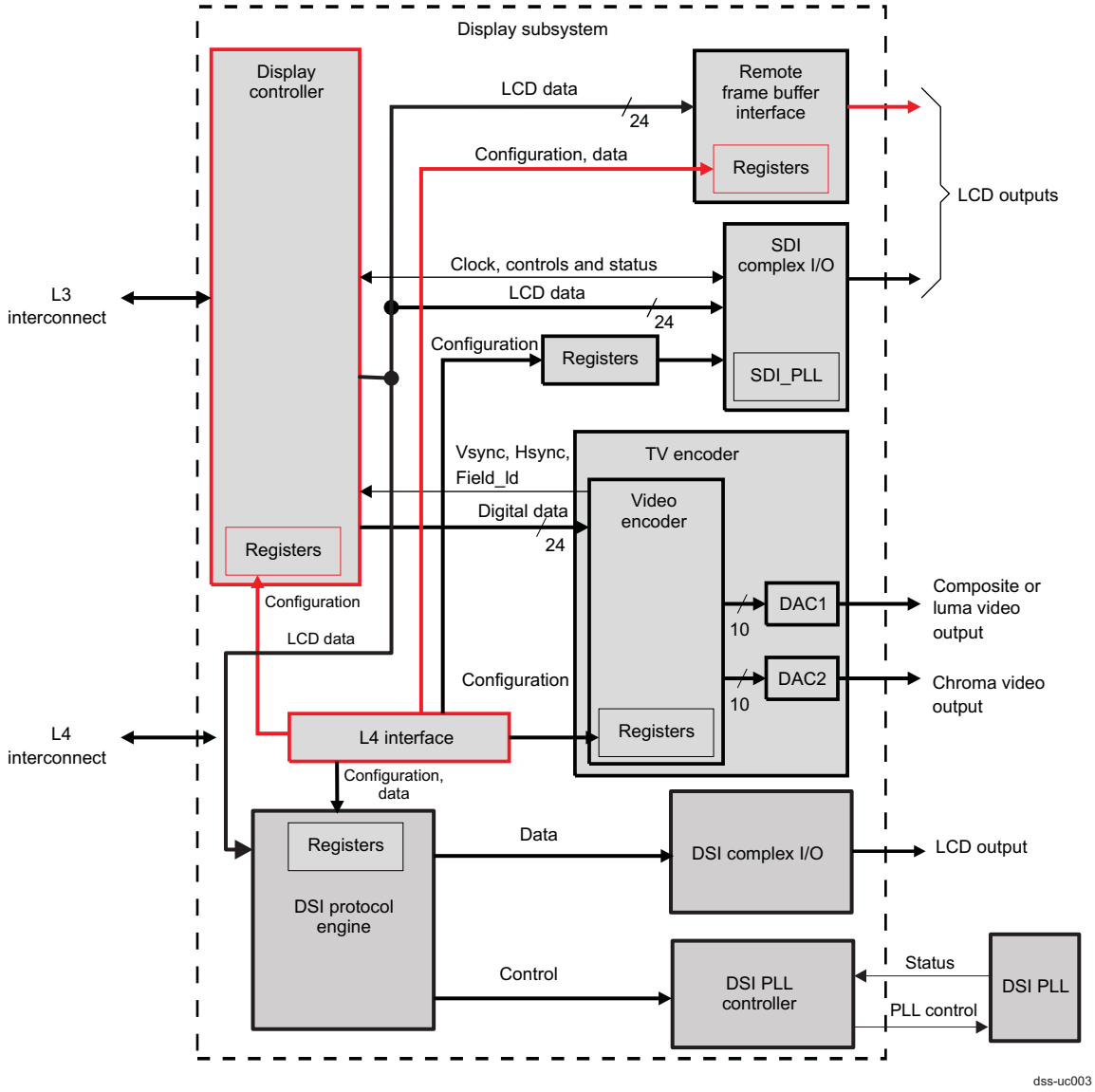
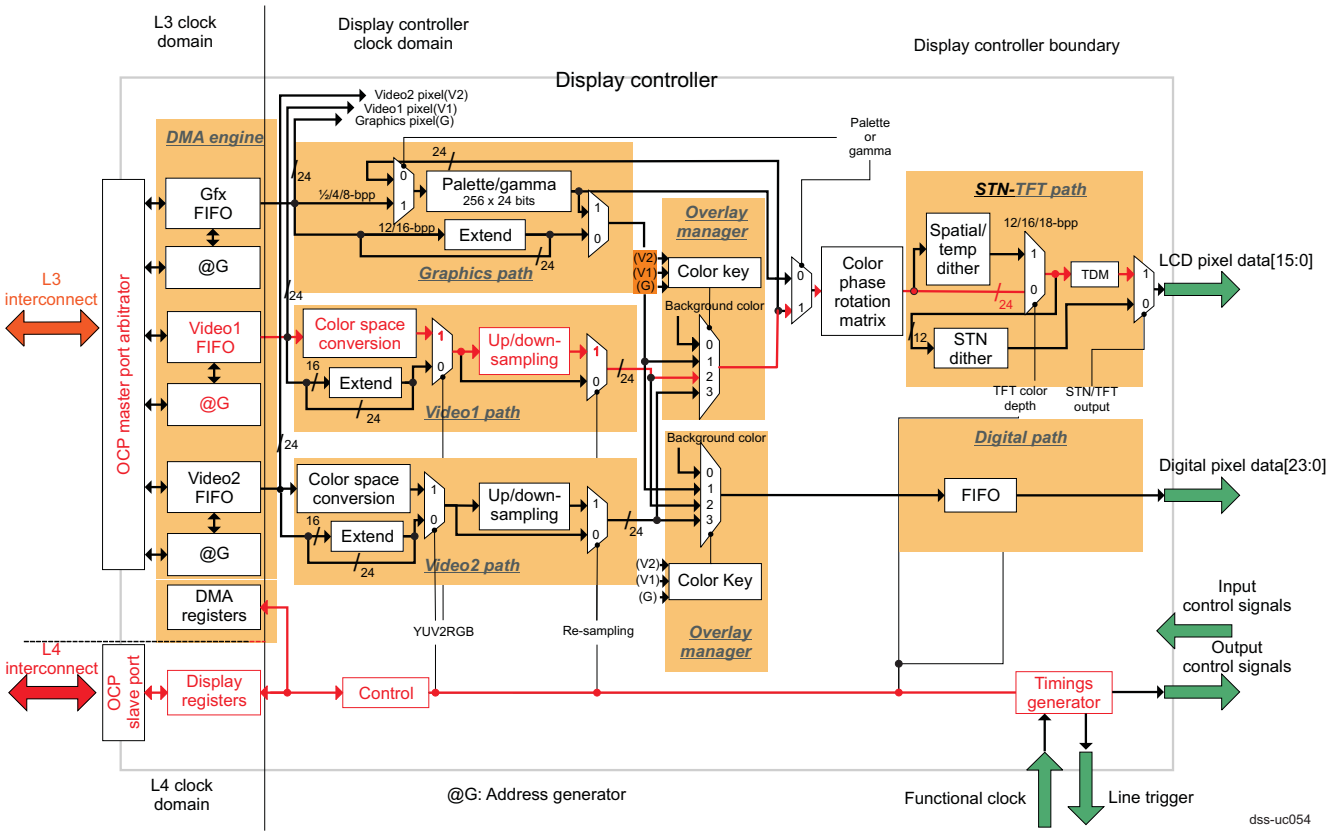


Figure 12-172 shows the data flow inside the display controller submodule highlighted in red.

Figure 12-172. Display Controller Data Flow



12.6.5.4 Programming Flow

This section details the programming flow of the display panel configuration for the camcorder use case. The display panel is configured in 3 steps (see Figure 12-173).

Figure 12-173. Display Panel Configuration for the Camcorder Use Case

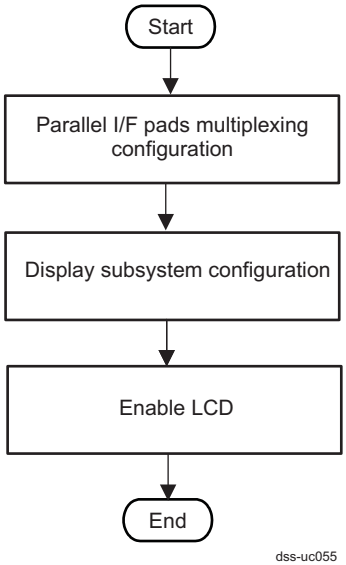
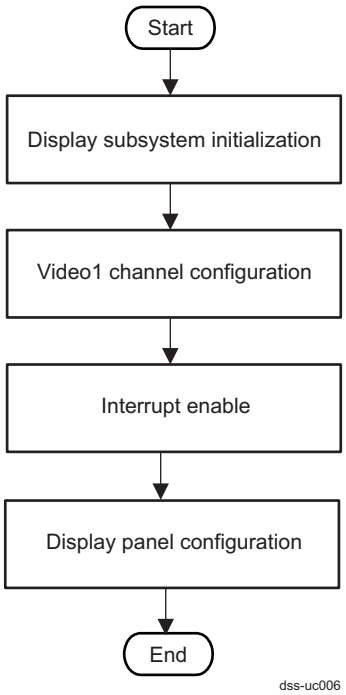


Figure 12-174 shows the programming flow of the display subsystem initial configuration, which consists of four steps:

Figure 12-174. Display Subsystem Configuration for Camcorder Use Case



12.6.5.4.1 Pads Multiplexing Configuration

All parallel pads are multiplexed in mode 1 on the dss_data pins:

Table 12-83 shows all the System Control Module registers.

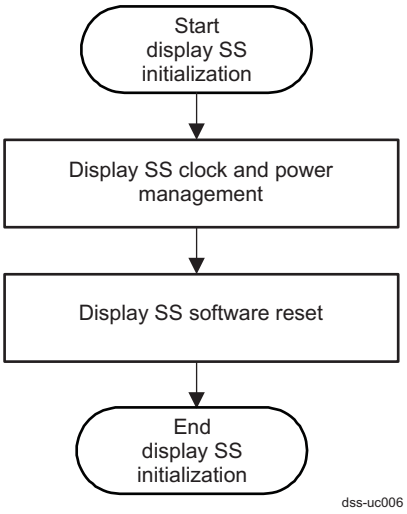
Table 12-83. Registers Print for QVGA LCD panel Pads Multiplexing Configuration

Register Name	Address	Value	Value Description
SCM.CONTROL_PADCONF_DSS_PCLK	0x4800 20D4	0x0110 0110	Configure dss_pclk and dss_hsync pads in mode 0
SCM.CONTROL_PADCONF_DSS_VSYNC	0x4800 20D8	0x0100 0110	Configure dss_vsync and dss_acbias pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA0	0x4800 20DC	0x0100 0100	Configure dss_data0 and dss_data1 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA2	0x4800 20E0	0x0100 0100	Configure dss_data2 and dss_data3 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA4	0x4800 20E4	0x0100 0100	Configure dss_data4 and dss_data5 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA6	0x4800 20E8	0x0100 0100	Configure dss_data6 and dss_data7 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA8	0x4800 20EC	0x0100 0100	Configure dss_data8 and dss_data9 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA10	0x4800 20F0	0x0100 0100	Configure dss_data10 and dss_data11 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA12	0x4800 20F4	0x0100 0100	Configure dss_data12 and dss_data13 pads in mode 0
SCM.CONTROL_PADCONF_DSS_DATA14	0x4800 20F8	0x0100 0100	Configure dss_data14 and dss_data15 pads in mode 0

12.6.5.4.2 Display Subsystem Initialization

Figure 12-175 details the display subsystem initialization.

Figure 12-175. Display Subsystem Initialization



12.6.5.4.2.1 Display Subsystem Clock and Power Management

The display subsystem clock (DSS1_ALWON_FCLK) is provided by the DPLL4 module in the PRCM (DPLL4_M4_X2). The software should set the following PRCM registers:

1. The DPLL4_ALWON_FCLKOUT clock is sourced by the system clock (SYS_CLK) and calculated as follows:

$$DPLL4\_ALWON\_FCLKOUT = \frac{SYS\_CLK(19.2\ MHz) \times 2 \times M\ (0xE1)}{N\ (0x9) + 1} = 864\ MHz$$

dss-uc020

   Set the M multiplier to 225 by setting the PRCM.CM_CLKSEL2_PLL[18:8] PERIPH_DPLL_MULT bit field to 0xE1.

   Set the N divider to 9 by setting the PRCM.CM_CLKSEL2_PLL[6:0] PERIPH_DPLL_DIV bit field to 0x09.

2. To follow the timing requirements for the use case, the DPLL4_M4_X2 clock frequency is calculated as follows:

$$DPLL4\_M4X2\_CLK = \frac{DPLL4\_ALWON\_FCLKOUT(864\ MHz)}{9} = 96\ MHz$$

dss-ucE007

   Set the DPLL4 ratio by programming to 0x9 the PRCM.CM_CLKSEL_DSS[4:0] CLKSEL_DSS1 bit field.

3. Enable the display subsystem functional clock (DSS1_ALWON_FCLK): Set the PRCM.CM_FCLKEN_DSS[0] EN_DSS1to 0x1.
4. Enable the display subsystem TV functional clock (DSS_TV_FCLK): Set the PRCM.CM_FCLKEN_DSS[2] EN_TV to 0x1.
5. Enable the display subsystem interface clock (DSS_L4_ICLK): Set the PRCM.CM_ICLKEN_DSS[0] EN_DSS to 0x1.

   Table 12-84 describes the registers from the PRCM to be configured for the display subsystem clock configuration step and the use case values.

Table 12-84. Registers Print for Display Subsytem Clock Management

Register Name	Address	Value	Value Description
PRCM.CM_FCLKEN_DSS	0x4800 4E00	0x0000 0005	Enable DSS1_LWON_FCLk and DSS_TV_FCLK
PRCM.CM_ICLKEN_DSS	0x4800 4E10	0x0000 0001	Enable DSS_L4_ICLK
PRCM.CM_CLKSEL2_PLL	0x4800 4D44	0x0000 E109	DPLL4 multiplier set to 0xE1 and divider set to 0x9
PRCM.CM_CLKSEL_DSS	0x4800 4E40	0x0001 0009	DPLL4 M4X2 divider set to 0x9

The display subsystem power management is programmed by the following PRCM registers:

1. Disable the autoidle mode: Set the PRCM.CM_AUTOIDLE_DSS register to 0x0.
2. Disable the DSS domain sleep dependency: Set the PRCM.CM_SLEEPDEP_DSS register to 0x0.
3. Disable the automatic transition between active and inactive: Set the PRMC.CM_CLKSTCTRL_DSS register to 0x0.

   Table 12-85 describes the registers from the PRCM to be configured for the display subsystem power management step and the use case values.

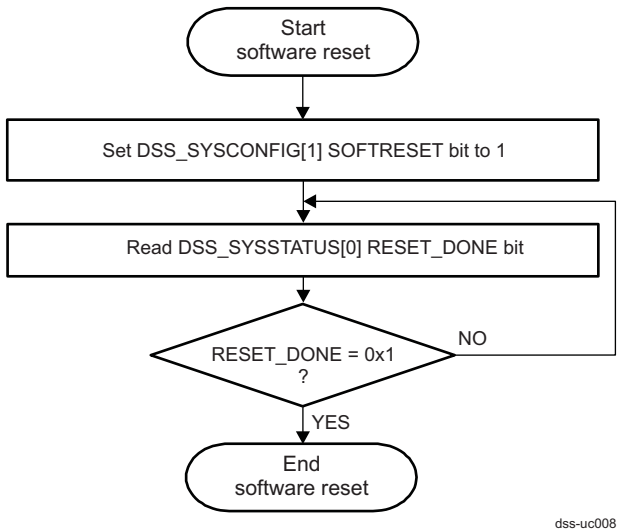
Table 12-85. Registers Print for Display Subsystem Power Management

Register Name	Address	Value	Value Description
PRCM.CM_AUTOIDLE_DSS	0x48004E30	0x0000 0000	Disable autoidle mode
PRCM.CM_SLEEPDEP_DSS	0x48004E44	0x0000 0000	Disable the DSS domain sleep dependency
PRCM.CM_CLKSTCTRL_DSS	0x48004E48	0x0000 0000	Disable the automatic clock state transition

12.6.5.4.2.2 Display Subsystem Software Reset

Perform a software reset as described in the following flowchart (see Figure 12-176).

Figure 12-176. Software Reset Flowchart



CAUTION

To update the RESETDONE status bit correctly, all interface and functional clocks, even for the TV output, must be provided to the display subsystem.

Table 12-86 describes the registers to be configured for the display subsystem software reset step and the use case values.

Table 12-86. Registers Print for Display Subsystem Software Reset

Register Name	Address	Value	Value Description
DSS_SYSCONFIG	0x4805 0010	0x0000 0002	Initiate a software reset. The DSS_SYSCONFIG[1] SOFTRESET is automatically reset by hardware
DSS_SYSSTATUS	0x4805 0014	0x0000 0001	The DSS_SYSSTATUS[0] RESETDONE is set to 1 when the reset sequence is done

12.6.5.4.3 Video1 Channel Configuration

The video1 channel is configured by programming the following steps:

1. Select the UYVY 4:2:2 format by programming the DSS.DISPC_VID1_ATTRIBUTES[4:1] VIDFORMAT bit field to 0xB.
2. Select the LCD data flow by programming the DSS.DISPC_VID1_ATTRIBUTES[16] VIDCHANNELOUT bit to 0x0.
3. Select the video DMA burst size to 16x32 by setting the DSS.DISPC_VID1_ATTRIBUTES[15:14] VIDBURSTSIZE to 0x2.
4. Program the image base address by setting the DSS.DISPC_VID1_BA0 register.
5. Program the FIFO thresholds as follows:

   To set the FIFO high threshold to 1023 bytes (maximum value allowed), set the DSS.DISPC_VID1_FIFO_THRESHOLD[27:16] VIDFIFOHIGHTHRESHOLD bit field to 0x3FF.

   Set the FIFO low threshold as described in the following equation:

$$\text{VIDFIFOLOWTHRESHOLD} = \text{VIDFIFOHIGHTHRESHOLD (0x3FF)} - \frac{\text{VIDBURSTSIZE(16x32)}}{8} = 959\ (0x3BF)$$

dss-ucE009

   Program the DSS.DISPC_VID1_FIFO_THRESHOLD[11:0] VIDFIFOLOWTHRESHOLD to 0x3BF.
6. Program the windows X-position to 0 (first column starting on the left edge of the screen) and the window Y-position to 0 (first row starting at the top of the screen): Set the DSS.DISPC_VID1_POSITION[10:0] VIDPOSX and the DSS.DISPC_VID1_POSITION[26:16] VIDPOSY registers to 0x0.
7. Program the window width to 320 by setting the DSS.DISPC_VID1_SIZE[10:0] VIDSIZEX to 0x13F.
8. Program the window height to 240 by setting the DSS.DISPC_VID1_SIZE[26:16] VIDSIZEY to 0xEF.
9. Program the picture width to 640 by setting the DSS.DISPC_VID1_PICTURE_SIZE[10:0] VIDORGSIZEX to 0x27F.
10. Program the picture height to 480 by setting the DSS.DISPC_VID1_PICTURE_SIZE[26:16] VIDORGSIZEY to 0x1DF.
11. Because the video format is in UYVY, software users must program the color conversion module as follows:

    To enable the color conversion module, set the DSS.DISPC_VID1_ATTRIBUTES[9] VIDCOLORCONVENABLE bit to 1.

    To select a limited range for color space conversion, set the DSS.DISPC_VID1_ATTRIBUTES[11] VIDFULLRANGE bit to 0.

    The DSS.DISPC_VID1_COEFi registers (with i = 0 to 4) have nine 11-bit coefficients defined for the programmable color space conversion block for video pipeline 1:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256} * \begin{bmatrix} RY & RCr & RCb \\ GY & GCr & GCb \\ BY & BCr & BCb \end{bmatrix} * \begin{bmatrix} Y - 16 \\ Cr - 128 \\ Cb - 128 \end{bmatrix}$$

dss-ucE010

The nine coefficient values for color space conversion from UYVY 4:2:2 to RGB are as follows:
RY = 298, RCb = 0, RCr = 409
GY = 298, GCb = (unsigned) (-11), GCr = (unsigned) (-208)
BY = 298, BCb = 517, BCr = 0

12. Configure the video1 channel for VRFB rotation settings as shown in Table 12-87:

Table 12-87. VRFB Rotation Configuration

Rotation	Registers	Value
0 degree	DSS.DISPC_VID1_BA0	VBA0
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	(2048 - iw) x ps + 1
90 degrees	DSS.DISPC_VID1_BA0	VBA90
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	(2048 - ih) x ps + 1
180 degrees	DSS.DISPC_VID1_BA0	VBA180
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	(2048 - iw) x ps + 1
270 degrees	DSS.DISPC_VID1_BA0	VBA270
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	(2048 - ih) x ps + 1

In the use case context, the input video format is UYVY for a VGA display (640 * 480). As a consequence, the VRFB configuration is as described in Table 12-88.

Table 12-88. VRFB Rotation Configuration for a VGA Display (UYVY format)

Rotation	Registers	Value
0 degree	DSS.DISPC_VID1_BA0	VBA0
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	6273
90 degrees	DSS.DISPC_VID1_BA0	VBA90
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	7233
180 degrees	DSS.DISPC_VID1_BA0	VBA180
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	6273
270 degrees	DSS.DISPC_VID1_BA0	VBA270
	DSS.DISPC_VID1_PIXEL_INC	1
	DSS.DISPC_VID1_ROW_INC	7233

For more information on VRFB module and settings, see the VRFB use case and tips section in , *Memory Subsystem*.

According to the rotation angle, program the DSS.DISPC_VID1_ATTRIBUTES[13] VIDROTATION and the DSS.DISPC_VID1_ATTRIBUTES[18] VIDROWREPEATENABLE bits as described in Table 12-89.

Table 12-89. Video Rotation Register Settings (UYVY Only)

Rotation	Registers Field Name	Registers Field Value
0 degree	DSS.DISPC_VID1_ATTRIBUTES[13] VIDROTATION	0x0
	DSS.DISPC_VID1_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0
90 degrees	DSS.DISPC_VID1_ATTRIBUTES[13] VIDROTATION	0x1
	DSS.DISPC_VID1_ATTRIBUTES[18] VIDROWREPEATENABLE	0x1
180 degrees	DSS.DISPC_VID1_ATTRIBUTES[13] VIDROTATION	0x2
	DSS.DISPC_VID1_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0
270 degrees	DSS.DISPC_VID1_ATTRIBUTES[13] VIDROTATION	0x3
	DSS.DISPC_VID1_ATTRIBUTES[18] VIDROWREPEATENABLE	0x1

13. Because the video size is in 640 * 480, the scaling unit must be used to downsample to a size of 320 * 240 using a 3 taps configuration.

    The video scaling block for the video pipeline 1 is set in downscaling for horizontal and vertical by setting the following bit field in DSS.DISPC_VIDn_ATTRIBUTES[6:5] :

    - VIDRESIZEENABLE bit field to 0x03
    - VIDHRESIZECONF bit field to 0x1
    - VIDVRESIZECONF bit field to 0x1

14. Program the downscaling of 2 by setting DSS.DISPC_VIDn_FIR to 0x0800 0800.

    All taps coefficients are programmed as in Table 12-90.

    Table 12-90 describes the registers to be configured for the video1 channel configuration step and the use case values.

Table 12-90. Registers Print for Video1 Channel Configuration

Register Name	Address	Value	Value Description
DISPC_VID1_ATTRIBUTES	0x4805 04CC	0x0000 8216 (for 0-degree rotation)	See the programming steps described above for details.
DISPC_VID1_BA0	0x4805 04BC	Base address value	Base address for an image in SDRAM. Depends on the rotation angle (see Table 12-88).
DISPC_VID1_FIFO_THRESHOLD	0x4805 04D0	0x03FF 03BF	FIFO high and low thresholds for DMA request generation
DISPC_VID1_POSITION	0x4805 04C4	0x0000 0000	No X-position and Y-position offsets
DISPC_VID1_SIZE	0x4805 04C8	0x00EF 013F	QVGA display:320*240
DISPC_VID1_PICTURE_SIZE	0x4805 04E4	0x01DF 027F	VGA display: 640*480
DISPC_VID1_CONV_COEF0	0x4805 0530	0x0199 012A	Color space conversion: RCR and RY coefficients
DISPC_VID1_CONV_COEF1	0x4805 0534	0x012A 0000	Color Space Conversion: GY and RCB coefficients
DISPC_VID1_CONV_COEF2	0x4805 0538	0x07F5 0730	Color Space Conversion: GCB and GCR coefficients
DISPC_VID1_CONV_COEF3	0x4805 053C	0x0000 012A	Color Space Conversion: BCR and BY coefficients
DISPC_VID1_CONV_COEF4	0x4805 0540	0x0000 0205	Color Space Conversion: BCB coefficient
DISPC_ROW_INC	0x4805 04D8	Image row increment	Image stride. Depends on the rotation angle (see Table 12-88).
DISPC_PIXEL_INC	0x4805 04DC	0x0000 0001	Pixel increment set to 1 because VRFB module is in charge of rotating the image
DISPC_VID1_FIR	0x4805 04E0	0x0800 0800	Vertical and horizontal ratio for downscaling

Table 12-90. Registers Print for Video1 Channel Configuration (continued)

DISPC_VID1_FIR_COEF_H_0	0x4805 04F0	0x24382400	Horizontal and vertical coefficients
DISPC_VID1_FIR_COEF_HV_0	0x4805 04F4	0x24382400	
DISPC_VID1_FIR_COEF_H_1	0x4805 04F8	0x28371FFE	
DISPC_VID1_FIR_COEF_HV_1	0x4805 04FC	0x28371F04	
DISPC_VID1_FIR_COEF_H_2	0x4805 0500	0x2C361BFB	
DISPC_VID1_FIR_COEF_HV_2	0x4805 0504	0x2C361B08	
DISPC_VID1_FIR_COEF_H_3	0x4805 0508	0x303516F9	
DISPC_VID1_FIR_COEF_HV_3	0x4805 050C	0x3035160C	
DISPC_VID1_FIR_COEF_H_4	0x4805 0510	0x11343311	
DISPC_VID1_FIR_COEF_HV_4	0x4805 0514	0x113433F7	
DISPC_VID1_FIR_COEF_H_5	0x4805 0518	0x1635300C	
DISPC_VID1_FIR_COEF_HV_5	0x4805 051C	0x163530F9	
DISPC_VID1_FIR_COEF_H_6	0x4805 0520	0x1B362C08	
DISPC_VID1_FIR_COEF_HV_6	0x4805 0524	0x1B362CFB	
DISPC_VID1_FIR_COEF_H_7	0x4805 0528	0x1F372804	
DISPC_VID1_FIR_COEF_HV_7	0x4805 052C	0x1F3728FE	
DISPC_VID1_FIR_COEF_V_0	0x4805 05E0	0x00000000	
DISPC_VID1_FIR_COEF_V_1	0x4805 05E4	0x000004FE	
DISPC_VID1_FIR_COEF_V_2	0x4805 05E8	0x000008FB	
DISPC_VID1_FIR_COEF_V_3	0x4805 05EC	0x00000CF9	
DISPC_VID1_FIR_COEF_V_4	0x4805 05F0	0x0000F711	
DISPC_VID1_FIR_COEF_V_5	0x4805 05F4	0x0000F90C	
DISPC_VID1_FIR_COEF_V_6	0x4805 05F8	0x0000FB08	
DISPC_VID1_FIR_COEF_V_7	0x4805 05FC	0x0000FE04	

12.6.5.4.4 Interrupts Enable

The interrupts enable stage is as follows:

1. To clear all display controller IRQs, set the DSS.DISPC_IRQSTATUS register to 0xFFFF.
2. To enable the VSYNC interrupt, set the DSS.DISPC_IRQENABLE[1] VSYNC bit to 1.
3. To enable the VID1FIFOUNDERFLOW interrupt, set the DSS.DISPC_IRQENABLE[10] VID1FIFOUNDERFLOW bit to 1.
4. To enable the ENDVID1WINDOW interrupt, set the DSS.DISPC_IRQENABLE[11] ENDVID1WINDOW bit to 1.
5. To enable the SYNCLOST interrupt, set the DSS.DISPC_IRQENABLE[14] SYNCLOST bit to 1.

Table 12-91 describes the registers to be configured for the interrupts enable step and the use case values.

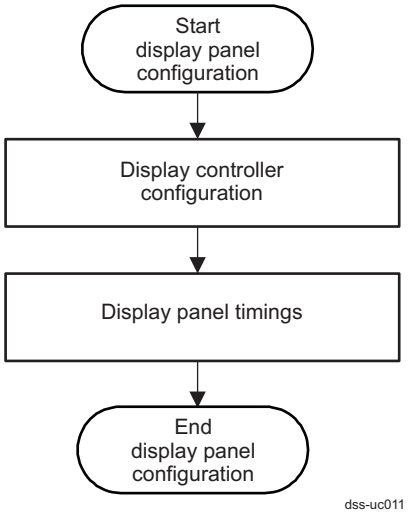
Table 12-91. Registers Print for Interrupts Enable

Register Name	Address	Value	Value Description
DISPC_IRQSTATUS	0x4805 0418	0xFFFF FFFF	Clear all DISPC interrupt requests. This register is automatically reset by hardware.
DISPC_IRQENABLE	0x4805 041C	0x0000 4C02	Enable the VSYNC, VID1FIFOUNDERFLOW, ENDVID1WINDOW, and SYNCLOST IRQs.

12.6.5.4.5 Display Panel Configuration

Figure 12-177 shows the structure of the display panel configuration stage:

Figure 12-177. Display Panel Configuration Flowchart



12.6.5.4.5.1 Display Controller Configuration

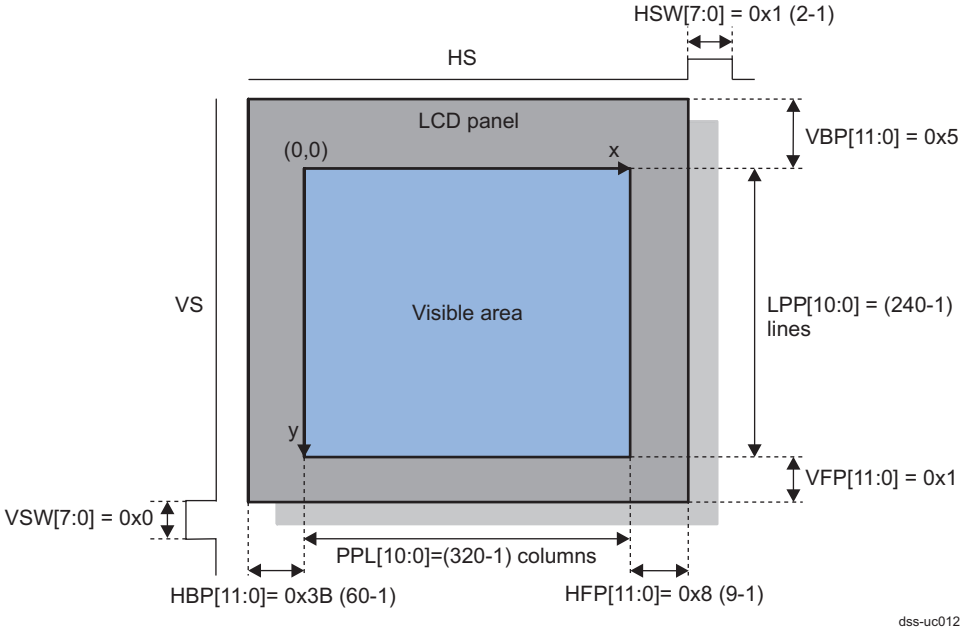
To interface the DISPC with the QVGA LCD panel (320*240), the following settings are required:

1. The QVGA LCD panel is an active matrix: set the DSS.DISPC_CONTROL[3] STNTFT bit to 1 (active matrix display operation mode).
2. Select 16-bit data output by setting the DSS.DISPC_CONTROL[9:8] TFTDATALINES bit field to 0x1.
3. Bypass the RFBI module by setting the DSS.DISPC_CONTROL[16:15] GPOUT[1:0] bit field to 0x3.
4. Select the color panel option by setting the DSS.DISPC_CONTROL[2]MONOCOLOR bit to 0x0.
5. Program the LCD enable signal active high by setting the DSS.DISPC_CONTROL[29] LCDENABLEPOL bit to 1.
6. Disable the free-running pixel clock by setting the DSS.DISPC_CONTROL[27] PCKFREEENABLE bit to 0.
7. Program the number of lines per panel to 240 by setting the DSS.DISPC_SIZE_LCD[26:16] LPP bit field to 0xEF (239 = 240 -1).
8. Program the number of pixels per line to 320 by setting the DSS.DISPC_SIZE_LCD[10:0] PPL bit field to 0x13F (319 = 320 -1).
9. The pallette module is bypassed: set the DSS.DISPC_CONFIG[2:1] LOADMODE bit field to 0x2 (frame data only).

12.6.5.4.5.2 Display Panel Timings

Figure 12-178 details the timing values description for the QVGA LCD panel.

Figure 12-178. QVGA LCD Panel Timings



The following steps define the timing generation of HSYNC/VSYNC signals based on the QVGA LCD panel specification:

1. Program the horizontal front porch to 9 by setting the DSS.DISPC_TIMING_H[19:8] HFP bit field to 0x8 (9-1).
2. Program the horizontal back porch to 60 by setting the DSS.DISPC_TIMING_H[31:20] HBP bit field to 0x3B (60-1).
3. Program the horizontal synchronization pulse width to 2 by setting the DSS.DISPC_TIMING_H[7:0] HSW bit field to 0x1 (2-1).
4. Program the vertical front porch to 1 by setting the DSS.DISPC_TIMING_V[19:8] VFP bit field to 1.
5. Program the vertical back porch to 5 by setting the DSS.DISPC_TIMING_V[31:20] VBP bit field to 5.
6. Program the vertical synchronization pulse width to 0 by setting the DSS.DISPC_TIMING_V[7:0] VSW bit field to 0.
7. For QVGA LCD panel, the line clock signal is active high and inactive low. As a consequence, invert HSYNC by setting the DSS.DISPC_POL_FREQ[13] IHS bit to 0.
8. For QVGA LCD panel, the frame clock signal is active high and inactive low. As a consequence, invert VSYNC by setting the DSS.DISPC_POL_FREQ[12] IVS bit to 0.
9. Drive HSYNC and VSYNC on the falling edge of PCLK by setting the DSS.DISPC_POL_FREQ[16] RF bit to 0 and the the DSS.DISPC_POL_FREQ[17] ONOFF bit to 0.
10. Drive the pixel data on the rising edge of PCLK by clearing the DSS.DISPC_POL_FREQ[14] IPC bit to 0.
11. Set the data enable active high by clearing the DSS.DISPC_POL_FREQ[15] IEO bit to 0.

The visible pixel count is 320 * 240 = 76 800 pixels.

The Total Frame Pixel Count (with the values given above) is (320+8+59+1) x (240+5+1+0) = 95 448 pixels.

The camcorder application displays 60 frame per second.

The pixel clock required for 60 fps is: 95 448 x 60 Hz = 5,72688 MHz.

On the other hand, the pixel clock is defined by the following equation:

$$PixelClock = \frac{Functional\ Clock}{LCD\ factor \times PCD\ factor}$$
dss-ucE013

To program the pixel clock to 5.72688 MHz, software users must set:

1. Set the DSS.DISPC_DIVISOR[23:16] LCD bit field to 0x2.

$$LogicClock = \frac{DSS1\_ALWON\_FCLK}{2} = 48\ Mhz$$
dss-ucE014

2. Set the DSS.DISPC_DIVISOR[7:0] PCD bit field to 0x8.

$$PixelClock = \frac{LogicClock}{8} = 6\ Mhz$$
dss-ucE015

Table 12-92 describes the registers to be configured for the display panel configuration step and the use case values.

Table 12-92. Registers Print for Display Panel Configuration

Register Name	Address	Value	Value Description
DISPC_CONTROL	0x4805 0440	0x0001 8108	See Section 12.6.5.4.5.1, *Display Controller Configuration*, for more details.
DISPC_SIZE_LCD	0x4805 047C	0x00EF 013F	QVGA display: 320 * 240
DISPC_CONFIG	0x4805 0444	0x0000 0004	Disable the palette
DISPC_TIMING_H	0x4805 0464	0x03B0 0801	Program horizontal timings
DISPC_TIMING_V	0x4805 0468	0x0050 0100	Program vertical timings
DISPC_POL_FREQ	0x4805 046C	0x0000 0000	Program the HSYNC and VSYNC signals for QVGA LCD panel
DISPC_DIVISOR	0x4805 0470	0x0002 0008	Set the divisors to program the PCLK frequency at 6 MHz.

12.6.5.4.6 LCD Enable

The LCD enable step is composed of the following stages:

1. To enable the video1 layer, set the DSS.DISPC_VID1_ATTRIBUTES[0] VIDENABLE bit to 1. The video DMA engine fetches encoded pixels from the system memory only when the video layer is enabled.
2. To enable the display controller output, set the DSS.DISPC_CONTROL[0] LCDENABLE bit to 1.
3. To account for all DISPC configuration done, set the DSS.DISPC_CONTROL[5] GOLCD bit to 1 to indicate that all shadow registers of the video1 pipeline connected to the LCD output are latched by the hardware. The hardware can now update the internal registers at the VFP start period and the image displayed on the panel.

Table 12-93 describes the registers to be configured for the LCD enable step and the use case values.

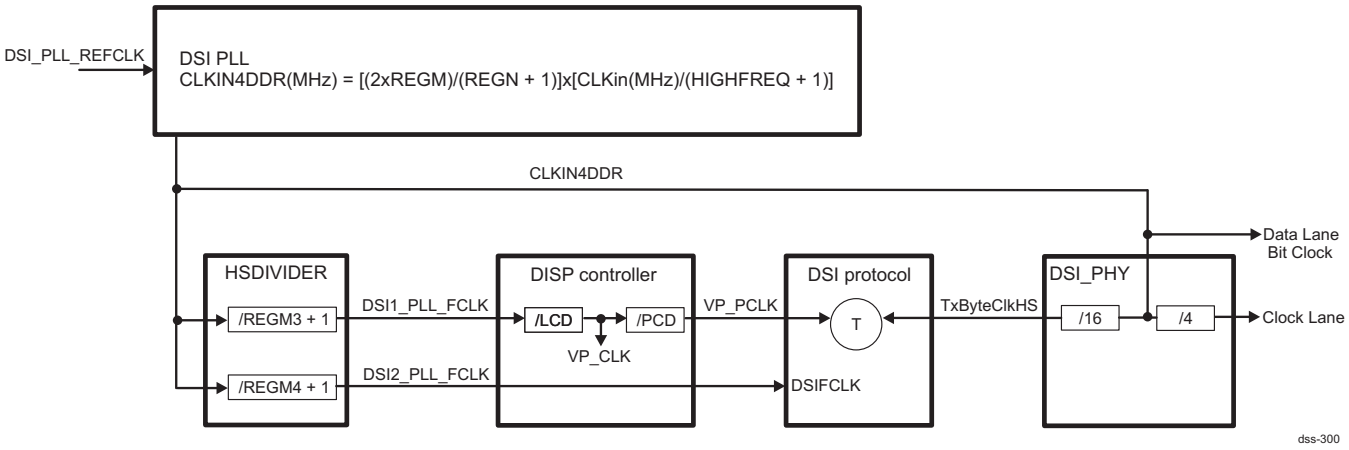
Table 12-93. Registers Print for LCD Enable

Register Name	Address	Value	Value Description
DISPC_CONTROL	0x4805 0440	0x0001 8129	Set DISPC_CONTROL[0] LCDENABLE and DISPC_CONTROL[5] GOLCD bits to 1.
DISPC_VID1_ATTRIBUTES	0x4805 04CC	0x0000 83F7	Enable the DISPC_VID1_ATTRIBUTES[0] VIDENABLE.

12.6.6 How to Configure the DSI PLL in Video Mode

Figure 12-179 shows a global overview of the DSI clock tree when used in video mode.

Figure 12-179. DSI Clock Tree in Video Mode



The settings of the DSI PLL registers can be summarized by four equations.

Equation 1

$$N * T_{VP\_CLK} = T_L * T_{TxByteClkHS} \quad \text{dss-301} \tag{19}$$

where

$T_L = T_{HS} + HSA_{DSI} + T_{HE} + HFP_{DSI} + (WC + 6)/NDL + HBP_{DSI}$

N is an integer

NDL: Number of data lane

WC: Word count or payload in bytes

T_{HS} and T_{HE} are equal to 4/NDL and 0 if they are disabled.

HSA_{DSI} is HSA period in video mode.

T_{HS} is the length of HSYNC start short packet in number of byte clock cycles (TxByteClkHS).

T_{HE} is the length of HSYNC end short packet in number of byte clock cycles (TxByteClkHS).

HBP_{DSI} is HBP period in video mode.

HFP_{DSI} is HFP period in video mode.

NOTE: HSA_{DSI} timing is not used and does not have to be programmed when HE short packet is not generated.

Equation 2

$$R = T_{TxByteClkHS} / T_{VP\_PCLK} \quad \text{dss-302} \tag{20}$$

To synchronize DISPC and DSI protocol engine, users should follow the ratio R between TxByteClkHS and VP_PCLK as listed in Table 12-94.

Table 12-94. Ratio R

Number of Data Lanes	Pixel Format	Ratio R
1	16-bits pixel	1/2
1	18-bits pixel	4/9
1	24-bits pixel	1/3

Table 12-94. Ratio R (continued)

Number of Data Lanes	Pixel Format	Ratio R
2	16-bits pixel	1
2	18-bits pixel	8/9
2	24-bits pixel	2/3

All cases are covered by:

$F_{VP\_PCLK} * bits\_per\_pixel = F_{TxByteClkHS} * NDL * 8$

dss-310

Equation 3

$$(HSA_{DISPC} + HFP_{DISPC} + PPL + HBP_{DISPC}) * T_{VP\_PCLK} = (4/NDL + HFP_{DSI} + (WC + 6)/NDL + HBP_{DSI}) * T_{TxByteClkHS}$$ dss-303 (21)

Equation 4

$$HFP_{DSI} = ((HFP_{DISPC} * bits\_per\_pixel) / (NDL * 8)) - (2 / NDL)$$ dss-309 (22)

Example

The desired performances are:

- Clock lane at 150 MHz
- RGB24-888
- 1-data lane
- LCD size 480*640 with $HSA_{\_DISP} = HFP_{\_DISP} = HBP_{\_DISP}$ =20, $VSA_{\_DISP} = VFP_{\_DISP} = VBP_{\_DISP}$ =2

Step 1. **Determine REGM and REGN**

To obtain correct stability, Fint should be kept between 0.75 MHz and 2.1 MHz. In this case, Fint is maintained at 2 MHz. For more information, refer to the DSI PLL programming model.

$$REGN = (F_{DSI\_PLL\_REFCLK} / F_{int}) – 1$$

$$REGN = 12$$

$$REGM = (REGN+1)*F_{CLKIN4DDR} / (2 * F_{DSI\_PLL\_REFCLK})$$

$$REGM = 150$$ dss-318

Where DSS2_ALWON_FCLK= 26 MHz is used as a reference clock for $F_{DSI\_PLL\_REFCLK}$

Step 2. **Determine VP_PCLK and TxByteClkHS clocks.**

TxByteClkHS frequency is equal to 37.5 MHz. With ratio R equal to 1/3, VP_PCLK frequency is equal to 12,5 MHz. The frame rate can be estimated by:

$$Frame\ rate = F_{VP\_PCLK} / (HSA_{DISPC} + HFP_{DISPC} + PPL + HBP_{DISPC}) * (VSA_{DISPC} + VFP_{DISPC} + LPP + VBP_{DISPC})$$

$$Frame\ rate = 12,5\ MHz / (540) * (646)$$

$$Frame\ rate = 35,83\ frame/sec$$ dss-323

Step 3. **Determine LCD, PCD and REGM3**

$$T_{CLKIN4DDR} = T_{TxByteClkHS} / 16 = T_{VP\_PCLK} / ((REGM3 + 1) * LCD * PCD)$$

$$((REGM3 + 1) * LCD * PCD) = 16 * 3$$ dss-315

If LCD and PCD are set to 1 and 3 respectively, REGM3 is equal to 15.

Step 4. **Verify N as integer**

Firstly, TL must be determined

$$(HSA_{DISPC} + HFP_{DISPC} + PPL + HBP_{DISPC}) * T_{VP\_PCLK} / T_{TxByteClkHS} = T_L = 1620$$ dss-316

From **Equation 1**,

$$N*T_{VP\_CLK} = T_L*T_{TxByteClkHS}$$

$$N= T_{TxByteclkHS}*T_L/T_{VP\_CLK}=T_L/(R*PCD)$$

$$N = 14580$$

N is an integer.

Step 5. **Determine HFP and HBP of the DSI protocol engine**

From **Equation 3**,

$$(HSA_{DISPC} + HFP_{DISPC} + PPL + HBP_{DISPC}) * T_{VP\_PCLK} / T_{TxByteClkHS}) - (4/NDL + (WC + 6)/NDL) = HFP_{DSI} + HBP_{DSI}$$

$$HFP_{DSI} + HBP_{DSI} = 170$$

dss-317

From **Equation 4**,

$HFP_{DSI} = ((HFP_{DISPC} * bits\_per\_pixel) / (NDL * 8)) - (2 / NDL)$

$$HFP_{DSI} = 58$$

$$HBP_{DSI} = 170 - 58 = 112$$

dss-321

12.6.7 DSI Video Mode Using the DISPC Video Port

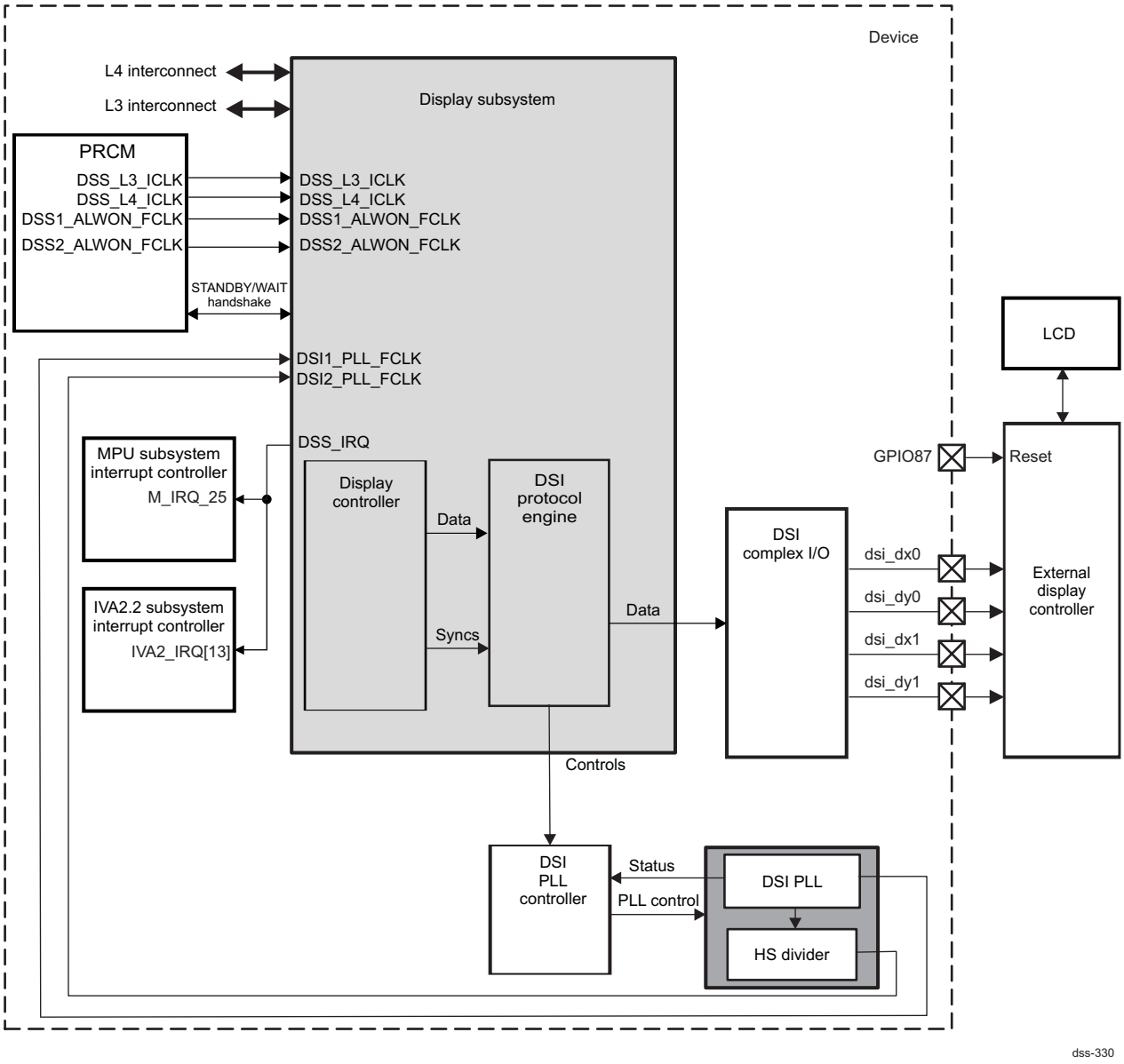
This section details the basic programming model of video mode using the DISPC video port.

The DSI interface is connected to an external MIPI display controller and the following parameters are used:

- 1 data lane: NDL = 1
- Clock lane at 150 MHz (DSI_DDR_CLK)
- LCD size is 640 × 480:
  - 480 pixels per line (PPL)
  - 680 lines per panel (LPP)
- Display controller input format: YUV
- Display controller output format: RGB888, that is 24 bits per pixel (BPP)
- Word Count: WC = 3 × PPL
- DSS2_ALWON_FLCK=26 MHz is used as a reference clock for DSI PLL
- Virtual channel 0 (VC0) is used for video mode.
- Interleaving is not used.
- It is assumed that all modules used in these programming models are in after-POR state.

Figure 12-180 is an overview of the connections in the display subsystem.

Figure 12-180. Overview



The steps listed in Table 12-95 are described in the following sections.

Table 12-95. Main Steps

Steps	Section
Configure DSS clocks	Section 12.6.7.1, *Display Subsystem Clock Configuration*
Configure the DSI and DSI PLL	Section 12.6.7.2, *Configure DSI, DSI PLL and Complex I/O*
Configure the external MIPI display controller	Section 12.6.7.3, *Initialization of the External MIPI Display Controller*
Configure the DISPC	Section 12.6.7.4, *Configure the DISPC*
Enable video mode using the DISPC video port	Section 12.6.7.5, *Enable Video Mode Using the DISPC Video Port*

The programming model must be followed in the order of the following sections.

12.6.7.1 Display Subsystem Clock Configuration

Table 12-96 lists the steps required to enable the clocks.

Table 12-96. PRCM Registers

Steps	Registers	Value
Set the divided DPLL value for DSS1.	CM_CLKSEL_DSS[4:0] CLKSEL_DSS1	0x9
Disable autoidle mode.	CM_AUTOIDLE_DSS[31:0]	0x0
Domain sleep is disabled.	CM_SLEEPDEP_DSS[31:0]	0x0
Automatic transition between active and inactive are disabled.	CM_CLKSTCTRL_DSS[31:0]	0x0
Enable DSS1, DSS2 and TV clock (DSS1_ALWON_FCLK, DSS2_ALWON_FCLK and TV_CLK). TV_CLK is only need for correct Reset.	CM_FCLKEN_DSS[31:0]	0x7
Enable the subsystem interface clock (DSS_L3_ICLK and DSS_L4_ICLK).	CM_ICLKEN_DSS[31:0]	0x1

12.6.7.2 Configure DSI, DSI PLL and Complex I/O

12.6.7.2.1 Reset DSI Modules

Table 12-97 lists the steps required to reset the DSI modules.

Table 12-97. Resets

Steps	Registers	Value
Reset IRQ status.	DSI_IRQSTATUS[31:0]	0x0
OCP and functional clock are maintained during wakeup, smart idle, and reset DSI.	DSI_SYSCONFIG[31:0]	0x312
Wait until RESET_DONE ≠ 0.	DSI_SYSSTATUS[0] RESET_DONE	Read 0x1

12.6.7.2.2 Set Up DSI DPLL

Table 12-98 lists the steps required to configure DSI PLL.

Table 12-98. DSI PLL Configuration Registers

Steps	Registers	Value
Turn on PLL and HSDIVIDER.	DSI_CLK_CTRL[31:30] PLL_PWR_CTRL	0x2
Wait until PLL_PWR_STATUS = 0x2.	DSI_CLK_CTRL[29:28] PLL_PWR_STATUS	Read 0x2
See the calculation following this table.	DSI_PLL_CONFIGURATION1[26:23] DSIPROTO_CLK_DIV	5
See the calculation following this table.	DSI_PLL_CONFIGURATION1[22:19] DSS_CLOCK_DIV	15
See the calculation following this table.	DSI_PLL_CONFIGURATION1[18:8] DSI_PLL_REGM	150
See calculation following this table.	DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN	12
Enable PLLStopMode.	DSI_PLL_CONFIGURATION1[0]	0x1
The DSI protocol engine clock divider, DSS clock divider, CLKIN4DDR, and PLL reference clock are enabled. PLL internal reference frequency is between 1.75 and 2.1MHz.	DSI_PLL_CONFIGURATION2[31:0]	0x5600E
Manual mode	DSI_PLL_CONTROL[31:0]	0x0
Request PLL locking sequence.	DSI_PLL_GO[0] DSI_PLL_GO	0x1
Read until DSI_PLL_GO = 0	DSI_PLL_GO[0] DSI_PLL_GO	Read 0x0
PLL is locked.	DSI_PLL_STATUS[1] DSI_PLL_LOCK	Read 0x1

Table 12-98. DSI PLL Configuration Registers (continued)

Steps	Registers	Value
Turn on PLL and HSDIVIDER; the DSI functional clock > 30-MHz sync is rising/rising; the DSIStopClk signal is automatically asserted/deasserted; the L3_ICLK clock to the DSI complex I/O is not gated; and LPCLKDIVISOR = 8	DSI_CLK_CTRL[31:0]	0x8024 4008

1. Calculate the divider value for the DSI protocol engine clock source:

   RegM4 = FCLKIN4DDR/DSI2_PLL_FCLK - 1

   FCLKIN4DDR = 4 x FCLKIN

   RegM4 = 5

2. Determine LCD, PCD, and REGM3:

   Calculate the divider value for DSS clock source: Same as Step 3.

   $$RegM3 = ((BPP \times 2)/(DISPC\_LCD \times DISPC\_PCD \times NDL)) - 1$$

   $$RegM3 = 15$$

   dss-E127

3. Calculate N Divider for PLL:

   $$FCLKIN4DDR = FCLKIN \times 4$$

   $$RegN = (FDSI\_PLL\_REFCLK / F\,INt) - 1$$

   $$FDSI\_PLL\_REFCLK = 26\ MHz\ (system\ clock)$$

   $$Fint = 2\ MHz\ (reduce\ PLL\ lock\ time)$$

   $$RegN = 12$$

   dss-E128

4. Calculate M divider for PLL:

   $$RegM = ((RegN + 1) \times (FCLKIN4DDR/(2 \times FDSI\_PLL\_REFCLK)))$$

   $$FCLKIN4DDR = 4 \times 150\ MHz$$

   $$RegM = 150$$

   dss-E129

12.6.7.2.3 Switch to DSI PLL Clock Source

Select the DSI_PLL1 clock as the DISPC functional clock, and select the DSI_PLL2 clock as the DSI functional clock: Set DSS_CONTROL to 0x3.

12.6.7.2.4 Set Up DSI Protocol Engine

12.6.7.2.4.1 Set Up DSI Control Registers

Table 12-99 lists the steps to set up the DSI control registers. Table 12-100 lists the steps to set up the DSI complex I/O registers.

Table 12-99. DSI Control Registers

Steps	Registers	Value
Enable SYNCLOST event.	DSI_IRQENABLE[31:0]	0x4 0000
Set PACKET_SENT_IRQ_EN.	DSI_VC0_IRQENABLE[31:0]	0x4
While the HSYNC START pulse is detected, the associated short packet HSYNC START is generated.	DSI_CTRL[17] VP_HSYNC_START	0x1
While the VSYNC START pulse is detected, the associated short packet VSYNC START is generated.	DSI_CTRL[15] VP_VSYNC_START	0x1

Table 12-99. DSI Control Registers (continued)

Steps	Registers	Value
Set the trigger reset mode to *immediate.*	DSI_CTRL[14] TRIGGER_RESET_MODE	0x1
Activate two line buffers.	DSI_CTRL[13:12] LINE_BUFFER	0x2
HSYNC signal on the video port is active high.	DSI_CTRL[10] VP_HSYNC_POL	0x1
VSYNC on the video port is active high.	DSI_CTRL[11] VP_VSYNC_POL	0x1
Set the Data Enable signal on the video port as active high.	DSI_CTRL[9] VP_DE_POL	0x1
Set the size of the video port data bus for the RGB format: 0x2 for RGB 888.	DSI_CTRL[7:6] VP_DATA_BUS_WIDTH	0x2
VP_CLK_RATIO is not used if video port is used to provide data in video mode	DSI_CTRL[4] VP_CLK_RATIO	0x0
Set the arbitration scheme for granting the VC pending ready requests in the TX FIFO as Sequential Scheme.	DSI_CTRL[3] TX_FIFO_ARBITRATION	0x1
Enable the ECC check for the received header.	DSI_CTRL[2] ECC_RX_EN	0x1

Table 12-100. DSI Complex I/O Registers

Steps	Registers	Value
GOBIT, complex I/O power ON, select data, and clock position	DSI_COMPLEXIO_CFG1	0x4800 0032
Clear Reg	DSI_COMPLEXIO_IRQSTATUS	0xC3F39CE7
Disable IRQ	DSI_COMPLEXIO_IRQENABLE	0x0
Enable I/F	DSI_CTRL[0] IF_EN	0x1
Disable I/F	DSI_CTRL[0] IF_EN	0x0
Wait until IF_EN = 0	DSI_CTRL[0] IF_EN	Read 0x0
Enable Low Power clock	DSI_CLK_CTRL[20] LP_CLK_ENABLE	0x1
Reset is done.	DSI_COMPLEXIO_CFG1[29] RESET_DONE	Read 0x1
Power control is on.	DSI_COMPLEXIO_CFG1[26:25] PWR_STATUS	Read 0x1
Reset is complete.	DSI_SYSSTATUS[0] RESETDONE	Read 0x1

12.6.7.2.4.2 Configure DSI Timing and Virtual Channels

Table 12-101 lists the steps to configure DSI timing and the virtual channels.

Table 12-101. DSI Timing Registers

Steps	Registers	Value
STOP_STATE_COUNTER_IO = 0x999	DSI_TIMING1[31:0]	0x0000 0999
HS_TX_TO_X8, HS_TX_TO_COUNTER = 0x0FD2, LP_RX_TO_X16, LP_RX_TO_COUNTER = 0x00CD	DSI_TIMING2[31:0]	0x2FD2 40CD
(HSA<<24)|(HFP<<12)|HBP	DSI_VM_TIMING1[31:0]	0x0000 700A
(WINDOW_SYNC<<24)|(VSA<<16)|(DSI_VFP <<8)|VBP	DSI_VM_TIMING2[31:0]	0x0401 0101
(TL<<16)|DSI_VACT	DSI_VM_TIMING3[31:0]	0x05BB 0280
(ENTER_HS_MODE_LATENCY<<16)|EXIT_H S_MODE_LATENCY	DSI_VM_TIMING7[31:0]	0x0000 C000A
DDR_CLK_PRE<<8|DDR_CLK_POST	DSI_CLK_TIMING[31:0]	0x0000 0F0B
HS speed, ECC generation for the Transmit, Enable CheckSum generation for the Payload.	DSI_VC0_CTRL[31:0]	0x2080 0390

- Freq TxByteClkHS :

$$FHSB = FCLKIN4DDR / 16$$

$$FVPP = FCLKIN4DDR / ((RegM3 + 1) \times DISPC\_LCD * DISPC\_PCD)$$

$$FVP = FCLKIN4DDR / (RegM3 + 1)$$

dss-E130

- Length of the line in video mode in Nb of byte clock cycles (TxByteClkHS):

$$TL = FHSB / FVPP \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP)$$

$$TL1f = (BPP / (8 \times NDL)) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP)$$

dss-E131

- Blanking periods (HBP + HFP) in DSI are calculated based on the following formula:

$$(DISPC\_HSA + DISPC\_HBP + PPL + DISPC\_HFP) \times Fppi = (HS + HBP + ((WC + 6) / NDL) + HFP) \times Fvp$$

$$HBP + HFP = (TVPP / THSB) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP) - (HS + (WC + 6) / NDL))$$

$$HBPplusHFP = (FHSB / FVPP) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP) - (HS + WC + 6) / NDL$$

$$HBPplusHFPf = ((FHSB / FVPP) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP)) - ((HS + WC + 6) / NDL)$$

$$HFP = (DISPC\_HFP \times BPP) / (NDL \times 8) - (2 / NDL)$$

$$HBP = HBPplusHFP - HFP$$

dss-E132

12.6.7.2.5 Configure DSI_PHY

For the calculation of DSI_PHY timing, see Table 12-102. See Section 12.4.3.2, *Clock Requirements*, for details on timing calculation.

Table 12-102. Calculate DSI_PHY Timing

Steps	Registers	Value
See Section 12.4.3.2	DSI_PHY_CFG0[31:24] THS_PREPARE	ceil(70 ns/DDR clock period) + 2
	DSI_PHY_CFG0[23:16] THS_PREPARE_THS_ZERO	ceil(175 ns/DDR clock period) + 2
	DSI_PHY_CFG0[7:0] THS_EXIT	ceil(145 ns/DDR clock period)
	DSI_PHY_CFG0[15:8] THS_TRAIL	ceil(60 ns/DDR clock period) + 5
	DSI_PHY_CFG2[7:0] TCLK_PREPARE	ceil(65 ns/DDR clock period)
	DSI_PHY_CFG1[7:0] TCLK_ZERO	ceil(260 ns/DDR clock period)
	DSI_PHY_CFG1[15:0] TCLK_TRAIL	ceil(60 ns/DDR clock period) + 2
	DSI_PHY_CFG120:16] TLPX_HALF	ceil(26.5 ns/DDR clock period)

NOTE: Keep Reserved bits at reset value in the DSI_PHY_CFG1 and DSI_PHY_CFG2 registers.

12.6.7.2.6 Drive Stop State

Table 12-103 lists the steps to Drive Stop State.

Table 12-103. Drive Stop State

Steps	Registers	Value
Force TX stop mode	DSI_TIMING[15] FORCE_TX_STOP_MODE_IO	0x1
Wait until FORCE_TX_STOP_MODE_IO = 0	DSI_TIMING[15] FORCE_TX_STOP_MODE_IO	Read 0x0

12.6.7.3 Initialization of the External MIPI Display Controller

1. Wait for the external MIPI display controller initialization after power up.
   In this example, the external MIPI display controller is reset using GPIO 87.
2. Configure the external MIPI display controller.

12.6.7.4 Configure the DISPC

12.6.7.4.1 Reset DISPC

Table 12-104 lists the step sequence to reset the DISPC.

Table 12-104. Reset DISPC

Steps	Registers	Value
Reset the DISPC.	DISPC_SYSCONFIG[1] SOFTRESET	0x1
DISPC is reset.	DISPC_SYSCONFIG[0] RESETDONE	Read 0x1
No standby: MStandby is never asserted.	DISPC_SYSCONFIG [13:12] MIDLEMODE	0x1
Disable idle mode.	DISPC_SYSCONFIG [4:3] SIDLEMODE	0x1
Disable all interrupts.	DISPC_IRQENABLE[31:0]	0x0

12.6.7.4.2 Configure DISPC Timing, Window, and Color

Table 12-105 lists the steps to configure the DISPC registers. Table 12-106 lists the steps to configure the color space coefficient registers.

Table 12-107 lists the steps to configure DISPC_CONTROL.

Table 12-105. Configure DISPC Registers

Steps	Registers	Value
Set horizontal timings.	DISPC_TIMING_H	(HBP – 1<<20)|(HFP – 1<<8)|HSA – 1
Set vertical timing.	DISPC_TIMING_V	(HBP – 1<<20)|(HFP – 1<<8)|HSA – 1
Set LCD – PCD display controller and pixel clock divisor.	DISPC_DIVISOR	LDC = %1 PXLCLK = %4
Set the size of the LCD – 1.	DISPC_SIZE_LCD	(LPP – 1 + DSI_VFP)<<16|(PPL – 1)
Set default RGB value when there is no data.	DISPC_DEFAULT_COLOR0	0XFF
Set video FIFO height and low threshold.	DISPC_VID1_FIFO_THRESHOLD	0xfc00c0
Set the X Y location of the upper left pixel (0 = 0) to the upper left corner.	DISPC_VID1_POSITION	0x0
Set size of the window.	DISPC_VID1_SIZE	(LPP – 1) <<16|(PPL – 1)
Set the size of the picture.	DISPC_VID1_PICTURE_SIZE	(LPP – 1) <<16|(PPL – 1)
Set the input address picture.	DISPC_VID1_BA0	0x–

Table 12-106. Configure Color Space Coefficient Registers

Comments	Registers	Value
RCR and RY coefficients	DISPC_VID1_CONV_COEF0	0X0199012A
GY and RCB coefficients	DISPC_VID1_CONV_COEF1	0X012A0000
GCB and GCR coefficients	DISPC_VID1_CONV_COEF2	0X079C0730
BCR and BY coefficients	DISPC_VID1_CONV_COEF3	0X0000012A
BCB coefficient	DISPC_VID1_CONV_COEF4	0X00000205

Table 12-107. Configure DISPC_CONTROL

Comments	Registers	Value
Enable pixel clock free-running.	DISPC_CONTROL[27] PCLKFREEENABLE	0x1
I/O pad mode selection: Bypass	DISPC_CONTROL[16] GPOUT1	0x1
I/O pad mode selection: Bypass	DISPC_CONTROL[15] GPOUT0	0x1
Normal mode selected	DISPC_CONTROL[11] STALLMODE	0x0
3 for RGB888	DISPC_CONTROL[9:8] TFTDATALINES	0x3
The hardware has finished updating the internal shadow registers of the pipeline(s) connected to the LCD output using the user values. The hardware resets the bit when the update completes.	DISPC_CONTROL[5] GOLCD	0x0
Active matrix display operation mode	DISPC_CONTROL[3] STNTFT	0x1
LCD interface is disabled.	DISPC_CONTROL[0] LCDENABLE	0x0

The input image is converted from UYVY into RGB (see Table 12-108) :

Table 12-108. Configure DISPC_VID1_ATTRIBUTES

Comments	Registers	Value
Row of VIDn will not be read twice.	DISPC_VID1_ATTRIBUTES[18] VIDROWREPEATENABLE	0x0
8 x 32-bit bursts	DISPC_VID1_ATTRIBUTES[15:14] VIDBURSTSIZE	0x1
Full range selected: Y is not modified before the color space conversion.	DISPC_VID1_ATTRIBUTES[11] VIDFULLRANGE	0x1
Enable color space conversion CbYCr to RGB.	DISPC_VID1_ATTRIBUTES[9] VIDCOLORCONVENABLE	0x1
UYVY	DISPC_VID1_ATTRIBUTES[4:1] VIDFORMAT	0xB
Video disabled	DISPC_VID1_ATTRIBUTES[0] VIDENABLE	0x0

12.6.7.5 Enable Video Mode Using the DISPC Video Port

Table 12-109 lists the steps to enable DISPC to send frames continuously.

Table 12-109. Enable DISPC

Steps	Registers	Value
Set up long packet header	DSI_VC0_LONG_PACKET_HEADER[31:0]	0x0005 A03E
Enable VC0.	DSI_VC0_CTRL[1] VC_EN	0x1
Enable IF.	DSI_CTRL[0] IF_EN	0x1
Wait until IF_EN ≠ 0.	DSI_CTRL[0] IF_EN	Read 0x0
Enable VID1.	DISPC_VID1_ATTRIBUTES[0] VIDENABLE	0x1
Enable LCD interface.	DISPC_CONTROL[0] LCDENABLE	0x1
Enable GOLCD.	DISPC_CONTROL[5] GOLCD	0x1
Wait until GOLCD = 0.	DISPC_CONTROL[5] GOLCD	Read 0x0

12.6.8 DSI Command Mode Using the DISPC Video Port

This section presents some generic use cases and tips for setting the modules of the display subsystem.

12.6.8.1 Display Subsystem Use Cases and Tips

This section explains the basic programming model of command mode using the DISPC video port.

The DSI interface is connected to an external MIPI DISPC using the following parameters:

- One data lane: NDL = 1
- Clock lane at 150 MHz (DSI_DDR_CLK)
- LCD size is 640 × 480:
  - 480 PPL
  - 680 LPP
- DISPC input format: YUV
- DISPC output format: RGB888 (24 BPP)
- Word Count: WC = 3 × PPL
- DSS2_ALWON_FLCK = 26 MHz used as a reference clock for DSI PLL
- Virtual channel 1 (VC1) used for command mode to configure the externel DISPC, and virtual channel 0 (VC0) used to send data
- Automatic TE used for synchronization
- Interleaving not used
- It is assumed that all modules used in these programming models are in after-POR state.

Figure 12-181 is an overview of the connections in the display subsystem.

Figure 12-181. Overview

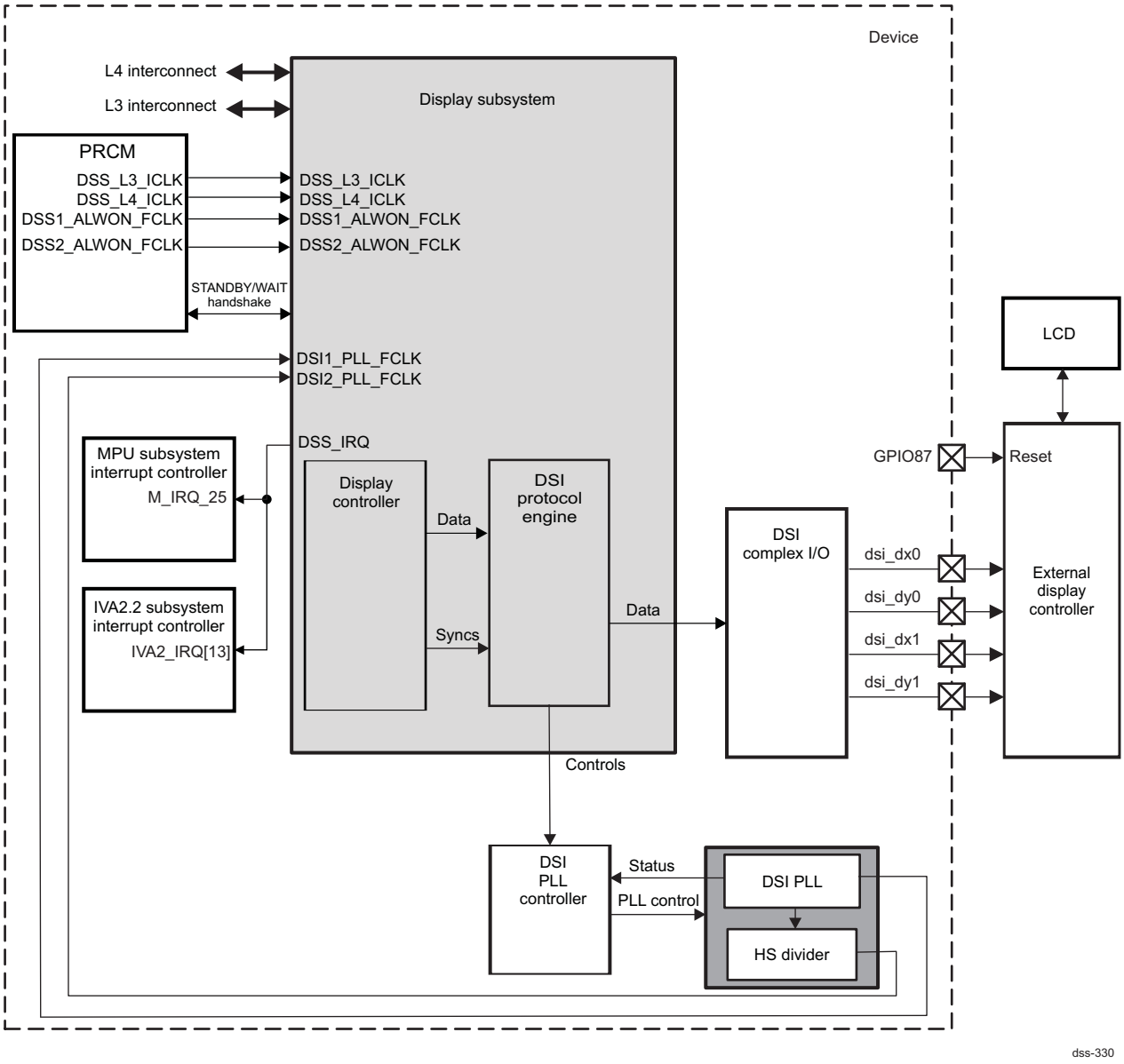


Table 12-110 lists the steps of the main sequence and the sections that describe them.

Table 12-110. Main Sequence

Steps	Register/Bit Field/Programming Model
Configure DSS clocks.	Section 12.6.8.1.1, *Configure DSS Clocks at the PRCM Module*
Configure the DSI protocol engine and DSI PLL.	Section 12.6.8.1.2, *Configure DSI Protocol Engine, DSI PLL, and Complex I/O*
Configure the external MIPI display controller.	Section 12.6.8.1.3, *Initialization of the External MIPI Display Controller*
Configure the DISPC.	Section 12.6.8.1.4, *Configure the DISPC*
Enable command mode using the DISPC video port.	Section 12.6.8.1.5, *Enable Command Mode Using the DISPC Video Port*
Send a frame data to LCD panel using automatic TE.	Section 12.6.8.1.6, *Send Frame Data to LCD Panel Using Automatic TE*

The programming model must follow the step sequence.

12.6.8.1.1 Configure DSS Clocks at the PRCM Module

Table 12-111 lists the steps required to enable the clocks.

Table 12-111. Configure DSS Clocks at the PRCM Module

Steps	Register/Bit Field/Programming Model	Value
Set the divided DPLL value for DSS1.	CM_CLKSEL_DSS[4:0] CLKSEL_DSS1	0x9
Disable autoidle mode.	CM_AUTOIDLE_DSS[0] AUTO_DSS	0x0
Domain sleep is disabled.	CM_SLEEPDEP_DSS[2:0] EN_IVA2, EN_MPU, EN_CORE	0x0
Disable automatic transition.	CM_CLKSTCTRL_DSS[1:0] CLKTRCTRL_DSS	0x0
Enable the DSS1_ALWON_FCLK, DSS2_ALWON_FCLK, and TV_CLK functional clocks.(1)	CM_FCLKEN_DSS[2:0] EN_TV, EN_DSS2, EN_DSS1	0x7
Enable the DSS_L3_ICLK and DSS_L4_ICLK interface clocks.	CM_ICLKEN_DSS[0] EN_DSS	0x1

(1) TV_CLK is required only for a correct reset.

12.6.8.1.2 Configure DSI Protocol Engine, DSI PLL, and Complex I/O

Table 12-112 lists the steps to configure the DSI protocol engine, DSI PLL, and complex I/O and the sections that describe the sequence.

Table 12-112. Configure DSI Protocol Engine, DSI PLL, and Complex I/O

Steps	Register/Bit Field/Programming Model
Reset DSI modules.	Section 12.6.8.1.2.1, *Reset DSI Modules*
Configure DSI DPLL.	Section 12.6.8.1.2.2, *Configure DSI PLL*
Switch to DSI PLL clock source.	Section 12.6.8.1.2.3, *Switch to DSI PLL Clock Source*
Configure DSI protocol engine.	Section 12.6.8.1.2.4, *Configure DSI Protocol Engine*
Configure DSI_PHY.	Section 12.6.8.1.2.5, *Configure DSI_PHY*
Drive stop state.	Section 12.6.8.1.2.6, *Drive Stop State*

12.6.8.1.2.1 Reset DSI Modules

Table 12-113 lists the steps required to reset the DSI modules.

Table 12-113. Reset DSI Modules

Steps	Register/Bit Field/Programming	Value
Clear DSI IRQ status.	DSI_IRQSTATUS[31:0]	0x0
Maintain interface and functional clock during wakeup.	DSI_SYSCONFIG[9:8] CLOCKACTIVITY	0x3
Enable smart-idle for power management.	DSI_SYSCONFIG[4:3] SIDLEMODE	0x2
Set DSI reset.	DSI_SYSCONFIG[1] SOFT_RESET	0x1
Wait until RESET_DONE = 1.	DSI_SYSSTATUS[0] RESET_DONE	

12.6.8.1.2.2 Configure DSI PLL

Table 12-114 lists the steps required to configure the DSI PLL.

Table 12-114. Configure DSI PLL

Steps	Register/Bit Field/Programming	Value
Enable PLL and HSDIVIDER.	DSI_CLK_CTRL[31:30] PLL_PWR_CMD	0x2
Wait until PLL_PWR_STATUS = 0x2.	DSI_CLK_CTRL[29:28] PLL_PWR_STATUS	
Set the REGM4 value (see).	DSI_PLL_CONFIGURATION1[26:23] DSIPROTO_CLK_DIV	5
Set the REGM3 value (see).	DSI_PLL_CONFIGURATION1[22:19] DSS_CLOCK_DIV	15
Set the REGN value (see).	DSI_PLL_CONFIGURATION1[7:1] DSI_PLL_REGN	12
Set the REGM value (see).	DSI_PLL_CONFIGURATION1[18:8] DSI_PLL_REGM	150
Enable PLL STOPMODE.	DSI_PLL_CONFIGURATION1[0] DSI_PLL_STOPMODE	0x1
Set the PLL internal reference frequency range between 1.75 and 2.1 MHz.	DSI_PLL_CONFIGURATION2[4:1] DSI_PLL_FREQSEL	0x7
Enable PLL reference clock control.	DSI_PLL_CONFIGURATION2[13] DSI_PLL_REFEN	0x1
Enable CLKIN4DDR control.	DSI_PLL_CONFIGURATION2[14] DSI_PHY_CLKINEN	0x1
Enable DSS clock divider.	DSI_PLL_CONFIGURATION2[16] DSS_CLOCK_EN	0x1
Enable DSI protocol engine clock divider.	DSI_PLL_CONFIGURATION2[18] DSI_PROTO_CLOCK_EN	0x1
Enable DSI configuration update with DISPC_UPDATE_SYNC.	DSI_PLL_CONTROL[0] DSI_PLL_AUTOMODE	0x0
Start PLL locking sequence.	DSI_PLL_GO[0] DSI_PLL_GO	0x1
Wait until DSI_PLL_GO = 0.	DSI_PLL_GO[0] DSI_PLL_GO	
Check whether PLL is locked.	DSI_PLL_STATUS[1] DSI_PLL_LOCK	0x1
Set the LP mode clock ratio.	DSI_CLK_CTRL[12:0] LP_CLK_DIVISOR	0x8
Set L3_ICLK clock to the DSI complex I/O to not gated.	DSI_CLK_CTRL[14] CIO_CLK_ICG	0x1
Enable the automatic assertion/deassertion of the DSIStopClk signal.	DSI_CLK_CTRL[18] HS_AUTO_STOP_ENABLE	0x1
Specify that the DSI functional clock is higher than 30 MHz with a synchronization rising/rising.	DSI_CLK_CTRL[21] LP_RX_SYNCHRO_ENABLE	0x1
Turn on PLL and HSDIVIDER.	DSI_CLK_CTRL[31:30] PLL_PWR_CMD	0x2

1. Calculate the divider value for the DSI protocol engine clock source:

$$RegM4 = FCLKIN4DDR / FDSI\_PLL\_REFCLK - 1$$

$$FCLKIN4DDR = 4 \times FCLKIN$$

$$RegM4 = 5$$

dss-E126

2. Determine LCD, PCD, and REGM3:

   Calculate the divider value for the DSS clock source: Same as Step 3.

$$RegM3 = ((BPP \times 2)/(DISPC\_LCD \times DISPC\_PCD \times NDL)) - 1$$

$$RegM3 = 15$$

dss-E127

3. Calculate N divider for PLL:

$$FCLKIN4DDR = FCLKIN \times 4$$

$$RegN = (FDSI\_PLL\_REFCLK / F\,INt) - 1$$

$$FDSI\_PLL\_REFCLK = 26\ MHz\ (system\ clock)$$

$$Fint = 2\ MHz\ (reduce\ PLL\ lock\ time)$$

$$RegN = 12$$

dss-E128

4. Calculate M divider for PLL:

$$RegM = ((RegN + 1) \times (FCLKIN4DDR / (2 \times FDSI\_PLL\_REFCLK)))$$

$$FCLKIN4DDR = 4 \times 150\ MHz$$

$$RegM = 150$$

dss-E129

12.6.8.1.2.3 Switch to DSI PLL Clock Source

Table 12-115 lists the sequence to switch the DSI and DISPC module clocks to DSI PLL clock source.

Table 12-115. Switch to DSI PLL Clock Source

Steps	Register/Bit Field/Programming	Value
Switch DISPC clock to DSI1_PLL_FCLK.	DSS_CONTROL[0] DISPC_CLK_SWITCH	0x1
Switch DSI clock to DSI2_PLL_FCLK.	DSS_CONTROL[1] DSI_CLK_SWITCH	0x1

12.6.8.1.2.4 Configure DSI Protocol Engine

12.6.8.1.2.4.1 Set Up DSI Control Registers

Table 12-116 lists the steps required to set up the DSI control registers. Table 12-117 lists the steps to set up the DSI complex I/O registers.

Table 12-116. DSI Control Registers

Steps	Register/Bit Field/Programming	Value
Enable SYNCLOST event.	DSI_IRQENABLE[18] SYNC_LOST_IRQ_EN	0x1
Enable IRQ to indicate that packet has been sent on VC1.	DSI_VC1_IRQENABLE[2] PACKET_SENT_IRQ	0x1
Enable IRQ to indicate that packet has been sent on VC0.	DSI_VC0_IRQENABLE[2] PACKET_SENT_IRQ	0x1
Set the trigger reset mode to *immediate.*	DSI_CTRL[14] TRIGGER_RESET_MODE	0x1
Activate the two line buffers.	DSI_CTRL[13:12] LINE_BUFFER	0x2
Set the size of the video port data bus to 24 bits (RGB 888).	DSI_CTRL[7:6] VP_DATA_BUS_WIDTH	0x2
Define the ratio between VP_CLK and VP_PCLK.	DSI_CTRL[4] VP_CLK_RATIO	0x1
Set the arbitration scheme for granting the VC pending ready requests in the TX FIFO as sequential scheme.	DSI_CTRL[3] TX_FIFO_ARBITRATION	0x1
Enable the ECC check for the received header.	DSI_CTRL[2] ECC_RX_EN	0x1

Table 12-117. DSI Complex I/O Registers

Steps	Register/Bit Field/Programming	Value
Determine the position of the clock lane.	DSI_COMPLEXIO_CFG1[2:0] CLOCK_POSITION	0x2
Determine the position of data 1 lane.	DSI_COMPLEXIO_CFG1[6:4] DATA1_POSITION	0x3
Turn on COMPLEXIO.	DSI_COMPLEXIO_CFG[28:27] PWR_CMD	0x1
Enable the synchronization of the shadow registers with DISPC_UPDATE_SYNC.	DSI_COMPLEXIO_CFG1[30] GOBIT	0x1
Clear all COMPLEXIO IRQ status.	DSI_COMPLEXIO_IRQSTATUS	0xC3F39CE7
Disable all COMPLEXIO IRQs.	DSI_COMPLEXIO_IRQENABLE	0x0
Enable interface.	DSI_CTRL[0] IF_EN	0x1
Disable interface.	DSI_CTRL[0] IF_EN	0x0
Wait until IF_EN = 0.	DSI_CTRL[0] IF_EN	
Enable the LP clock.	DSI_CLK_CTRL[20] LP_CLK_ENABLE	0x1
Check whether reset is complete.	DSI_COMPLEXIO_CFG1[29] RESET_DONE	0x1

Table 12-117. DSI Complex I/O Registers (continued)

Steps	Register/Bit Field/Programming	Value
Check whether power control is on.	DSI_COMPLEXIO_CFG1[26:25] PWR_STATUS	0x1
Check whether reset is complete.	DSI_SYSSTATUS[0] RESETDONE	0x1

12.6.8.1.2.4.2 Configure DSI Timing and Virtual Channels

Table 12-118 lists the steps to configure DSI timing and the virtual channels.

Table 12-118. DSI Timing Registers

Steps	Register/Bit Field/Programming	Value
Determine the number of DSI_FCLK clock cycles for the STOP-STATE counter.	DSI_TIMING1[12:0] STOP_STATE_COUNTER_IO	0x999
Disable the multiplication factor of 4 for the number of DSI_FCLK clock cycles for the STOP-STATE counter.	DSI_TIMING1[13] STOP_STATE_X4_IO	0x0
Disable the multiplication factor of 16 for the number of DSI_FCLK clock cycles for the STOP-STATE counter.	DSI_TIMING1[14] STOP_STATE_X16_IO	0x0
Clear turn-around timer settings.	DSI_TIMING1[30:16]	0x0000
Determine the number of DSI_FCLK clock cycles for the LP RX timer.	DSI_TIMING2[12:0] LP_RX_TO_COUNTER	0x0CD
Disable the multiplication factor of 4 for the number of DSI_FCLK clock cycles for the LP RX timer.	DSI_TIMING2[13] LP_RX_TO_X4	0x0
Enable the multiplication factor of 16 for the number of DSI_FCLK clock cycles for the LP RX timer.	DSI_TIMING2[14] LP_RX_TO_X16	0x1
Determine the number of TxByteClkHS clock cycles for the HS RX timer.	DSI_TIMING2[12:0] HS_TX_TO_COUNTER	0xFD2
Disable the multiplication factor of 8 for the number of TxByteClkHS clock cycles for the HS TX timer.	DSI_TIMING2[13] HS_TX_TO_X8	0x0
Enable the multiplication factor of 16 for the number of TxByteClkHS clock cycles for the HS TX timer.	DSI_TIMING2[14] HS_TX_TO_X16	0x1
DDR_CLK_PRE<<8|DDR_CLK_POST	DSI_CLK_TIMING[31:0]	0x0000 0F0B
Configuration of VC1		
Enable the checksum generation for the transmit payload.	DSI_VC1_CTRL[7] CS_TX_EN	0x1
Enable the ECC generation for the transmit header.	DSI_VC1_CTRL[8] ECC_TX_EN	0x1
Disable DMA request for TX FIFO.	DSI_VC1_CTRL[23:21] DMA_TX_REQ_NB	0x4
Disable DMA request for RX FIFO.	DSI_VC1_CTRL[29:27] DMA_RX_REQ_NB	0x4
Configuration of VC0		
Selects source of data and enable VP_STALL.	DSI_VC0_CTRL[1] SOURCE	0x1
Enable the checksum generation for the transmit payload.	DSI_VC0_CTRL[7] CS_TX_EN	0x1
Enable the ECC generation for the transmit header.	DSI_VC0_CTRL[8] ECC_TX_EN	0x1
Enable high-speed mode to send short and long packets to the peripheral.	DSI_VC0_CTRL[9] MODE_SPEED	0x1
Disable DMA request for TX FIFO.	DSI_VC0_CTRL[23:21] DMA_TX_REQ_NB	0x4
Disable DMA request for RX FIFO.	DSI_VC0_CTRL[29:27] DMA_RX_REQ_NB	0x4
Configuration TX and RX FIFO		

Table 12-118. DSI Timing Registers (continued)

Steps	Register/Bit Field/Programming	Value
Set size of the RX FIFO allocated for VC1 to 32 x 33 bits.	DSI_RX_FIFO_VC_SIZE[15:12] VC1_FIFO_SIZE	0x1
Set size of the TX and TX FIFO allocated for VC1 to 96 x 33 bits.	DSI_TX_FIFO_VC_SIZE[15:12] VC1_FIFO_SIZE	0x3

- Freq HSBYTE_CLK:

$$FHSB = FCLKIN4DDR / 16$$
$$FVPP = FCLKIN4DDR / ((RegM3 + 1) \times DISPC\_LCD * DISPC\_PCD)$$
$$FVP = FCLKIN4DDR / (RegM3 + 1)$$

dss-E130

- Length of the line in video mode in number of byte clock cycles (PPI clk):

$$TL = FHSB / FVPP \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP)$$
$$TL1f = (BPP / (8 \times NDL)) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP)$$

dss-E131

- Blanking periods (HBP + HFP) in DSI are calculated based on the following formula:

$$(DISPC\_HSA + DISPC\_HBP + PPL + DISPC\_HFP) \times Fppi = (HS + HBP + ((WC + 6) / NDL) + HFP) \times Fvp$$
$$HBP + HFP = (TVPP / THSB) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP) - (HS + (WC + 6) / NDL))$$
$$HBPplusHFP = (FHSB / FVPP) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP) - (HS + WC + 6) / NDL$$
$$HBPplusHFPf = ((FHSB / FVPP) \times (DISPC\_HSA + DISPC\_HFP + PPL + DISPC\_HBP)) - ((HS + WC + 6) / NDL)$$
$$HFP = (DISPC\_HFP \times BPP) / (NDL \times 8) - (2 / NDL)$$
$$HBP = HBPplusHFP - HFP$$

dss-E132

12.6.8.1.2.5 Configure DSI_PHY Timing

Table 12-119 summarizes DSI_PHY timing. For more details on timing calculation, see Section 12.4.3.2, *Clock Requirements*.

Table 12-119. Configure DSI_PHY Timing

Steps	Register/Bit Field/Programming	Value
Settings of the DSI protocol timing. For a complete description of timing specifications, see Section 12.4.3.2, *Clock Requirements*.	DSI_PHY_CFG0[31:24] THS_PREPARE	ceil(70 ns/DDR clock period) + 2
	DSI_PHY_CFG0[23:16] THS_PREPARE_THS_ZERO	ceil(175 ns/DDR clock period) + 2
	DSI_PHY_CFG0[7:0] THS_EXIT	ceil(145 ns/DDR clock period)
	DSI_PHY_CFG0[15:8] THS_TRAIL	ceil(60 ns/DDR clock period) + 5
	DSI_PHY_CFG2[7:0] TCLK_PREPARE	ceil(65 ns/DDR clock period)
	DSI_PHY_CFG1[7:0] TCLK_ZERO	ceil(260 ns/DDR clock period)
	DSI_PHY_CFG1[15:0] TCLK_TRAIL	ceil(60 ns/DDR clock period) + 2
	DSI_PHY_CFG1[20:16] TLPX_HALF	ceil(26.5 ns/DDR clock period)

NOTE: Keep Reserved bits at reset value in the DSI_PHY_CFG1 and DSI_PHY_CFG2 registers.

12.6.8.1.2.6 Drive Stop State

Table 12-120 lists the steps to drive the stop state.

Table 12-120. Drive Stop State

Steps	Register/Bit Field/Programming	Value
Force TX stop mode.	DSI_TIMING[15] FORCE_TX_STOP_MODE_IO	0x1
Wait until FORCE_TX_STOP_MODE_IO = 0.	DSI_TIMING[15] FORCE_TX_STOP_MODE_IO	

12.6.8.1.3 Initialization of the External MIPI LCD Controller

Table 12-121 lists the steps to initialize the external MIPI LCD controller.

Table 12-121. Initialization of the External MIPI LCD Controller

Steps	Register/Bit Field/Programming	Value
Reset the MIPI LCD controller using GPIO87.	–	0x1
Wait until initialization of the external MIPI LCD controller is finished after power up.	–	–
Configure the external MIPI LCD controller.	–	–

12.6.8.1.4 Configure the DISPC

12.6.8.1.4.1 Reset DISPC

Table 12-122 lists the steps to reset the DISPC.

Table 12-122. Reset DISPC

Steps	Register/Bit Field/Programming	Value
Reset the DISPC.	DISPC_SYSCONFIG[1] SOFTRESET	0x1
Wait until RESETDONE = 1.	DISPC_SYSCONFIG[0] RESETDONE	
Disable master interface power management.	DISPC_SYSCONFIG [13:12] MIDLEMODE	0x1
Disable slave interface power management.	DISPC_SYSCONFIG [4:3] SIDLEMODE	0x1
Disable all DISPC interrupts.	DISPC_IRQENABLE[31:0]	0x0

12.6.8.1.4.2 Configure DISPC Timing, Window, and Color

Table 12-123 lists the steps to configure the DISPC registers. Table 12-106 lists the steps to configure the color space coefficient registers.

Table 12-124 lists the steps to configure DISPC_CONTROL.

Table 12-123. Configure DISPC Registers

Steps	Register/Bit Field/Programming	Value
Configure LCD output		
Set the logic clock divisor (LCD).	DISPC_DIVISOR[23:16] LCD	0x1
Set the pixel clock divisor (PCD).	DISPC_DIVISOR[7:0] PCD	0x4

Table 12-123. Configure DISPC Registers (continued)

Steps	Register/Bit Field/Programming	Value
Set the number of lines on the LCD panel.	DISPC_SIZE_LCD[26:16] LPP	0x2A7
Set the number of PPL on the LCD panel.	DISPC_SIZE_LCD[10:0] PPL	0x1DF
Set solid background color.	DISPC_DEFAULT_COLOR0	0XFF
Configure VIDEO pipeline VID1.		
Set VID1 FIFO low threshold.	DISPC_VID1_FIFO_THRESHOLD[11:0] VIDFIFOLOWTHRESHOLD	0x0C0
Set VID1 FIFO high threshold.	DISPC_VID1_FIFO_THRESHOLD[27:16] VIDFIFOHIGHTHRESHOLD	0xFC0
Set the X position of the VID1 window.	DISPC_VID1_POSITION[10:0] VIDPOSX	0x0
Set the Y position of the VID1 window.	DISPC_VID1_POSITION[26:16] VIDPOSY	0x0
Set the number of lines of the VID1 window.	DISPC_VID1_SIZE[26:16] VIDSIZEY	0x2A7
Set the number of pixels of the VID1 window.	DISPC_VID1_SIZE[10:0] VIDSIZEX	0x1DF
Define the base address of the VID1 frame buffer.	DISPC_VID1_BA0	0x–

Table 12-124. Configure DISPC_CONTROL

Comments	Register/Bit Field/Programming	Value
Enable pixel clock free-running.	DISPC_CONTROL[27] PCLKFREEENABLE	0x1
Disable RFBI.	DISPC_CONTROL[16] GPOUT1	0x1
	DISPC_CONTROL[15] GPOUT0	0x1
Enable the stall mode.	DISPC_CONTROL[11] STALLMODE	0x1
Select size of DATALINES.	DISPC_CONTROL[9:8] TFTDATALINES	0x3
Select active matrix display operation mode.	DISPC_CONTROL[3] STNTFT	0x1
Disable LCD output interface.	DISPC_CONTROL[0] LCDENABLE	0x0
Update the internal DISPC registers.	DISPC_CONTROL[5] GOLCD	0x1

12.6.8.1.5 Enable Command Mode Using DISPC Video Port

Table 12-125 lists the steps to enable DISPC to send frames continuously. Two bus turn-arounds (BTA) must be generated (see the MIPI DSI specification for details):

- The first BTA gives bus possession to the display module.
- The second BTA obtains the TE trigger.

Table 12-125. Enable Command Mode and Automatic TE

Steps	Register/Bit Field/Programming	Value
Insert DCS write memory continue code.	DSI_CTRL[25] DCS_CMD_CODE	0x0
Enable automatic insertion of DCS command codes when data is sourced by the video port.	DSI_CTRL[24] DCS_CMD_ENABLE	0x1
Enable VC1.	DSI_VC1_CTRL[0] VC_EN	0x1
Enable VC0.	DSI_VC0_CTRL[0] VC_EN	0x1
Enable the interface.	DSI_CTRL[0] IF_EN	0x1
Wait until IF_EN = 1.	DSI_CTRL[0] IF_EN	
Send the sequence to receive theTE trigger from the peripheral. In this use case, code 0x35 + 1 parameter VC = 0, data type = 0x15, DCS write + 1 parameter.	DSI_VC1_SHORT_PACKET_HEADER[31:0] HEADER	0x0000 3515
Wait until PACKET_SENT_IRQ = 1.	DSI_VC1_IRQSTATUS[2] PACKET_SENT_IRQ	
Write 1 to clear PACKET_SENT_IRQ.	DSI_VC1_IRQSTATUS[2] PACKET_SENT_IRQ	0x1

12.6.8.1.6 Send Frame Data to LCD Panel Using Automatic TE

Table 12-126 summarizes the steps to send a frame data to the LCD panel using automatic TE.

Table 12-126. Send Frame Data to LCD Panel Using Automatic TE

Steps	Register/Bit Field/Programming	Value
Enable the transfer between DISPC and DSI. Reset after the transfer is done.	DISPC_CONTROL[0] LCDENABLE	0x1
Specify the number of bytes to send. When DCS insertions is used, word count (WC) must include this one DCS byte.	DSI_VC0_TE[23:0] TE_SIZE	(WC+1)*LPP
Set up long packet header. Send 0x39 DCS long write/write_LUT command packet used to send larger blocks of data to a display module that implements a DCS.	DSI_VC0_LONG_PACKET_HEADER[31:0] HEADER	(WC+1) << 8 + 0x39
Enable TE control.	DSI_VC0_TE[30] TE_EN	0x1
Wait until RX FIFO is empty, RX_FIFO_NOT_EMPTY = 0.	DSI_VC1_CTRL[20] RX_FIFO_NOT_EMPTY	0x0
Wait until TX FIFO is not full. TX_FIFO_FULL = 0.	DSI_VC1_CTRL[16] TX_FIFO_FULL	0x0
Enable first BTA to give bus possession to the display module.	DSI_VC1_CTRL[6] BTA_EN	0x1
Wait until BTA IRQ.	DSI_VC1_IRQSTATUS[5] BTA_IRQ	0x1
Write 1 to clear BTA IRQ.	DSI_VC1_IRQSTATUS[5] BTA_IRQ	0x1
Enable second BTA to get the TE trigger.	DSI_VC1_CTRL[6] BTA_EN	0x1
Wait until BTA IRQ.	DSI_VC1_IRQSTATUS[5] BTA_IRQ	0x1
Write 1 to clear BTA IRQ.	DSI_VC1_IRQSTATUS[5] BTA_IRQ	0x1
Wait until transfer is complete.	DSI_VC0_TE[30] TE_EN	Read 0x0

12.7 Display Subsystem Register Manual

CAUTION

- The DISS, DISPC, RFBI, and VENC registers have no register data width access restriction and can be accessed in 8-bit, 16-bit and 32-bit access..
- The DSI complex I/O and DSI PLL control module registers are limited to 32-bit data access; 16-bit and 8-bit data accesses are not allowed and can corrupt register content.
- The DSI protocol engine DSS.DSI_VCn_LONG_PACKET_HEADER and DSS.DSI_VCn_SHORT_PACKET_HEADER registers are limited to 32-bit data access; 16-bit and 8-bit data accesses are not allowed and can corrupt register content.
- The DSI protocol engine DSS.DSI_VCn_LONG_PACKET_PAYLOAD register is limited to 32-bit and 16-bit data access; 8-bit data accesses are not allowed and can corrupt register content.
- All other DSI protocol engine registers have no register data width access restriction and can be accessed in 8-bit, 16-bit and 32-bit access.

Table 12-127 summarizes the display subsystem instance.

Table 12-127. Instance Summary

Module Name	Base Address	Size
DSI Protocol Engine	0x4804 FC00	512 bytes
DSI_PHY	0x4804 FE00	64 bytes
DSI PLL Controller	0x4804 FF00	32 bytes
Display Subsystem	0x4805 0000	512 bytes
Display Controller	0x4805 0400	1K byte
Display Controller VID1	0x4805 0400	1K byte
Display Controller VID2	0x4805 0400	1K byte
RFBI	0x4805 0800	256 bytes
Video Encoder	0x4805 0C00	256 bytes

12.7.1 Display Subsystem Register Mapping Summary

Table 12-128 through Table 12-133 summarize the display subsystem register mapping.

Table 12-128. Display Subsystem Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DSS_REVISIONNUMBER	R	32	0x000	0x4805 0000
DSS_SYSCONFIG	RW	32	0x010	0x4805 0010
DSS_SYSSTATUS	R	32	0x014	0x4805 0014
DSS_IRQSTATUS	R	32	0x018	0x4805 0018
DSS_CONTROL	RW	32	0x040	0x4805 0040
DSS_SDI_CONTROL	RW	32	0x044	0x4805 0044
DSS_PLL_CONTROL	RW	32	0x048	0x4805 0048
DSS_SDI_STATUS	R	32	0x05C	0x4805 005C

Table 12-129. Display Controller Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DISPC_REVISION	R	32	0x000	0x4805 0400
DISPC_SYSCONFIG	RW	32	0x010	0x4805 0410
DISPC_SYSSTATUS	R	32	0x014	0x4805 0414
DISPC_IRQSTATUS	RW	32	0x018	0x4805 0418
DISPC_IRQENABLE	RW	32	0x01C	0x4805 041C
DISPC_CONTROL	RW	32	0x040	0x4805 0440
DISPC_CONFIG	RW	32	0x044	0x4805 0444
DISPC_DEFAULT_COLOR_m	RW	32	0x04C+(m * 0x04)(1)	0x4805 044C+(m * 0x04)(1)
DISPC_TRANS_COLOR_m	RW	32	0x054+(m * 0x04)(1)	0x4805 0454+(m * 0x04)(1)
DISPC_LINE_STATUS	R	32	0x05C	0x4805 045C
DISPC_LINE_NUMBER	RW	32	0x060	0x4805 0460
DISPC_TIMING_H	RW	32	0x064	0x4805 0464
DISPC_TIMING_V	RW	32	0x068	0x4805 0468
DISPC_POL_FREQ	RW	32	0x06C	0x4805 046C
DISPC_DIVISOR	RW	32	0x070	0x4805 0470
DISPC_GLOBAL_ALPHA	RW	32	0x074	0x4805 0474
DISPC_SIZE_DIG	RW	32	0x078	0x4805 0478
DISPC_SIZE_LCD	RW	32	0x07C	0x4805 047C
DISPC_GFX_BAj	RW	32	0x080+(j * 0x04)(2)	0x4805 0480+(j * 0x04)(2)
DISPC_GFX_POSITION	RW	32	0x088	0x4805 0488
DISPC_GFX_SIZE	RW	32	0x08C	0x4805 048C
DISPC_GFX_ATTRIBUTES	RW	32	0x0A0	0x4805 04A0
DISPC_GFX_FIFO_THRESHOLD	RW	32	0x0A4	0x4805 04A4
DISPC_GFX_FIFO_SIZE_STATUS	R	32	0x0A8	0x4805 04A8
DISPC_GFX_ROW_INC	RW	32	0x0AC	0x4805 04AC
DISPC_GFX_PIXEL_INC	RW	32	0x0B0	0x4805 04B0
DISPC_GFX_WINDOW_SKIP	RW	32	0x0B4	0x4805 04B4
DISPC_GFX_TABLE_BA	RW	32	0x0B8	0x4805 04B8
DISPC_DATA_CYCLEk	RW	32	0x1D4+(k * 0x04)(3)	0x4805 05D4+(k * 0x04)(3)
DISPC_CPR_COEF_R	RW	32	0x220	0x4805 0620
DISPC_CPR_COEF_G	RW	32	0x224	0x4805 0624
DISPC_CPR_COEF_B	RW	32	0x228	0x4805 0628
DISPC_GFX_PRELOAD	RW	32	0x22C	0x4805 062C

(1) m = 0 to 1
(2) j = 0 to 1
(3) k = 0 to 2

Table 12-130. Display Controller VID1 Register Summary

Register Name (n = 1 for VID1)	Type	Register Width (Bits)	Address Offset	Display Controller VID1 Physical Address
DISPC_VIDn_BAj	RW	32	0x0BC+((n–1)* 0x90) + (j * 0x04)(1)	0x4805 04BC + (j *0x04)(1)
DISPC_VIDn_POSITION	RW	32	0x0C4+((n–1)* 0x90)	0x4805 04C4
DISPC_VIDn_SIZE	RW	32	0x0C8+((n–1)* 0x90)	0x4805 04C8

(1) j = 0 to 1

Table 12-130. Display Controller VID1 Register Summary (continued)

Register Name (n = 1 for VID1)	Type	Register Width (Bits)	Address Offset	Display Controller VID1 Physical Address
DISPC_VIDn_ATTRIBUTES	RW	32	0x0CC+((n–1)* 0x90)	0x4805 04CC
DISPC_VIDn_FIFO_THRESHOLD	RW	32	0x0D0+((n–1)* 0x90)	0x4805 04D0
DISPC_VIDn_FIFO_SIZE_STATUS	R	32	0x0D4+((n–1)* 0x90)	0x4805 04D4
DISPC_VIDn_ROW_INC	RW	32	0x0D8+((n–1)* 0x90)	0x4805 04D8
DISPC_VIDn_PIXEL_INC	RW	32	0x0DC+((n–1)* 0x90)	0x4805 04DC
DISPC_VIDn_FIR	RW	32	0x0E0+((n–1)* 0x90)	0x4805 04E0
DISPC_VIDn_PICTURE_SIZE	RW	32	0x0E4+((n–1)* 0x90)	0x4805 04E4
DISPC_VIDn_ACCUI	RW	32	0x0E8 + ((n–1)* 0x90) + (l* 0x04)[2]	0x4805 04E8 + (l* 0x04)[2]
DISPC_VIDn_FIR_COEF_Hi	RW	32	0x0F0+ ((n–1)* 0x90) + (i* 0x08)[3]	0x4805 04F0+ (i* 0x08)[3]
DISPC_VIDn_FIR_COEF_HVi	RW	32	0x0F4+ ((n–1)* 0x90) + (i* 0x08)[3]	0x4805 04F4 + (i*0x08)[3]
DISPC_VIDn_CONV_COEF0	RW	32	0x130+((n–1)* 0x90)	0x4805 0530
DISPC_VIDn_CONV_COEF1	RW	32	0x134+((n–1)* 0x90)	0x4805 0534
DISPC_VIDn_CONV_COEF2	RW	32	0x138+((n–1)* 0x90)	0x4805 0538
DISPC_VIDn_CONV_COEF3	RW	32	0x13C+((n–1)* 0x90)	0x4805 053C
DISPC_VIDn_CONV_COEF4	RW	32	0x140+((n–1)* 0x90)	0x4805 0540
DISPC_VIDn_FIR_COEF_Vi	RW	32	0x1E0+ ((n–1)*0x20) + (i* 0x04)[4]	0x4805 05E0 + (i* 0x04)[4]
DISPC_VIDn_PRELOAD	RW	32	0x230+((n–1)* 0x04)	0x4805 0630

(2) l = 0 to 1
(3) i = 0 to 7
(4) i = 0 to 7

Table 12-131. Display Controller VID2 Register Summary

Register Name (n = 2 for VID2)	Type	Register Width (Bits)	Address Offset	Display Controller VID2 Physical Address
DISPC_VIDn_BAj	RW	32	0x0BC+((n–1)* 0x90) + (j * 0x04)[1]	0x4805 054C+ (j *0x04)[1]
DISPC_VIDn_POSITION	RW	32	0x0C4+((n–1)* 0x90)	0x4805 0554
DISPC_VIDn_SIZE	RW	32	0x0C8+((n–1)* 0x90)	0x4805 0558
DISPC_VIDn_ATTRIBUTES	RW	32	0x0CC+((n–1)* 0x90)	0x4805 055C
DISPC_VIDn_FIFO_THRESHOLD	RW	32	0x0D0+((n–1)* 0x90)	0x4805 0560
DISPC_VIDn_FIFO_SIZE_STATUS	R	32	0x0D4+((n–1)* 0x90)	0x4805 0564
DISPC_VIDn_ROW_INC	RW	32	0x0D8+((n–1)* 0x90)	0x4805 0568
DISPC_VIDn_PIXEL_INC	RW	32	0x0DC+((n–1)* 0x90)	0x4805 056C
DISPC_VIDn_FIR	RW	32	0x0E0+((n–1)* 0x90)	0x4805 0570
DISPC_VIDn_PICTURE_SIZE	RW	32	0x0E4+((n–1)* 0x90)	0x4805 0574
DISPC_VIDn_ACCUI	RW	32	0x0E8 + ((n–1)* 0x90) + (l* 0x04)[2]	0x4805 0578 + (l* 0x04)[2]
DISPC_VIDn_FIR_COEF_Hi	RW	32	0x0F0+ ((n–1)* 0x90) + (i* 0x08)[3]	0x4805 0580 + (i* 0x08)[3]
DISPC_VIDn_FIR_COEF_HVi	RW	32	0x0F4+ ((n–1)* 0x90) + (i* 0x08)[3]	0x4805 0584 + (i*0x08)[3]
DISPC_VIDn_CONV_COEF0	RW	32	0x130+((n–1)* 0x90)	0x4805 05C0

(1) j = 0 to 1
(2) l = 0 to 1
(3) i = 0 to 7

Table 12-131. Display Controller VID2 Register Summary (continued)

Register Name (n = 2 for VID2)	Type	Register Width (Bits)	Address Offset	Display Controller VID2 Physical Address
DISPC_VIDn_CONV_COEF1	RW	32	0x134+((n–1)* 0x90)	0x4805 05C4
DISPC_VIDn_CONV_COEF2	RW	32	0x138+((n–1)* 0x90)	0x4805 05C8
DISPC_VIDn_CONV_COEF3	RW	32	0x13C+((n–1)* 0x90)	0x4805 05CC
DISPC_VIDn_CONV_COEF4	RW	32	0x140+((n–1)* 0x90)	0x4805 05D0
DISPC_VIDn_FIR_COEF_Vi	RW	32	0x1E0+ ((n–1)*0x20) + (i* 0x04)(3)	0x4805 0670 + (i* 0x04)(3)
DISPC_VIDn_PRELOAD	RW	32	0x230+((n–1)* 0x04)	0x4805 0634

Table 12-132. RFBI Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
RFBI_REVISION	R	32	0x00	0x4805 0800
RFBI_SYSCONFIG	RW	32	0x10	0x4805 0810
RFBI_SYSSTATUS	R	32	0x14	0x4805 0814
RFBI_CONTROL	RW	32	0x40	0x4805 0840
RFBI_PIXEL_CNT	RW	32	0x44	0x4805 0844
RFBI_LINE_NUMBER	RW	32	0x48	0x4805 0848
RFBI_CMD	W	32	0x4C	0x4805 084C
RFBI_PARAM	W	32	0x50	0x4805 0850
RFBI_DATA	W	32	0x54	0x4805 0854
RFBI_READ	RW	32	0x58	0x4805 0858
RFBI_STATUS	RW	32	0x5C	0x4805 085C
RFBI_CONFIGi	RW	32	0x60+ (i* 0x18)(1)	0x4805 0860+ (i* 0x18)(1)
RFBI_ONOFF_TIMEi	RW	32	0x64+ (i* 0x18)(1)	0x4805 0864+ (i* 0x18)(1)
RFBI_CYCLE_TIMEi	RW	32	0x68+ (i* 0x18)(1)	0x4805 0868+ (i* 0x18)(1)
RFBI_DATA_CYCLE1_i	RW	32	0x6C+ (i* 0x18)(1)	0x4805 086C+ (i* 0x18)(1)
RFBI_DATA_CYCLE2_i	RW	32	0x70+ (i* 0x18)(1)	0x4805 0870+ (i* 0x18)(1)
RFBI_DATA_CYCLE3_i	RW	32	0x74+ (i* 0x18)(1)	0x4805 0874+ (i* 0x18)(1)
RFBI_VSYNC_WIDTH	RW	32	0x90	0x4805 0890
RFBI_HSYNC_WIDTH	RW	32	0x94	0x4805 0894

(1) i = 0 to 1

Table 12-133. Video Encoder Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
VENC_REV_ID	R	32	0x00	0x4805 0C00
VENC_STATUS	R	32	0x04	0x4805 0C04
VENC_F_CONTROL	RW	32	0x08	0x4805 0C08
VENC_VIDOUT_CTRL	RW	32	0x10	0x4805 0C10
VENC_SYNC_CTRL	RW	32	0x14	0x4805 0C14
VENC_LLEN	RW	32	0x1C	0x4805 0C1C
VENC_FLENS	RW	32	0x20	0x4805 0C20
VENC_HFLTR_CTRL	RW	32	0x24	0x4805 0C24
VENC_CC_CARR_WSS_CARR	RW	32	0x28	0x4805 0C28
VENC_C_PHASE	RW	32	0x2C	0x4805 0C2C
VENC_GAIN_U	RW	32	0x30	0x4805 0C30

Table 12-133. Video Encoder Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
VENC_GAIN_V	RW	32	0x34	0x4805 0C34
VENC_GAIN_Y	RW	32	0x38	0x4805 0C38
VENC_BLACK_LEVEL	RW	32	0x3C	0x4805 0C3C
VENC_BLANK_LEVEL	RW	32	0x40	0x4805 0C40
VENC_X_COLOR	RW	32	0x44	0x4805 0C44
VENC_M_CONTROL	RW	32	0x48	0x4805 0C48
VENC_BSTAMP_WSS_DATA	RW	32	0x4C	0x4805 0C4C
VENC_S_CARR	RW	32	0x50	0x4805 0C50
VENC_LINE21	RW	32	0x54	0x4805 0C54
VENC_LN_SEL	RW	32	0x58	0x4805 0C58
VENC_L21_WC_CTL	RW	32	0x5C	0x4805 0C5C
VENC_HTRIGGER_VTRIGGER	RW	32	0x60	0x4805 0C60
VENC_SAVID_EAVID	RW	32	0x64	0x4805 0C64
VENC_FLEN_FAL	RW	32	0x68	0x4805 0C68
VENC_LAL_PHASE_RESET	RW	32	0x6C	0x4805 0C6C
VENC_HS_INT_START_STOP_X	RW	32	0x70	0x4805 0C70
VENC_HS_EXT_START_STOP_X	RW	32	0x74	0x4805 0C74
VENC_VS_INT_START_X	RW	32	0x78	0x4805 0C78
VENC_VS_INT_STOP_X_VS_INT_START_Y	RW	32	0x7C	0x4805 0C7C
VENC_VS_INT_STOP_Y_VS_EXT_START_X	RW	32	0x80	0x4805 0C80
VENC_VS_EXT_STOP_X_VS_EXT_START_Y	RW	32	0x84	0x4805 0C84
VENC_VS_EXT_STOP_Y	RW	32	0x88	0x4805 0C88
VENC_AVID_START_STOP_X	RW	32	0x90	0x4805 0C90
VENC_AVID_START_STOP_Y	RW	32	0x94	0x4805 0C94
VENC_FID_INT_START_X_FID_INT_START_Y	RW	32	0xA0	0x4805 0CA0
VENC_FID_INT_OFFSET_Y_FID_EXT_START_X	RW	32	0xA4	0x4805 0CA4
VENC_FID_EXT_START_Y_FID_EXT_OFFSET_Y	RW	32	0xA8	0x4805 0CA8
VENC_TVDETGP_INT_START_STOP_X	RW	32	0xB0	0x4805 0CB0
VENC_TVDETGP_INT_START_STOP_Y	RW	32	0xB4	0x4805 0CB4
VENC_GEN_CTRL	RW	32	0xB8	0x4805 0CB8
VENC_OUTPUT_CONTROL	RW	32	0xC4	0x4805 0CC4
VENC_OUTPUT_TEST	RW	32	0xC8	0x4805 0CC8

Table 12-134. DSI Protocol Engine Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DSI_REVISION	R	32	0x000	0x4804 FC00
DSI_SYSCONFIG	RW	32	0x010	0x4804 FC10
DSI_SYSSTATUS	R	32	0x014	0x4804 FC14
DSI_IRQSTATUS	RW	32	0x018	0x4804 FC18
DSI_IRQENABLE	RW	32	0x01C	0x4804 FC1C
DSI_CTRL	RW	32	0x040	0x4804 FC40
DSI_COMPLEXIO_CFG 1	RW	32	0x048	0x4804 FC48
DSI_COMPLEXIO_IRQ STATUS	RW	32	0x04C	0x4804 FC4C

Table 12-134. DSI Protocol Engine Register Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DSI_COMPLEXIO_IRQ ENABLE	RW	32	0x050	0x4804 FC50
DSI_CLK_CTRL	RW	32	0x054	0x4804 FC54
DSI_TIMING1	RW	32	0x058	0x4804 FC58
DSI_TIMING2	RW	32	0x05C	0x4804 FC5C
DSI_VM_TIMING1	RW	32	0x060	0x4804 FC60
DSI_VM_TIMING2	RW	32	0x064	0x4804 FC64
DSI_VM_TIMING3	RW	32	0x068	0x4804 FC68
DSI_CLK_TIMING	RW	32	0x06C	0x4804 FC6C
DSI_TX_FIFO_VC_SIZE	RW	32	0x070	0x4804 FC70
DSI_RX_FIFO_VC_SIZ E	RW	32	0x074	0x4804 FC74
DSI_COMPLEXIO_CFG 2	RW	32	0x078	0x4804 FC78
DSI_RX_FIFO_VC_FUL LNESS	R	32	0x07C	0x4804 FC7C
DSI_VM_TIMING4	RW	32	0x080	0x4804 FC80
DSI_TX_FIFO_VC_EMP TINESS	R	32	0x084	0x4804 FC84
DSI_VM_TIMING5	RW	32	0x088	0x4804 FC88
DSI_VM_TIMING6	RW	32	0x08C	0x4804 FC8C
DSI_VM_TIMING7	RW	32	0x090	0x4804 FC90
DSI_STOPCLK_TIMING	RW	32	0x094	0x4804 FC94
DSI_VCn_CTRL	RW	32	0x100+ (n* 0x20)(1)	0x4804 FD00+ (n* 0x20)(1)
DSI_VCn_TE	RW	32	0x104+ (n* 0x20)(1)	0x4804 FD04+ (n* 0x20)(1)
DSI_VCn_LONG_PACK ET_HEADER	W	32	0x108+ (n* 0x20)(1)	0x4804 FD08+ (n* 0x20)(1)
DSI_VCn_LONG_PACK ET_PAYLOAD	W	32	0x10C+ (n* 0x20)(1)	0x4804 FD0C+ (n* 0x20)(1)
DSI_VCn_SHORT_PAC KET_HEADER	RW	32	0x110+ (n* 0x20)(1)	0x4804 FD10+ (n* 0x20)(1)
DSI_VCn_IRQSTATUS	RW	32	0x118+ (n* 0x20)(1)	0x4804 FD18+ (n* 0x20)(1)
DSI_VCn_IRQENABLE	RW	32	0x11C+ (n* 0x20)(1)	0x4804 FD1C+ (n* 0x20)(1)

(1) n = 0 to 3

Table 12-135. DSI_PHY Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DSI_PHY_CFG0	RW	32	0x0000 0000	0x4804 FE00
DSI_PHY_CFG1	RW	32	0x0000 0004	0x4804 FE04
DSI_PHY_CFG2	RW	32	0x0000 0008	0x4804 FE08
DSI_PHY_CFG3	RW	32	0x0000 000C	0x4804 FE0C
DSI_PHY_CFG4	RW	32	0x0000 0010	0x4804 FE10
DSI_PHY_CFG5	R	32	0x0000 0014	0x4804 FE14

Table 12-136. DSI PLL Controller Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
DSI_PLL_CONTROL	RW	32	0x0000 0000	0x4804 FF00
DSI_PLL_STATUS	R	32	0x0000 0004	0x4804 FF04
DSI_PLL_GO	RW	32	0x0000 0008	0x4804 FF08
DSI_PLL_CONFIGURATION1	RW	32	0x0000 000C	0x4804 FF0C
DSI_PLL_CONFIGURATION2	RW	32	0x0000 0010	0x4804 FF10

12.7.2 Register Descriptions

12.7.2.1 Display Subsystem and SDI Registers

Table 12-137. DSS_REVISIONNUMBER

Address Offset	0x000		
Physical address	0x4805 0000	**Instance**	DISS
Description	This register contains the display subsystem revision number.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REV	Revision number [7:4] Major revision [3:0] Minor revision	R	TI internal data

Table 12-138. Register Call Summary for Register DSS_REVISIONNUMBER

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [0]

Table 12-139. DSS_SYSCONFIG

Address Offset	0x010		
Physical address	0x4805 0010	**Instance**	DISS
Description	This register controls the various parameters of the interconnect interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											Reserved		Reserved		SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Write 0s for future compatibility. Reads return zero.	RW	0x00000000
4:3	Reserved	Reserved. Keep at reset value.	RW	0x0
2	Reserved	Write 0s for future compatibility . Reads return zero.	RW	0

Bits	Field Name	Description	Type	Reset
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. During reads, it always returns 0. 0x0: Normal mode 0x1: The module is reset	RW	0
0	AUTOIDLE	Enable power management capability 0x0: OCP clock is free-running 0x1: Automatic OCP clock gating strategy is applied based on the OCP interface activity	RW	1

Table 12-140. Register Call Summary for Register DSS_SYSCONFIG

Display Subsystem Integration • Software Reset: [0] • Autoidle Mode: [1] • DISPC Interrupt Request: [2]
Display Subsystem Basic Programming Model • Display Subsystem Reset: [3]
Display Subsystem Use Cases and Tips • Autoidle: [4] • Smart-Idle: [5] • Display Subsystem Software Reset: [6] [7]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [8]

Table 12-141. DSS_SYSSTATUS

Address Offset	0x014		
Physical address	0x4805 0014	**Instance**	DISS
Description	This register provides status information about the module.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns 0.	R	0x00000000
0	RESETDONE	Internal reset monitoring Read 0x0: Internal module reset is ongoing. Read 0x1: Reset completed	R	1

Table 12-142. Register Call Summary for Register DSS_SYSSTATUS

Display Subsystem Integration • Software Reset: [0]
Display Subsystem Basic Programming Model • Display Subsystem Reset: [1]
Display Subsystem Use Cases and Tips • Display Subsystem Software Reset: [2] [3]

Table 12-142. Register Call Summary for Register DSS_SYSSTATUS (continued)

Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-143. DSS_IRQSTATUS

Address Offset	0x0000 0018		
Physical Address	0x4805 0018	**Instance**	DSS
Description	The register indicates the source of the interrupt and the status of the interrupt line.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																														DSI_IRQ	DISPC_IRQ

Bits	Field Name	Description	Type	Reset
31:2	RESERVED	Reads returns 0.	R	0x00000000
1	DSI_IRQ	DSI interrupt status (related to DSI_IRQSTATUS) 0x0: DSI interrupt inactive 0x1: DSI interrupt active	R	0x0
0	DISPC_IRQ	DISPC interrupt status (related to DISPC_IRQSTATUS) 0x0: DISPC interrupt inactive 0x1: DISPC interrupt active	R	0x0

Table 12-144. Register Call Summary for Register DSS_IRQSTATUS

Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [0]

Table 12-145. DSS_CONTROL

Address Offset	0x040		
Physical address	0x4805 0040	**Instance**	DISS
Description	This register contains the display subsystem control bits.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								VENC_OUT_SEL	DAC_POWERDN_BGZ	DAC_DEMEN	VENC_CLOCK_4X_ ENABLE	VENC_CLOCK_MODE	DSI_CLK_SWITCH	DISPC_CLK_SWITCH	

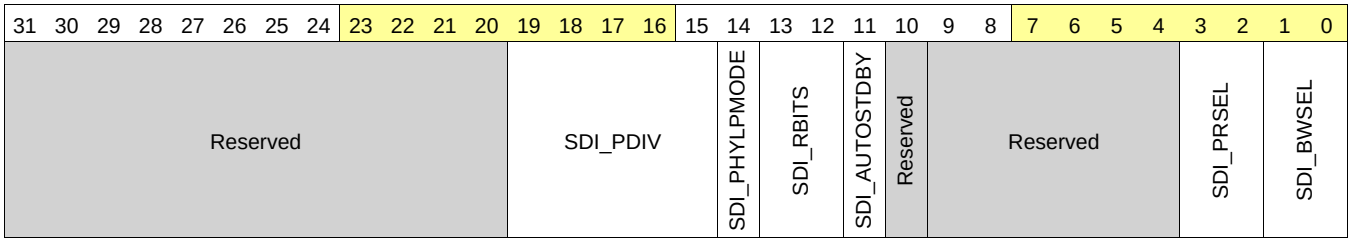
Bits	Field Name	Description	Type	Reset
31:7	Reserved	Reserved for future DAC use	RW	0x0000000
6	VENC_OUT_SEL	Video DAC1 input selection: 0x0: CVBS VENC output selected for composite video mode 0x1: Luminance VENC output selected for s-video mode	RW	0
5	DAC_POWERDN_BGZ	DAC Power-Down Control 0x0: DAC Power-Down Band Gap powered down 0x1: DAC Power-Down Band Gap powered up	RW	0
4	DAC_DEMEN	DAC dynamic element matching enable 0x0: DAC Dynamic Element Matching Disabled 0x1: DAC Dynamic Element Matching Enabled	RW	0
3	VENC_CLOCK_4X_ ENABLE	VENC clock 4x enable 0x0: Disable 0x1: Enable	RW	0
2	VENC_CLOCK_MODE	VENC clock mode See Table 12-19*Possible Digital Clock Division for the Vidoe Encoder.* 0x0: Mode 0 All three balanced clocks, derived from DSS_TV_CLK, are provided to the VENC, if the VENC_CLOCK_4X_ENABLE [3] bit is set to 1 by software. 0x1: Mode 1 The VENC_CLOCK_4X_ENABLE [3] bit is used to control clock gating.	RW	0
1	DSI_CLK_SWITCH	Selects the clock source for the DSI functional clock 0x0: DSS1_ALWON_FCLK clock is selected (from PRCM) 0x1: DSI2_PLL_FCLK clock is selected (from DSI PLL)	RW	0
0	DISPC_CLK_SWITCH	Selects the clock source for the DISPC functional clock 0x0: DSS1_ALWON_FCLK clock is selected (from PRCM) 0x1: DSI1_PLL_FCLK clock is selected (from DSI PLL)	RW	0

Table 12-146. Register Call Summary for Register DSS_CONTROL

Display Subsystem Environment • TV Display Support: [0] • Digital-to-Analog Converter: [1]
Display Subsystem Integration • Clocks: [2] [3] [4] [5] [6] [7]
Display Subsystem Functional Description • Video Encoder Functionalities: [8]
Display Subsystem Basic Programming Model • Display Subsystem Configuration Phase: [9] [10] [11] [12] • Video DAC Settings: [13]
Display Subsystem Use Cases and Tips • Display Subsystem Clock Configuration: [14] [15] [16] • DPLL4 in Low-Power Mode: [17] [18] • Switch to DSI PLL Clock Source: [19] • Switch to DSI PLL Clock Source: [20] [21]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [22]

Table 12-147. DSS_SDI_CONTROL

Address Offset	0x044		
Physical address	0x4805 0044	**Instance**	DISS
Description	This register contains the display subsystem control bits.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:20	Reserved	Reserved. Write 0s for future compatibility. Read returns0.	RW	0x000
19:15	SDI_PDIV	Specifies the ratio of PLL output to pixel clock frequency	RW	0x00
14	SDI_PHYLPMODE	FlatLink3G output buffer low power option Disables the internal transmitter termination to reduce power. Requires reduced data and signal integrity verification 0x0: Standard mode 0x1: Low-power mode	RW	0
13:12	SDI_RBITS	FlatLink3G reserved bits F1 and F0	RW	0x0
11	SDI_AUTOSTDBY	FlatLink3G auto-standby 0x0: High-speed serial buffers only enabled when PLL is locked and SDI enable is active 0x1: High-speed serial buffers enabled while SDI enable is active	RW	0
10	Reserved	Must be programmed to the default value for SN65LVDS302 compatibility	RW	0
9:4	Reserved	Must be programmed to the default value	RW	0x0
3:2	SDI_PRSEL	Selection of the number of active data pairs 0x0: 1 pair: DATA0 0x1: 2 pairs: DATA0 and DATA1 0x2: 3 pairs: DATA0, DATA1, and DATA2	RW	0x0
1:0	SDI_BWSEL	Selects the color depth: must be programmed to 0x2 for FlatLink3G 0x0: Reserved 0x1: Reserved 0x2: Color depth is 24 bits.	RW	0x0

Table 12-148. Register Call Summary for Register DSS_SDI_CONTROL

Display Subsystem Basic Programming Model • SDI Configuration: [0] • Number of Data Pairs: [1] [2]
Display Subsystem Use Cases and Tips • SDI PLL Settings for 1-Channel Mode:: [3] [4] • SDI PLL Settings for 2-Channel Mode:: [5] [6] • SDI Configuration: [7] • SDI PLL settings for 3-Channel Mode:: [8] [9]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [10]

Table 12-149. DSS_PLL_CONTROL

Address Offset	0x048		
Physical address	0x4805 0048	**Instance**	DISS
Description	This register contains the display subsystem control bits.		
Type	RW		

31 30 29	28	27 26	25 24	23 22	21	20	19	18	17	16	15 14 13 12 11	10 9 8	7 6 5 4 3 2 1	0
Reserved	SDI_PLL_GOBIT	SDI_PLL_LOCKSEL	SDI_PLL_FREQSEL		SDI_PLL_PLLLPMODE	SDI_PLL_LOWCURRSTBY	SDI_PLL_HIGHFREQ	SDI_PLL_SYSRESET	SDI_PLL_STOPMODE		SDI_PLL_REGN	SDI_PLL_REGM		SDI_PLL_IDLE

Bits	Field Name	Description	Type	Reset
31:29	Reserved	Reserved	RW	0x0
28	SDI_PLL_GOBIT	Requests PLL locking sequence. See the programming guide section for the use of this bit in conjunction with DSS_STATUS[6] SDI_PLL_BUSYFLAG 0: Inactive 1: Request PLL locking sequence	RW	0
27:26	SDI_PLL_LOCKSEL	Selects the lock criteria for PLL 0x0: Phase Lock (recommended setting for FlatLink3G) 0x1: Fine Phase Lock 0x2: Frequency Lock	RW	0x0
25:22	SDI_PLL_FREQSEL	PLL internal reference frequency (Fint) range selection 0x3: 0.75MHz to 1.0MHz 0x4: 1.0MHz to 1.25MHz 0x5: 1.25MHz to 1.5MHz 0x6: 1.5MHz to 1.75MHz 0x7: 1.75MHz to 2.1MHz 0xB: 7.5MHz to 10MHz 0xC: 10MHz to 12.5MHz 0xD: 12.5MHz to 15MHz 0xE: 15MHz to 17.5MHz 0xF: 17.5MHz to 21MHz Others: Reserved	RW	0x0
21	SDI_PLL_PLLLPMODE	Select the power/performance of the PLL 0x0: Full performance, minimized jitter 0x1: Reduced power, increased jitter	RW	0
20	SDI_PLL_LOWCURRSTBY	PLL Low Current Standby 0x0: Low Current Standby is not selected. 0x1: Low Current Standby is selected.	RW	0
19	SDI_PLL_HIGHFREQ	Enables a division of pixel clock by 2 before input of PLL. Required for pixel clock frequency above 32 MHz 0x0: Pixel clock is not divided. 0x1: Pixel clock is divided by 2.	RW	0
18	SDI_PLL_SYSRESET	SDI PLL reset bit. Active low default 0x0 : PLL under reset 0x1 : PLL operational	RW	0
17	SDI_PLL_STOPMODE	0xSDI PLL STOPMODE 0x0 : STOPMODE is not selected 0x1 : STOPMODE is selected	RW	0
16:11	SDI_PLL_REGN	SDI PLL REGN register	RW	0x00
10:1	SDI_PLL_REGM	SDI PLL REGM register	RW	0x000

Bits	Field Name	Description	Type	Reset
0	SDI_PLL_IDLE	SDI PLL IDLE 0x0: IDLE is not selected 0x1: IDLE is selected	RW	0

Table 12-150. Register Call Summary for Register DSS_PLL_CONTROL

Display Subsystem Integration • SDI Idle Mode: [0] [1] [2]
Display Subsystem Basic Programming Model • SDI PLL Configuration: [3] [4] [5] • SDI Reset State: [6] • SDI Power_On Sequence: [7] • SDI Power-Down Sequence: [8] • Complete Sequence: [9] [10]
Display Subsystem Use Cases and Tips • SDI PLL Configuration: [11] [12] [13] • SDI PLL Settings for 1-Channel Mode:: [14] [15] [16] [17] • SDI PLL Settings for 2-Channel Mode:: [18] [19] [20] [21] • SDI PLL settings for 3-Channel Mode:: [22] [23] [24] [25]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [26] • Display Subsystem and SDI Registers: [27]

Table 12-151. DSS_SDI_STATUS

Address Offset	0x05C		
Physical address	0x4805 005C	**Instance**	DISS
Description	This register contains the display subsystem register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							DSI_PLL_CLK2_STATUS	DSS_DSI_CLK1_STATUS	SDI_PLL_BUSYFLAG	SDI_PLL_LOCK	SDI_PLL_RECAL	SDI_ERROR	SDI_RESET_DONE	DSI_PLL_CLK1_STATUS	DSS_DISPC_CLK1_STATUS

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Reserved	R	0x0000000
8	DSI_PLL_CLK2_ STATUS	DSI2_PLL_FCLK clock selection status (DSI mux) Indicates if the DSI protocol engine is running from the DSI2_PLL_FCLK clock Read 0: DSI2_PLL_FCLK is not selected (unused by DSI). Read 1: DSI2_PLL_FCLK is selected (used by DSI).	R	0
7	DSS_DSI_CLK1_ STATUS	DSS1_ALWON_FCLK clock selection status (DSI mux) Indicates if the DSI protocol engine is running from the DSS1_ALWON_FCLK clock Read 0: DSS1_ALWON_FCLK is not selected (unused by DSI). Read 1: DSS1_ALWON_FCLK is selected (used by DSI).	R	1

Bits	Field Name	Description	Type	Reset
6	SDI_PLL_BUSYFLAG	PLL locking sequence status. See the programming guide section for the use of this bit in conjunction with DSS_PLL_CONTROL[28] SDI_PLL_GOBIT Read 0: PLL lock request has completed. Read 1: PLL lock request is in progress.	R	0
5	SDI_PLL_LOCK	SDI PLL lock status See the programming guide section for the use of this bit Read 0: PLL is not locked Read 1: PLL is locked	R	0
4	SDI_PLL_RECAL	SDI DPLL re-calibration status If this bit is active, the PLL must be recalibrated Read 0x0: Recalibration is not required. Read 0x1: Recalibration is required.	R	0
3	SDI_ERROR	SDI error status bit See programming guide section for error recovery procedure Read 0x0: No error Read 0x1: Error condition required	R	0
2	SDI_RESET_DONE	SDI reset done status This status is delayed until the PLL output has started running Read 0x0: SDI reset is in progress Read 0x1: SDI reset has completed	R	0
1	DSI_PLL_CLK1_ STATUS	DSI1_PLL_FCLK clock selection status (DISPC mux) Indicates if the display controller is running from the DSI1_PLL_FCLK clock Read 0: DSI1_PLL_FCLK is not selected (unused by DISPC). Read 1: DSI1_PLL_FCLK is selected (used by DISPC).	R	0
0	DSS_DISPC_CLK1_ STATUS	DSS1_ALWON_FCLK clock selection status (DISPC mux) Indicates if the display controller is running from the DSS1_ALWON_FCLK clock Read 0: DSS1_ALWON_FCLK is not selected (unused by DISPC). Read 1: DSS1_ALWON_FCLK is selected (used by DISPC).	R	1

Table 12-152. Register Call Summary for Register DSS_SDI_STATUS

Display Subsystem Basic Programming Model • SDI Error Management: [0] [1]
Display Subsystem Use Cases and Tips • SDI PLL Architecture: [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

12.7.2.2 Display Controller Registers

Table 12-153. DISPC_REVISION

Address Offset	0x000		
Physical address	0x4805 0400	**Instance**	DISC
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision	R	TI internal data

Table 12-154. Register Call Summary for Register DISPC_REVISION

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [0]

Table 12-155. DISPC_SYSCONFIG

Address Offset	0x010		
Physical address	0x4805 0410	**Instance**	DISC
Description	This register allows the control of various parameters of the interconnect interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		MIDLEMODE		Reserved		CLOCKACTIVITY		Reserved			SIDLEMODE		ENWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
13:12	MIDLEMODE	Master interface power management, standby/waitcontrol	RW	0x0
		0x0: Force standby. MStandby is asserted only when the module is disabled.		
		0x1: No standby: MStandby is never asserted.		
		0x2: Smart Standby. MStandby is asserted based on the internal activity of the module.		
		0x3: Reserved		
11:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
9:8	CLOCKACTIVITY	Clock activity during wakeup mode period	RW	0x0
		0x0: interface and functional clocks can be switched off.		
		0x1: Functional clocks can be switched off and interface clocks are maintained during wakeup period.		
		0x2: Interface clocks can be switched off and functional clocks are maintained during wakeup period.		
		0x3: Interface and functional clocks are maintained during wakeup period.		
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4:3	SIDLEMODE	Slave interface power management, idle req/ack control	RW	0x0
		0x0: Force idle. An idle request is acknowledged unconditionally.		
		0x1: No idle. An idle request is never acknowledged.		
		0x2: Smart idle. Idle request is acknowledged based on the internal activity of the module.		
		0x3: Reserved		
2	ENWAKEUP	Wakeup feature control	RW	0

Bits	Field Name	Description	Type	Reset
		0x0: Wakeup is disabled. 0x1: Wakeup is enabled.		
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. During reads, it always returns 0. 0x0: Normal mode 0x1: The module is reset.	RW	0
0	AUTOIDLE	Internal interface clock gating strategy 0x0: Interface clock is free-running. 0x1: Automatic L3 and L4 interface clock gating strategy is applied based on interface activity.	RW	1

Table 12-156. Register Call Summary for Register DISPC_SYSCONFIG

Display Subsystem Integration • Software Reset: [0] • Clock Activity Mode: [1] [2] [3] [4] [5] • Autoidle Mode: [6] • Idle Mode: [7] [8] [9] • Wake-Up Mode: [10] • Standby Mode: [11]
Display Subsystem Basic Programming Model • Display Controller Configuration: [12]
Display Subsystem Use Cases and Tips • Autoidle: [13] • Smart-Idle: [14] • Reset DISPC: [15] [16] [17] [18] • Reset DISPC: [19] [20] [21] [22]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [23]

Table 12-157. DISPC_SYSSTATUS

Address Offset	0x014		
Physical address	0x4805 0414	**Instance**	DISC
Description	This register provides status information about the module, excluding interrupt status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								Reserved							RESETDONE

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x000000
7:1	Reserved	Reserved. Read returns 0.	R	0x00
0	RESETDONE	Internal reset monitoring Read 0x0: Internal module reset is ongoing. Read 0x1: Reset complete	R	0

Table 12-158. Register Call Summary for Register DISPC_SYSSTATUS

Display Subsystem Basic Programming Model • Display Controller Configuration: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1]

Table 12-159. DISPC_IRQSTATUS

Address Offset	0x018		
Physical address	0x4805 0418	**Instance**	DISC
Description	This register regroups all the status of module internal events that generate an interrupt. A write of 1 to a given bit resets the bit.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WAKEUP	SYNCLOSTDIGITAL	SYNCLOST	VID2ENDWINDOW	VID2FIFOUNDERFLOW	VID1ENDWINDOW	VID1FIFOUNDERFLOW	OCPERROR	PALETTEGAMMALOADING	GFXENDWINDOW	GFXFIFOUNDERFLOW	PROGRAMMEDLINENUMBER	ACBIASCOUNTSTATUS	EVSYNC_ODD	EVSYNC_EVEN	VSYNC	FRAMEDONE

Bits	Field Name	Description		Type	Reset
31:17	Reserved	Write 0s for future compatibility. Read returns 0		RW	0x0000
16	WAKEUP	Wakeup		RW	0
		Read 0x0:	Wakeup is false.		
		Write 0x0:	Wakeup status bit unchanged		
		Read 0x1:	Wakeup is true (pending).		
		Write 0x1:	Wakeup status bit reset		
15	SYNCLOSTDIGITAL	SyncLostDigital		RW	0
		Read 0x0:	SyncLostDigital is false.		
		Write 0x0:	SyncLostDigital status bit unchanged		
		Read 0x1:	SyncLostDigital is true (pending).		
		Write 0x1:	SyncLostDigital status bit reset		
14	SYNCLOST	SyncLost		RW	0
		Read 0x0:	SyncLost is false.		
		Write 0x0:	SyncLost status bit unchanged		
		Read 0x1:	SyncLost is true (pending).		
		Write 0x1:	SyncLost status bit reset		
13	VID2ENDWINDOW	Vid2EndWindow		RW	0
		Read 0x0:	Vid2EndWindow is false.		
		Write 0x0:	Vid2EndWindow status bit unchanged		
		Read 0x1:	Vid2EndWindow is true (pending).		
		Write 0x1:	Vid2EndWindow status bit reset		
12	VID2FIFOUNDERFLOW	Vid2FIFOUnderflow		RW	0
		Read 0x0:	Vid2FIFOUnderflow is false.		
		Write 0x0:	Vid2FIFOUnderflow status bit unchanged		

Bits	Field Name	Description	Type	Reset
		Read 0x1: Vid2FIFOUnderflow is true (pending).		
		Write 0x1: Vid2FIFOUnderflow status bit reset		
11	VID1ENDWINDOW	Vid1EndWindow	RW	0
		Read 0x0: Vid1EndWindow is false.		
		Write 0x0: Vid1EndWindow status bit unchanged		
		Read 0x1: Vid1EndWindow is true (pending).		
		Write 0x1: Vid1EndWindow status bit reset		
10	VID1FIFOUNDERFLOW	Vid1FIFOUnderflow	RW	0
		Read 0x0: Vid1FIFOUnderflow is false.		
		Write 0x0: Vid1FIFOUnderflow status bit unchanged		
		Read 0x1: Vid1FIFOUnderflow is true (pending).		
		Write 0x1: Vid1FIFOUnderflow status bit reset		
9	OCPERROR	OCPError	RW	0
		Read 0x0: OCPError is false.		
		Write 0x0: OCPError status bit unchanged		
		Read 0x1: OCPError is true (pending).		
		Write 0x1: OCPError status bit reset		
8	PALETTEGAMMA LOADING	PaletteGammaLoading	RW	0
		Read 0x0: PaletteGammaLoading is false.		
		Write 0x0: PaletteGammaLoading status bit unchanged		
		Read 0x1: PaletteGammaLoading is true (pending).		
		Write 0x1: PaletteGammaLoading status bit reset		
7	GFXENDWINDOW	GfxEndWindow	RW	0
		Read 0x0: GfxEndWindow is false.		
		Write 0x0: GfxEndWindow status bit unchanged		
		Read 0x1: GfxEndWindow is true (pending).		
		Write 0x1: GfxEndWindow status bit reset		
6	GFXFIFOUNDERFLOW	GfxFIFOUnderflow	RW	0
		Read 0x0: GfxFIFOUnderflow is false.		
		Write 0x0: GfxFIFOUnderflow status bit unchanged		
		Read 0x1: GfxFIFOUnderflow is true (pending).		
		Write 0x1: GfxFIFOUnderflow status bit reset		
5	PROGRAMMEDLINE NUMBER	ProgrammedLineNumber	RW	0
		Read 0x0: ProgrammedLineNumber is false.		
		Write 0x0: ProgrammedLineNumber status bit unchanged		
		Read 0x1: ProgrammedLineNumber is true (pending).		
		Write 0x1: ProgrammedLineNumber status bit reset		
4	ACBIASCOUNTSTATUS	ACBiasCountStatus	RW	0
		Read 0x0: ACBiasCountStatus is false.		
		Write 0x0: ACBiasCountStatus status bit unchanged		
		Read 0x1: ACBiasCountStatus is true (pending).		
		Write 0x1: ACBiasCountStatus status bit reset		
3	EVSYNC_ODD	EVSYNC_ODD	RW	0
		Read 0x0: EVSYNC_ODD is false.		
		Write 0x0: EVSYNC_ODD status bit unchanged		
		Read 0x1: EVSYNC_ODD is true (pending).		
		Write 0x1: EVSYNC_ODD status bit reset		
2	EVSYNC_EVEN	EVSYNC_EVEN	RW	0

Bits	Field Name	Description		Type	Reset
		Read 0x0:	EVSYNC_EVEN is false.		
		Write 0x0:	EVSYNC_EVEN status bit unchanged		
		Read 0x1:	EVSYNC_EVEN is true (pending).		
		Write 0x1:	EVSYNC_EVEN status bit reset		
1	VSYNC	VSYNC		RW	0
		Read 0x0:	VSYNC is false.		
		Write 0x0:	VSYNC status bit unchanged		
		Read 0x1:	VSYNC is true (pending).		
		Write 0x1:	VSYNC status bit reset		
0	FRAMEDONE	FrameDone		RW	0
		Read 0x0:	FrameDone is false.		
		Write 0x0:	FrameDone status bit unchanged		
		Read 0x1:	FrameDone is true (pending).		
		Write 0x1:	FrameDone status bit reset		

Table 12-160. Register Call Summary for Register DISPC_IRQSTATUS

Display Subsystem Integration • DISPC Interrupt Request: [0] [1]
Display Subsystem Basic Programming Model • Display Subsystem Reset: [2] [3] • Display Controller Configuration: [4] • TV Set-Specific Control Registers: [5] • Video Encoder Programming Sequence: [6]
Display Subsystem Use Cases and Tips • Interrupts Enable: [7] [8]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [9] • Display Subsystem and SDI Registers: [10]

Table 12-161. DISPC_IRQENABLE

Address Offset	0x01C		
Physical address	0x4805 041C	**Instance**	DISC
Description	This register allows the masking/unmasking of module internal interrupt sources, on an event-by-event basis.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																WAKEUP	SYNCLOSTDIGITAL	SYNCLOST	VID2ENDWINDOW	VID2FIFOUNDERFLOW	ENDVID1WINDOW	VID1FIFOUNDERFLOW	OCPERROR	PALETTEGAMMAMASK	GFXENDWINDOW	GFXFIFOUNDERFLOW	PROGRAMMEDLINENUMBER	ACBIASCOUNTSTATUS	EVSYNC_ODD	EVSYNC_EVEN	VSYNC	FRAMEMASK

Bits	Field Name	Description	Type	Reset
31:17	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
16	WAKEUP	Wakeup mask 0x0: Wakeup is masked. 0x1: Wakeup generates an interrupt when it occurs.	RW	0
15	SYNCLOSTDIGITAL	SyncLostDigital 0x0: SyncLostDigital is masked. 0x1: SyncLostDigital generates an interrupt when it occurs.	RW	0
14	SYNCLOST	SyncLost 0x0: SyncLost is masked. 0x1: SyncLost generates an interrupt when it occurs.	RW	0
13	VID2ENDWINDOW	Vid2EndWindow 0x0: Vid2EndWindow is masked. 0x1: Vid2EndWindow generates an interrupt when it occurs.	RW	0
12	VID2FIFOUNDERFLOW	Vid2FIFOUnderflow 0x0: Vid2FIFOUnderflow is masked. 0x1: Vid2FIFOUnderflow generates an interrupt when it occurs.	RW	0
11	ENDVID1WINDOW	EndVid1Window 0x0: EndVid1Window is masked. 0x1: EndVid1Window generates an interrupt when it occurs.	RW	0
10	VID1FIFOUNDERFLOW	Vid1FIFOUnderflow 0x0: Vid1FIFOUnderflow is masked. 0x1: Vid1FIFOUnderflow generates an interrupt when it occurs.	RW	0
9	OCPERROR	OCPError 0x0: OCPError is masked. 0x1: OCPError generates an interrupt when it occurs.	RW	0
8	PALETTEGAMMAMASK	PaletteGammaMask 0x0: PaletteGammaMask is masked. 0x1: PaletteGammaMask generates an interrupt when it occurs.	RW	0
7	GFXENDWINDOW	GfxEndWindow 0x0: GfxEndWindow is masked. 0x1: GfxEndWindow generates an interrupt when it occurs.	RW	0
6	GFXFIFOUNDERFLOW	GfxFIFOUnderflow 0x0: GfxFIFOUnderflow is masked. 0x1: GfxFIFOUnderflow generates an interrupt when it occurs.	RW	0
5	PROGRAMMEDLINE NUMBER	ProgrammedLineNumber 0x0: ProgrammedLineNumber is masked. 0x1: ProgrammedLineNumber generates an interrupt when it occurs.	RW	0
4	ACBIASCOUNTSTATUS	ACBiasCountStatus 0x0: ACBiasCountStatus is masked. 0x1: ACBiasCountStatus generates an interrupt when it occurs.	RW	0
3	EVSYNC_ODD	EVSYNC_ODD 0x0: EVSYNC_ODD is masked. 0x1: EVSYNC_ODD generates an interrupt when it occurs.	RW	0
2	EVSYNC_EVEN	EVSYNC_EVEN	RW	0

Bits	Field Name	Description	Type	Reset
		0x0: EVSYNC_EVEN is masked.		
		0x1: EVSYNC_EVEN generates an interrupt when it occurs.		
1	VSYNC	VSYNC	RW	0
		0x0: VSYNC is masked.		
		0x1: VSYNC generates an interrupt when it occurs.		
0	FRAMEMASK	FrameMask	RW	0
		0x0: FrameMask is masked.		
		0x1: FrameMask generates an interrupt when it occurs.		

Table 12-162. Register Call Summary for Register DISPC_IRQENABLE

Display Subsystem Integration • DISPC Interrupt Request: [0] [1]
Display Subsystem Basic Programming Model • Display Controller Configuration: [2] • TV Set-Specific Control Registers: [3] • Video Encoder Programming Sequence: [4] [5] [6]
Display Subsystem Use Cases and Tips • Interrupts Enable: [7] [8] [9] [10] [11] • Reset DISPC: [12] • Reset DISPC: [13]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [14]

Table 12-163. DISPC_CONTROL

Address Offset	0x040		
Physical address	0x4805 0440	**Instance**	DISC
Description	The control register configures the display controller module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPATIALTEMPORALDITHERINGFRAMES		LCDENABLEPOL	LCDENABLESIGNAL	PCKFREEENABLE	TDMUNUSEDBITS		TDMCYCLEFORMAT	TDMPARALLELMODE			TDMENABLE	HT			GPOUT1	GPOUT0	GPIN1	GPIN0	OVERLAYOPTIMIZATION	STALLMODE	Reserved	TFTDATALINES		STDITHERENABLE	GODIGITAL	GOLCD	M8B	STNTFT	MONOCOLOR	DIGITALENABLE	LCDENABLE

Bits	Field Name	Description	Type	Reset
31:30	SPATIALTEMPORAL DITHERINGFRAMES	Spatial/Temporal dithering number of frames wr: VFP	RW	0x0
		0x0: Spatial only		
		0x1: Spatial and temporal over two frames		
		0x2: Spatial and temporal over four frames		

Bits	Field Name	Description	Type	Reset
		0x3: Reserved		
29	LCDENABLEPOL	LCD Enable Signal Polarity	RW	0
		0x0: Active low		
		0x1: Active high		
28	LCDENABLESIGNAL	LCD Enable Signal: LCD interface active/inactive	RW	0
		0x0: Signal disabled		
		0x1: Signal enabled		
27	PCKFREEENABLE	Pixel clock free-running enabled/disabled	RW	0
		0x0: Clock disabled		
		0x1: Clock enabled		
26:25	TDMUNUSEDBITS	State of unused bits (TDM mode only) WR: VFP	RW	0x0
		0x0: Low level (0)		
		0x1: High level (1)		
		0x2: Unchanged from previous state		
		0x3: Reserved		
24:23	TDMCYCLEFORMAT	Cycle format (TDM mode only) WR: VFP	RW	0x0
		0x0: 1 cycle for 1 pixel		
		0x1: 2 cycles for 1 pixel		
		0x2: 3 cycles for 1 pixel		
		0x3: 3 cycles for 2 pixels		
22:21	TDMPARALLELMODE	Output Interface width (TDM mode only) WR: VFP	RW	0x0
		0x0: 8-bit parallel output interface selected		
		0x1: 9-bit parallel output interface selected		
		0x2: 12-bit parallel output interface selected		
		0x3: 16-bit parallel output interface selected		
20	TDMENABLE	Enable the multiple cycle format (TDM mode used only for Active Matrix mode with the RFBI enable bit off). WR: VFP	RW	0
		0x0: TDM disabled		
		0x1: TDM enabled		
19:17	HT	Hold Time for digital output WR: EVSYNC Encoded value (from 0 to 7) holds time for digital output. The data will be held for (HT + 1) external digital clock periods.	RW	0x0
16	GPOUT1	General Purpose Output Signal	RW	0
		0x0: The GPout1 is reset.		
		0x1: The GPout1 is set.		
15	GPOUT0	General Purpose Output Signal	RW	0
		0x0: The GPout0 is reset.		
		0x1: The GPout0 is set.		
14	GPIN1	General Purpose Input Signal WR: VFP	R	0
		Read 0x0: The GPin1 has been reset.		
		Read 0x1: The GPin1 has been set.		
13	GPIN0	General Purpose Input Signal WR: VFP	R	0
		Read 0x0: The GPin0 has been reset.		

Bits	Field Name	Description	Type	Reset
		Read 0x1: The GPin0 has been set.		
12	OVERLAYOPTIMIZATION	Overlay Optimization (available when graphics format is NOT is 1-, 2, and 4-BPP) WR: VFP or EVSYNC 0x0: Graphics data below video1 window fetched from memory or no overlap between graphics and video1 windows. 0x1: Graphics data below video1 window not fetched from memory.	RW	0
11	STALLMODE	Stall mode for the LCD output wr: VFP 0x0: Normal mode selected 0x1: Stall mode selected. The Display Controller sends the data without considering the VSYNC/HSYNC. The LCD output is disabled at the end of the transfer of the frame. The S/W has to re-enable the LCD output to generate a new frame. The stall mode is used in RFBI and DSI command modes.	RW	0
10	Reserved	Reserved for non-GP devices	RW	0
9:8	TFTDATALINES	Number of lines of the LCD interface WR: VFP 0x0: 12-bit output aligned on the LSB of the pixel data interface 0x1: 16-bit output aligned on the LSB of the pixel data interface 0x2: 18-bit output aligned on the LSB of the pixel data interface 0x3: 24-bit output aligned on the LSB of the pixel data interface	RW	0x0
7	STDITHERENABLE	Spatial temporal dithering enable WR: VFP 0x0: Spatial/temporal dithering logic disabled 0x1: Spatial/temporal dithering logic enabled	RW	0
6	GODIGITAL	Digital GO Command 0x0: The hardware has finished updating the internal shadow registers of the pipeline(s) associated with the digital output using the user values. The hardware resets the bit when the update is completed. 0x1: Users have finished programming the shadow registers of the pipeline(s) associated with the digital output and the hardware can update the internal registers at the external VSYNC.	RW	0
5	GOLCD	LCD GO Command 0x0: The hardware has finished updating the internal shadow registers of the pipeline(s) connected to the LCD output using the user values. The hardware resets the bit when the update is completed. 0x1: Users have finished programming the shadow registers of the pipeline(s) associated with the LCD output and the hardware can update the internal registers at the VFP start period.	RW	0
4	M8B	Mono 8-bit mode WR: VFP 0x0: Pixel data [3:0] is used to output four pixel values to the panel at each pixel clock transition (only in Passive Mono 8-bit mode). 0x1: Pixel data [7:0] is used to output eight pixel values to the panel each pixel clock transition (only in Passive Mono 8-bit mode).	RW	0

Bits	Field Name	Description	Type	Reset
3	STNTFT	LCD display type WR: VFP 0x0: Passive or Passive Matrix display operation enabled. Passive Matrix dither logic enabled. 0x1: Active Matrix display operation enabled. Passive Matrix Dither logic and output FIFO bypassed.	RW	0
2	MONOCOLOR	Monochrome/Color WR: VFP 0x0: Color operation enabled (Passive Matrix mode only) 0x1: Monochrome operation enabled (Passive Matrix mode only)	RW	0
1	DIGITALENABLE	Digital enable 0x0: Digital output disabled (at the end of the current field if interlace output when the bit is reset) 0x1: Digital output enabled	RW	0
0	LCDENABLE	LCD enable 0x0: LCD output disabled (at the end of the frame when the bit is reset) 0x1: LCD output enabled	RW	0

Table 12-164. Register Call Summary for Register DISPC_CONTROL

Display Subsystem Environment • Parallel Interface: [0] [1] [2] • Transaction Timing Diagrams: [3] • Video Port Used on Command Mode: [4]
Display Subsystem Integration • Clocks: [5] [6] • DISPC Interrupt Request: [7] [8]
Display Subsystem Functional Description • Overlay Support: [9] • Multiple Cycle Output Format: [10] • Command Mode: [11]

Table 12-164. Register Call Summary for Register DISPC_CONTROL (continued)

Display Subsystem Basic Programming Model

- Display Subsystem Reset: [12]
- Display Controller Basic Programming Model: [13] [14] [15] [16] [17] [18] [19]
- Graphics DMA Registers: [20]
- Graphics Layer Configuration Registers: [21] [22]
- Graphics Window Attributes: [23]
- Video DMA Registers: [24]
- Video Configuration Register: [25] [26]
- Video Up-/Down-Sampling Configuration: [27] [28]
- DMA Register Settings: [29]
- LCD-Specific Control Registers: [30] [31]
- LCD Attributes: [32] [33] [34] [35]
- LCD Timings: [36] [37] [38]
- LCD Overlay: [39] [40] [41]
- LCD TDM: [42] [43] [44] [45] [46] [47] [48]
- LCD Spatial/Temporal Dithering: [49] [50] [51] [52] [53]
- LCD Color Phase Rotation: [54] [55] [56]
- TV Set-Specific Control Registers: [57] [58] [59] [60]
- Digital Timings: [61]
- Digital Overlay: [62] [63] [64]
- Video Mode Transfer: [65]
- Command Mode Transfer Example 1: [66]
- Command Mode Transfer Example 2: [67]
- DISPC Control Registers: [68] [69] [70] [71] [72] [73]
- Enable: [74]
- Video Encoder Programming Sequence: [75] [76]
- SDI Configuration: [77] [78]
- Signal Features Configuration: [79]
- SDI Error Management: [80]

Display Subsystem Use Cases and Tips

- SDI Configuration: [81] [82]
- Signal Features Configuration: [83]
- Display Controller Configuration: [84] [85] [86] [87] [88] [89]
- Display Panel Timings: [90]
- LCD Enable: [91] [92] [93] [94] [95]
- Configure DISPC Timing, Window, and Color: [96] [97] [98] [99] [100] [101] [102] [103] [104]
- Enable Video Mode Using the DISPC Video Port: [105] [106] [107]
- Configure DISPC Timing, Window, and Color: [108] [109] [110] [111] [112] [113] [114] [115] [116]
- Send Frame Data to LCD Panel Using Automatic TE: [117]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [118]
- Display Controller Registers: [119]

Table 12-165. DISPC_CONFIG

Address Offset	0x044		
Physical address	0x4805 0444	**Instance**	DISC
Description	This control register configures the display controller module. Shadow register, updated on VFP start period or EVSYNC		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												TVALPHABLENDERENABLE	LCDALPHABLENDERENABLE	FIFOFILLING	FIFOHANDCHECK	CPR	FIFOMERGE	TCKDIGSELECTION	TCKDIGENABLE	TCKLCDSELECTION	TCKLCDENABLE	FUNCGATED	ACBIASGATED	VSYNCGATED	HSYNCGATED	PIXELCLOCKGATED	PIXELDATAGATED	PALETTEGAMMATABLE	LOADMODE		PIXELGATED

Bits	Field Name	Description	Type	Reset
31: 20	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00000
19	TVALPHABLENDER ENABLE	Selects the alpha blender (TV output) 0x0: Alpha blender is disabled. 0x1: The alpha blender is enabled.	RW	0
18	LCDALPHABLENDER ENABLE	Selects the alpha blender (LCD output) 0x0: Alpha blender is disabled. 0x1: The alpha blender is enabled.	RW	0
17	FIFOFILLING	Controls if the FIFO are refilled only when the LOW threshold is reached or if all FIFO are refilled when at least one of them reaches the LOW threshold. 0x0: Each FIFO is refilled when it reaches LOW threshold. 0x1: All FIFOs are refilled up to high threshold when at least one of them reaches the LOW threshold. (only active FIFOs should be considered and when reaching the end of the frame the FIFO goes to empty condition so no need to fill it again).	RW	0
16	FIFOHANDCHECK	Controls the handshake between FIFO and RFBI STALL to prevent from underflow. The bit should be set to 0 when the module is not in STALL mode. 0x0: Only the STALL signal from RFBI is used regardless of the FIFO fullness information to provide data to the RFBI module. 0x1: The STALL signal from RFBI is used in combination with the FIFO fullness information to provide data to the RFBI module only when it does not generated FIFO underflow.	RW	0
15	CPR	Color phase rotation control wr: VFP 0x0: Color phase rotation disabled 0x1: Color phase rotation enabled	RW	0
14	FIFOMERGE	FIFO merge control wr: EVSYNC or VFP 0x0: FIFO merge disabled Each FIFO is dedicated to one pipeline. 0x1: FIFO merge enabled All the FIFOS are merged into a single one to be used by the single active pipeline.	RW	0
13	TCKDIGSELECTION	Transparency color key selection (digital output) wr: EVSYNC 0x0: Graphics destination transparency color key selected in normal mode or graphics source transparency color key selected in alpha mode 0x1: Video source transparency color key selected in normal mode	RW	0

Bits	Field Name	Description	Type	Reset
12	TCKDIGENABLE	Transparency color key enabled (digital output) wr: EVSYNC 0x0: Disable the transparency color key for digital output 0x1: Enable the transparency color key for digital output	RW	0
11	TCKLCDSELECTION	Transparency color key selection (LCD output) WR: VFP 0x0: Graphics destination transparency color key selected in normal mode or graphics source transparency color key selected in alpha mode 0x1: Video source transparency color key selected in normal mode	RW	0
10	TCKLCDENABLE	Transparency color key enabled (LCD output) WR: VFP * 0x0: Disable the transparency color key for the LCD 0x1: Enable the transparency color key for the LCD	RW	0
9	FUNCGATED	Functional clocks gated enabled WR: immediate 0x0: Functional clocks gated disabled 0x1: Functional clocks gated enabled	RW	0
8	ACBIASGATED	ACBias Gated Enabled WR: VFP 0x0: AcBias Gated Disabled 0x1: AcBias Gated Enabled	RW	0
7	VSYNCGATED	VSYNC Gated Enabled WR: VFP 0x0: VSYNC Gated Disabled 0x1: VSYNC Gated Enabled	RW	0
6	HSYNCGATED	HSYNC Gated Enabled WR: VFP 0x0: HSYNC Gated Disabled 0x1: HSYNC Gated Enabled	RW	0
5	PIXELCLOCKGATED	Pixel Clock Gated Enabled WR: VFP 0x0: Pixel Clock Gated Disabled 0x1: Pixel Clock Gated Enabled	RW	0
4	PIXELDATAGATED	Pixel Data Gated Enabled WR: VFP 0x0: Pixel Data Gated Disabled 0x1: Pixel Data Gated Enabled	RW	0
3	PALETTEGAMMATABLE	Palette/Gamma Table selection WR: EVSYNC or VFP 0x0: LUT used as palette (only if graphics format is BITMAP1, 2, 4, and 8) 0x1: LUT used as gamma table (only if graphics format is NOT BITMAP1, 2, 4, and 8 or no graphics window present)	RW	0
2:1	LOADMODE	Loading Mode for the Palette/Gamma Table WR: EVSYNC or VFP 0x0: Palette/Gamma Table and data are loaded every frame. 0x1: Palette/Gamma Table to be loaded. Users set the bit when the palette/gamma table has to be loaded. H/W resets the bit when table has been loaded. (DISPC_GFX_ATTRIBUTES. GfxEnable has to be set to 1). 0x2: Frame data only loaded every frame	RW	0x0

Bits	Field Name	Description	Type	Reset
		0x3: Palette/Gamma Table and frame data loaded on first frame then switch to 10 (H/W).		
0	PIXELGATED	Pixel Gated Enable (only for Active Matrix Display) WR: VFP 0x0: Pixel clock always toggles (only in Active Matrix mode) 0x1: Pixel clock only toggles when there is valid data to display. (only in Active Matrix mode)	RW	0

Table 12-166. Register Call Summary for Register DISPC_CONFIG

Display Subsystem Environment • Transaction Timing Diagrams: [0]
Display Subsystem Integration • Autoidle Mode: [1] • Wake-Up Mode: [2]
Display Subsystem Functional Description • Normal Mode: [3] [4] [5] [6] [7] [8] [9] [10] • Alpha Mode: [11] [12] [13] [14]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [15] • Graphics DMA Registers: [16] [17] [18] • Graphics Layer Configuration Registers: [19] • Video DMA Registers: [20] [21] • Video Configuration Register: [22] • LCD-Specific Control Registers: [23] • LCD Timings: [24] [25] [26] [27] [28] • LCD Overlay: [29] [30] [31] • LCD Color Phase Rotation: [32] • TV Set-Specific Control Registers: [33] • Digital Overlay: [34] [35] [36]
Display Subsystem Use Cases and Tips • Autoidle: [37] • Display Controller Configuration: [38] • Display Panel Timings: [39]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [40]

Table 12-167. DISPC_DEFAULT_COLOR_m

Address Offset	0x04C + m * 0x04	**Indexm**	m = 0 to 1
Physical address	0x4805 044C + m * 0x04	**Instance**	DISPC
Description	The control register allows to configure the default solid background color for the LCD (DISPC_DEFAULT_COLOR_0) and for 24-bit digital output (DISPC_DEFAULT_COLOR_1). Shadow register, updated on VFP start period for DISPC_DEFAULT_COLOR_0 and EVSYNC for DISPC_DEFAULT_COLOR_1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DEFAULTCOLOR																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
23:0	DEFAULTCOLOR	24-bit RGB color value to specify the default solid color to display when there is no data from the overlays.	RW	0x000000

Table 12-168. Register Call Summary for Register DISPC_DEFAULT_COLOR_m

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0] [1]
- LCD-Specific Control Registers: [2]
- LCD Overlay: [3]
- TV Set-Specific Control Registers: [4]
- Digital Overlay: [5]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [6]

Table 12-169. DISPC_TRANS_COLOR_m

Address Offset	0x054 + m * 0x04	**Index**	m = 0 to 1
Physical address	0x4805 0454 + m * 0x04	**Instance**	DISPC
Description	The register sets the transparency color value for the video/graphics overlays for the LCD output (DISPC_TRANS_COLOR_0) for 24-bit digital output(DISPC_TRANS_COLOR_1). Shadow register, updated on VFP start period for DISPC_TRANS_COLOR_0 and EVSYNC for DISPC_TRANS_COLOR_1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRANSCOLORKEY																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
23:0	TRANSCOLORKEY	Transparency Color Key Value in RGB format [0] BITMAP 1 (CLUT), [23,1] set to 0s [1:0] BITMAP 2 (CLUT), [23,2] set to 0s [3:0] BITMAP 4 (CLUT), [23,4] set to 0s [7:0] BITMAP 8 (CLUT), [23,8] set to 0s [11:0] RGB 12, [23,12] set to 0s [15:0] RGB 16, [23,16] set to 0s [23:0] RGB 24	RW	0x000000

Table 12-170. Register Call Summary for Register DISPC_TRANS_COLOR_m

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0] [1]
- LCD-Specific Control Registers: [2]
- TV Set-Specific Control Registers: [3]
- Digital Overlay: [4]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [5]

Table 12-171. DISPC_LINE_STATUS

Address Offset	0x05C		
Physical address	0x4805 045C	**Instance**	DISC
Description	The control register indicates the current LCD panel display line number.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					LINENUMBER										

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0	R	0x000000
10:0	LINENUMBER	Current LCD panel line number Current display line number. The first active line has the value 0. During blanking lines the line number is not incremented.	R	0x7FF

Table 12-172. Register Call Summary for Register DISPC_LINE_STATUS

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [0]

Table 12-173. DISPC_LINE_NUMBER

Address Offset	0x060		
Physical address	0x4805 0460	**Instance**	DISC
Description	The control register indicates the LCD panel display line number for the interrupt and the DMA request. Shadow register, updated on VFP start period.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								LINENUMBER							

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x000000
10:0	LINENUMBER	LCD panel line number programming LCD line number defines the line on which the programmable interrupt is generated and the DMA request occurs.	RW	0x000

Table 12-174. Register Call Summary for Register DISPC_LINE_NUMBER

Display Subsystem Integration
- Display Controller DMA Request (Line Trigger): [0]

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [1]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [2]

Table 12-175. DISPC_TIMING_H

Address Offset	0x064		
Physical address	0x4805 0464	**Instance**	DISC
Description	The register configures the timing logic for the HSYNC signal. Shadow register, updated on VFP start period		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HBP																HFP								HSW							

Bits	Field Name	Description	Type	Reset
31:20	HBP	Horizontal Back Porch. Encoded value (from 1 to 4096) to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display (program to value minus 1).	RW	0x00

Bits	Field Name	Description	Type	Reset
19:8	HFP	Horizontal front porch. Encoded value (from 1 to 4096) to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted (program to value minus 1).	RW	0x00
7:0	HSW	Horizontal synchronization pulse width Encoded value (from 1 to 256) to specify the number of pixel clock periods to pulse the line clock at the end of each line (program to value minus 1).	RW	0x00

Table 12-176. Register Call Summary for Register DISPC_TIMING_H

Display Subsystem Environment • Transaction Timing Diagrams: [0] [1] [2]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [3] • LCD-Specific Control Registers: [4] • LCD Timings: [5] [6] [7] [8] [9] [10]
Display Subsystem Use Cases and Tips • Vertical and Horizontal Timings: [11] [12] [13] • Display Panel Timings: [14] [15] [16] [17] • Configure DISPC Timing, Window, and Color: [18]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [19]

Table 12-177. DISPC_TIMING_V

Address Offset	0x068		
Physical address	0x4805 0468	**Instance**	DISC
Description	The register configures the timing logic for the VSYNC signal. Shadow register, updated on VFP start period		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBP								VFP																VSW							

Bits	Field Name	Description	Type	Reset
31:20	VBP	Vertical back porch Encoded value (from 0 to 4095) to specify the number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display.	RW	0x00
19:8	VFP	Vertical front porch Encoded value (from 0 to 4095) to specify the number of line clock periods to add to the end of each frame.	RW	0x00
7:0	VSW	Vertical synchronization pulse width In active mode, encoded value (from 1 to 256) to specify the number of line clock periods (program to value minus one) to pulse the frame clock (VSYNC) pin at the end of each frame after the end of frame wait (VFP) period elapses. Frame clock uses as VSYNC signal in active mode. In passive mode, encoded value (from 1 to 256) to specify the number of extra line clock periods (program to value minus one) to insert after the vertical front porch (VFP) period has elapsed.	RW	0x00

Table 12-178. Register Call Summary for Register DISPC_TIMING_V

Display Subsystem Environment • Transaction Timing Diagrams: [0] [1] [2]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [3] • LCD-Specific Control Registers: [4] • LCD Timings: [5] [6] [7] [8] [9] [10]
Display Subsystem Use Cases and Tips • Vertical and Horizontal Timings: [11] [12] [13] • Display Panel Timings: [14] [15] [16] [17] • Configure DISPC Timing, Window, and Color: [18]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [19]

Table 12-179. DISPC_POL_FREQ

Address Offset	0x06C		
Physical address	0x4805 046C	**Instance**	DISC
Description	The register configures the signal configuration. Shadow register, updated on VFP start period		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ONOFF	RF	IEO	IPC	IHS	IVS	ACBI				ACB							

Bits	Field Name	Description	Type	Reset
31:18	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x0000
17	ONOFF	HSYNC/VSYNC Pixel clock Control On/Off 0x0: HSYNC and VSYNC are driven on opposite edges of pixel clock than pixel data 0x1: HSYNC and VSYNC are driven according to bit 16	RW	0
16	RF	Program HSYNC/VSYNC Rise or Fall 0x0: HSYNC and VSYNC are driven on falling edge of pixel clock (if bit 17 set to 1) 0x1: HSYNC and VSYNC are driven on the rising edge of pixel clock (if bit 17 set to 1)	RW	0
15	IEO	Invert output enable 0x0: Ac-bias is active high (active display mode) 0x1: Ac-bias is active low (active display mode)	RW	0
14	IPC	Invert pixel clock 0x0: Data is driven on the LCD data lines on the rising-edge of the pixel clock 0x1: Data is driven on the LCD data lines on the falling-edge of the pixel clock	RW	0
13	IHS	Invert HSYNC 0x0: Line clock pin is active high and inactive low 0x1: Line clock pin is active low and inactive high	RW	0
12	IVS	Invert VSYNC 0x0: Frame clock pin is active high and inactive low 0x1: Frame clock pin is active low and inactive high	RW	0

Bits	Field Name	Description	Type	Reset
11:8	ACBI	AC-bias pin transitions per interrupt Value (from 0 to 15) used to specify the number of AC Bias pin transitions	RW	0x0
7:0	ACB	AC-bias pin frequency Value (from 0 to 255) used to specify the number of line clocks to count before transitioning the ac-bias pin. This pin is used to periodically invert the polarity of the power supply to prevent DC charge build-up within the display.	RW	0x00

Table 12-180. Register Call Summary for Register DISPC_POL_FREQ

Display Subsystem Environment
- Transaction Timing Diagrams: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29]
- Video Port (VP) Interface: [30]

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [31]
- LCD-Specific Control Registers: [32]
- LCD Timings: [33] [34] [35] [36] [37] [38] [39] [40]
- Signal Features Configuration: [41] [42] [43] [44] [45] [46]

Display Subsystem Use Cases and Tips
- Signal Features Configuration: [47] [48] [49] [50] [51] [52]
- Display Panel Timings: [53] [54] [55] [56] [57] [58] [59]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [60]

Table 12-181. DISPC_DIVISOR

Address Offset	0x070		
Physical address	0x4805 0470	**Instance**	DISC
Description	The register configures the divisors. Shadow register, updated on VFP start period		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LCD								Reserved								PCD							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
23:16	LCD	Display Controller Logic Clock Divisor Value (from 1 to 255) to specify the frequency of the display controller logic clock based on the function clock. The value 0 is invalid.	RW	0x01
15:8	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
7:0	PCD	Pixel Clock Divisor Value (from 1 to 255) to specify the frequency of the pixel clock based on the Logic clock which is the functional clock divided by LCD. The values 0 and 1 are invalid.	RW	0x02

Table 12-182. Register Call Summary for Register DISPC_DIVISOR

Display Subsystem Environment
- Video Port (VP) Interface: [0]
- Video Port Used on Command Mode: [1]

Table 12-182. Register Call Summary for Register DISPC_DIVISOR (continued)

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [2] • LCD-Specific Control Registers: [3] • LCD Timings: [4] [5] [6] • Digital Timings: [7] [8] • Simplified Sequence When LCD and PCD Are Swapped: [9] [10]
Display Subsystem Use Cases and Tips • Display Subsystem Clock Configuration: [11] [12] [13] [14] • Display Subsystem Divider Settings to Reduce Power Consumption: [15] [16] [17] [18] [19] [20] [21] [22] • Display Panel Timings: [23] [24] [25] • Configure DISPC Timing, Window, and Color: [26] • Configure DISPC Timing, Window, and Color: [27] [28]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [29]

Table 12-183. DISPC_GLOBAL_ALPHA

Address Offset	0x0000 0074		
Physical Address	0x4805 0474	**Instance**	DISC
Description	The register defines the global alpha value for the graphics and video 2 pipelines. Shadow register, updated on VFP start period or EVSYNC for each bit field depending on the association of the each pipeline with the LCD or TV output.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VID2GLOBALALPHA								RESERVED								GFXGLOBALALPHA							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Write 0s for future compatibility. Reads return 0	RW	0x00
23:16	VID2GLOBALALPHA	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 to fully opaque.	RW	0x00
15:8	RESERVED	Write 0s for future compatibility. Reads return 0	RW	0x00
7:0	GFXGLOBALALPHA	Global alpha value from 0 to 255. 0 corresponds to fully transparent and 255 to fully opaque.	RW	0x00

Table 12-184. Register Call Summary for Register DISPC_GLOBAL_ALPHA

Display Subsystem Basic Programming Model • LCD Overlay: [0] [1] • Digital Overlay: [2] [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-185. DISPC_SIZE_DIG

Address Offset	0x078		
Physical address	0x4805 0478	**Instance**	DISC
Description	The register configures the size of the digital output field (interlace), frame (progressive) (horizontal and vertical). Shadow register, updated on EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LPP								Reserved								PPL							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	LPP	Lines per panel Encoded value (from 1 to 2048) to specify the number of lines per panel (program to value minus one).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	PPL	Pixels per line Encoded value (from 1 to 2048) to specify the number of pixels contained within each line on the display (program to value minus one)	RW	0x000

Table 12-186. Register Call Summary for Register DISPC_SIZE_DIG

Display Subsystem Functional Description • Up-/Down-Sampling: [0]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [1] • TV Set-Specific Control Registers: [2] • Digital Frame/Field Size: [3] [4] • Video Encoder Register Settings: [5] [6]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [7]

Table 12-187. DISPC_SIZE_LCD

Address Offset	0x07C		
Physical address	0x4805 047C	**Instance**	DISC
Description	The register configures the panel size (horizontal and vertical). Shadow register, updated on VFP start period		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LPP								Reserved								PPL							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	LPP	Lines per panel Encoded value (from 1 to 2048) to specify the number of lines per panel (program to value minus one).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	PPL	Pixels per line Encoded value (from 1 to 2048) to specify the number of pixels contains within each line on the display (program to value minus one). When running in normal mode (stall mode is bypassed by setting DSS.DISPC_CONTROL[11] STALLMODE =0) the line width must be set to a value multiple of 8 pixels (ex: PPL=0x7)	RW	0x000

Table 12-188. Register Call Summary for Register DISPC_SIZE_LCD

Display Subsystem Environment • Transaction Timing Diagrams: [0] [1]
Display Subsystem Functional Description • Up-/Down-Sampling: [2]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [3] • LCD-Specific Control Registers: [4] • LCD Attributes: [5] [6]
Display Subsystem Use Cases and Tips • Display Controller Configuration: [7] [8] • Display Panel Timings: [9] • Configure DISPC Timing, Window, and Color: [10] • Configure DISPC Timing, Window, and Color: [11] [12]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [13]

Table 12-189. DISPC_GFX_BAj

Address Offset	0x080 + j * 0x04	**Index**	j = 0 to 1
Physical address	0x4805 0480+ j * 0x04	**Instance**	DISC
Description	The register configures the base address of the graphics buffer displayed in the graphics window (0 & 1 :for ping-pong mechanism with external trigger, based on the field polarity, 0 only used when graphics pipeline on the LCD output and 0 & 1 when on the 24-bit digital output). Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GFXBA																															

Bits	Field Name	Description	Type	Reset
31:0	GFXBA	Graphics base address Base address of the graphics buffer (aligned on pixel size boundary) (in case 1-, 2-, and 4-BPP, byte alignment is required)	RW	0x00000000

Table 12-190. Register Call Summary for Register DISPC_GFX_BAj

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Graphics DMA Registers: [1] [2] • Rotation/Mirroring Registers: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-191. DISPC_GFX_POSITION

Address Offset	0x088		
Physical address	0x4805 0488	**Instance**	DISC
Description	The register configures the position of the graphics window. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GFXPOSY								Reserved								GFXPOSX							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	GFXPOSY	Y position of the graphics window. Encoded value (from 0 to 2047) to specify the Y position of the graphics window on the screen. The line at the top has the Y-position 0.	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	GFXPOSX	X position of the graphics window. Encoded value (from 0 to 2047) to specify the X position of the graphics window on the screen. The first pixel on the left of the screen has the X-position 0.	RW	0x000

Table 12-192. Register Call Summary for Register DISPC_GFX_POSITION

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Graphics Layer Configuration Registers: [1]
- Graphics Window Attributes: [2] [3]
- Rotation/Mirroring Registers: [4]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [5]

Table 12-193. DISPC_GFX_SIZE

Address Offset	0x08C		
Physical address	0x4805 048C	**Instance**	DISC
Description	The register configures the size of the graphics window. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GFXSIZEY								Reserved								GFXSIZEX							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
26:16	GFXSIZEY	Number of lines of the graphics window. Encoded value (from 1 to 2048) to specify the number of lines of the graphics window (program to value minus one).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
10:0	GFXSIZEX	Number of pixels of the graphics window. Encoded value (from 1 to 2048) to specify the number of pixels per line of the graphics window (program to value minus one).	RW	0x000

Table 12-194. Register Call Summary for Register DISPC_GFX_SIZE

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Graphics Layer Configuration Registers: [1]
- Graphics Window Attributes: [2] [3]
- Rotation/Mirroring Registers: [4]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [5]

Table 12-195. DISPC_GFX_ATTRIBUTES

Address Offset	0x0A0		
Physical address	0x4805 04A0	**Instance**	DISC
Description	The register configures the graphics attributes. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																GFXSELFREFRESH	GFXARBITRATION	GFXROTATION		GFXFIFOPRELOAD	GFXENDIANNESS	GFXNIBBLEMODE	GFXCHANNELOUT	GFXBURSTSIZE		GFXREPLICATIONENABLE	GFXFORMAT				GFXENABLE

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000000
15	GFXSELFREFRESH	Enables the self refresh of the graphics window from its own FIFO only. 0x0: The graphics pipeline accesses the interconnect to fetch data from the system memory 0x1: The graphics pipeline does not need anymore to fetch data from memory. Only the graphics FIFO is used. It takes effect after the frame has been loaded in the FIFO	RW	0
14	GFXARBITRATION	Determines the priority of the graphics pipeline. The graphics pipeline is one of the high priority pipeline. The arbitration wheel gives always the priority first to the high priority pipelines using round-robin between them. When there is only normal priority pipelines sending requests, the round-robin applies between them. 0x0: The graphics pipeline is one of the normal priority pipeline. 0x1: The graphics pipeline is one of the high priority pipeline.	RW	0
13:12	GFXROTATION	Graphics rotation flag (used only in case of RGB24 packed format) 0x0: No rotation 0x1: Rotation by 90 degrees 0x2: Rotation by 180 degrees 0x3: Rotation by 270 degrees	RW	0x0
11	GFXFIFOPRELOAD	Graphics preload value 0x0: H/W prefetches pixels up to the preload value defined in the preload register. 0x1: H/W prefetches pixels up to high threshold value.	RW	0
10	GFXENDIANNESS	Graphics endianness 0x0: Little endian operation is selected. 0x1: Big endian operation is selected.	RW	0
9	GFXNIBBLEMODE	Graphics Nibble Mode (only for 1-, 2- and 4-BPP) 0x0: Nibble mode is disabled 0x1: Nibble mode is enabled	RW	0
8	GFXCHANNELOUT	Graphics Channel Out configuration wr: immediate 0x0: LCD output selected 0x1: 24-bit output selected	RW	0

Bits	Field Name	Description	Type	Reset
7:6	GFXBURSTSIZE	Graphics DMA Burst Size	RW	0x0
		0x0: 4x32bit bursts		
		0x1: 8x32bit bursts		
		0x2: 16x32bit bursts		
		0x3: Reserved		
5	GFXREPLICATION ENABLE	GfxReplicationEnable	RW	0
		0x0: Disable Graphics replication logic		
		0x1: Enable Graphics replication logic		
4:1	GFXFORMAT	Graphics format; Other enums: Reserved (0x7, 0xA, 0xB and 0xF)	RW	0x0
		0x0: BITMAP 1 (CLUT)		
		0x1: BITMAP 2 (CLUT)		
		0x2: BITMAP 4 (CLUT)		
		0x3: BITMAP 8 (CLUT)		
		0x4: RGB 12 (un-packed in 16-bit container)		
		0x5: ARGB16		
		0x6: RGB 16		
		0x8: RGB 24 (un-packed in 32-bit container)		
		0x9: RGB 24 (packed in 24-bit container)		
		0xC: ARGB32		
		0xD: RGBA32		
		0xE: RGBx 32 (24-bit RGB aligned on MSB of the 32-bit container)		
0	GFXENABLE	GfxEnable	RW	0
		0x0: Graphics disabled (graphics pipeline inactive and graphics window not present)		
		0x1: Graphics enabled (graphics pipeline active and graphics window present on the screen)		

Table 12-196. Register Call Summary for Register DISPC_GFX_ATTRIBUTES

Display Subsystem Basic Programming Model

- Display Controller Basic Programming Model: [0]
- Graphics DMA Registers: [1] [2] [3] [4] [5] [6] [7]
- Graphics Layer Configuration Registers: [8] [9]
- Graphics Window Attributes: [10] [11] [12] [13] [14]
- Rotation/Mirroring Registers: [15]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [16]
- Display Controller Registers: [17]

Table 12-197. DISPC_GFX_FIFO_THRESHOLD

Address Offset	0x0A4		
Physical address	0x4805 04A4	**Instance**	DISC
Description	The register configures the graphics FIFO. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GFXFIFOHIGHTHRESHOLD								Reserved								GFXFIFOLOWTHRESHOLD							

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
27:16	GFXFIFOHIGH THRESHOLD	Graphics FIFO High Threshold Number of bytes defining the threshold value.	RW	0x3FF
15:12	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
11:0	GFXFIFOLOW THRESHOLD	Graphics FIFO Low Threshold Number of bytes defining the threshold value	RW	0x3C0

Table 12-198. Register Call Summary for Register DISPC_GFX_FIFO_THRESHOLD

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Graphics DMA Registers: [1] [2] [3] [4] [5] [6]
- Rotation/Mirroring Registers: [7] [8] [9]

Display Subsystem Use Cases and Tips
- FIFO Thresholds: [10] [11]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [12]

Table 12-199. DISPC_GFX_FIFO_SIZE_STATUS

Address Offset	0x0A8		
Physical address	0x4805 04A8	**Instance**	DISC
Description	This register defines the graphics FIFO size.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					GFXFIFOSIZE										

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0	R	0x000000
10:0	GFXFIFOSIZE	Graphics FIFO Size Number of bytes defining the FIFO value.	R	0x400

Table 12-200. Register Call Summary for Register DISPC_GFX_FIFO_SIZE_STATUS

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [0]

Table 12-201. DISPC_GFX_ROW_INC

Address Offset	0x0AC		
Physical address	0x4805 04AC	**Instance**	DISC
Description	The register configures the number of bytes to increment at the end of the row. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GFXROWINC																															

Bits	Field Name	Description	Type	Reset
31:0	GFXROWINC	Number of bytes to increment at the end of the row Encoded signed value (from -2^{31}- 1 to 2^{31}) to specify the number of bytes to increment at the end of the row in the graphics buffer. The value 0 is invalid. The value 1 means next pixel. The value 1+n*BPP means increment of n pixels. The value 1- (n+1)*BPP means decrement of n pixels.	RW	0x00000001

Table 12-202. Register Call Summary for Register DISPC_GFX_ROW_INC

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Graphics DMA Registers: [1]
- Graphics Window Attributes: [2] [3]
- Rotation/Mirroring Registers: [4]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [5]

Table 12-203. DISPC_GFX_PIXEL_INC

Address Offset	0x0B0		
Physical address	0x4805 04B0	**Instance**	DISC
Description	The register configures the number of bytes to increment between two pixels. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GFXPIXELINC															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x0000
15:0	GFXPIXELINC	Number of bytes to increment between two pixels Encoded signed value (from -2^{15}- 1 to 2^{15}) to specify the number of bytes between two pixels in the graphics buffer. The value 0 is invalid. The value 1 means next pixel. The value 1+n*BPP means increment of n pixels. The value 1- (n+1)*BPP means decrement of n pixels.	RW	0x0001

Table 12-204. Register Call Summary for Register DISPC_GFX_PIXEL_INC

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Graphics DMA Registers: [1]
- Rotation/Mirroring Registers: [2]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [3]

Table 12-205. DISPC_GFX_WINDOW_SKIP

Address Offset	0x0B4		
Physical address	0x4805 04B4	**Instance**	DISC
Description	The register configures the number of bytes to skip during video window display. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GFXWINDOWSKIP																															

Bits	Field Name	Description	Type	Reset
31:0	GFXWINDOWSKIP	Number of bytes to skip during video window #1. Encoded signed value (from -2^{31}-1 to 2^{31}) to specify the number of bytes to skip in the graphics buffer when video window #1 is displayed on top of the graphics and no transparency color is enabled.	RW	0x00000000

Table 12-206. Register Call Summary for Register DISPC_GFX_WINDOW_SKIP

Display Subsystem Functional Description • Overlay Support: [0]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [1] • Graphics Window Attributes: [2] [3] [4] [5]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [6]

Table 12-207. DISPC_GFX_TABLE_BA

Address Offset	0x0B8		
Physical address	0x4805 04B8	**Instance**	DISC
Description	The register configures the base address of the palette buffer or the gamma table buffer. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GFXTABLEBA																															

Bits	Field Name	Description	Type	Reset
31:0	GFXTABLEBA	Base address of the palette/gamma table buffer (24-bit entries in 32-bit containers, aligned on 32-bit boundary).	RW	0x00000000

Table 12-208. Register Call Summary for Register DISPC_GFX_TABLE_BA

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Graphics DMA Registers: [1] [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

Table 12-209. DISPC_VIDn_BAj

Address Offset	0x0BC + ((n–1)* 0x90) + (j* 0x04)	**Index**	n = 1 for VID1 or 2 for VID2 j = 0 to 1
Physical address	0x4805 04BC+ ((n–1)* 0x90) + (j* 0x04)	**Instance**	DISC
Description	The register configures the base address of the video buffer for video window #n(#j for ping-pong mechanism with external trigger, based on the field polarity: 0 for even field and 1 for odd field). Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VIDBA																															

Bits	Field Name	Description	Type	Reset
31:0	VIDBA	Video base address Base address of the video buffer (aligned on pixel size boundary)	RW	0x00000000

Table 12-210. Register Call Summary for Register DISPC_VIDn_BAj

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Video DMA Registers: [1] • Rotation/Mirroring Registers: [2] • DMA Register Settings: [3] [4]
Display Subsystem Use Cases and Tips • Register List: [5]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [6] [7]

Table 12-211. DISPC_VIDn_POSITION

Address Offset	0x0C4 + ((–1) *0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04C4 + ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the position of video window #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					VIDPOSY											Reserved					VIDPOSX										

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	VIDPOSY	Y position of video window #n Encoded value (from 0 to 2047) to specify the Y position of video window #n. The line at the top has the Y-position 0.	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	VIDPOSX	X position of video window #n Encoded value (from 0 to 2047) to specify the X position of video window #n. The first pixel on the left of the display screen has the X-position 0.	RW	0x000

Table 12-212. Register Call Summary for Register DISPC_VIDn_POSITION

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Video Configuration Register: [1] • Video Window Attributes: [2] [3] • Rotation/Mirroring Registers: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5] [6]

Table 12-213. DISPC_VIDn_SIZE

Address Offset	0x0C8+((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04C8+((–1)* 0x90)	**Instance**	DISC
Description	The register configures the size of video window #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VIDSIZEY								Reserved								VIDSIZEX							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	VIDSIZEY	Number of lines of video #n Encoded value (from 1 to 2048) to specify the number of lines of video window #n (program to value minus one).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	VIDSIZEX	Number of pixels of video window #n Encoded value (from 1 to 2048) to specify the number of pixels of video window #n (program to value minus one).	RW	0x000

Table 12-214. Register Call Summary for Register DISPC_VIDn_SIZE

Display Subsystem Basic Programming Model

- Display Controller Basic Programming Model: [0]
- Video Configuration Register: [1]
- Video Window Attributes: [2] [3]
- Rotation/Mirroring Registers: [4]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [5] [6]

Table 12-215. DISPC_VIDn_ATTRIBUTES

Address Offset	0x0CC+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04CC+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the attributes of video window #n such as format, resizeenable, shadow register, updated on VFP start period, or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								VIDSELFREFRESH	VIDARBITRATION	VIDLINEBUFFERSPLIT	VIDVERTICALTAPS	VIDDMAOPTIMIZATION	VIDFIFOPRELOAD	VIDROWREPEATENABLE	VIDENDIANNESS	VIDCHANNELOUT	VIDBURSTSIZE		VIDROTATION		VIDFULLRANGE	VIDREPLICATIONENABLE	VIDCOLORCONVENABLE	VIDVRESIZECONF	VIDHRESIZECONF	VIDRESIZEENABLE	VIDFORMAT					VIDENABLE

Bits	Field Name	Description	Type	Reset
31: 25	RESERVED	Write 0s for future compatibility. Read returns 0.	RW	0x0000
24	VIDSELFREFRESH	Enables the self refresh of the video window from its own FIFO only.	RW	0

Bits	Field Name	Description	Type	Reset
		0x0: The video pipeline accesses the interconnect to fetch data from the system memory 0x1: The video pipeline does not need anymore to fetch data from memory. Only the video FIFO is used. It takes effect after the frame has been loaded in the FIFO		
23	VIDARBITRATION	Determines the priority of the video pipeline. The video pipeline is one of the high priority pipeline. The arbitration wheel gives always the priority first to the high priority pipelines using round-robin between them. When there is only normal priority pipelines sending requests, the round-robin applies between them. 0x0: The video pipeline is one of the normal priority pipeline. 0x1: The video pipeline is one of the high priority pipeline.	RW	0
22	VIDLINEBUFFER SPLIT	Video vertical line buffer split 0x0: Vertical line buffers are not split. 0x1: Vertical line buffers are split into two.	RW	0
21	VIDVERTICALTAPS	Video vertical resize tap number 0x0: Three taps are used for the vertical filtering logic. The other two taps are not used. 0x1: Five taps are used for the vertical filtering logic.	RW	0
20	VIDDMAOPTI MIZATION	Video optimization in case of 0x0: The DMA engine fetches one pixel for each 32-bit OCP request (RGB16 and YUV422) while doing 90- and 270-degree rotation (accessing on-chip memory and off-chip memory). 0x1: The DMA engine fetches two pixels for each 32-bit OCP request (RGB16 and YUV422) while doing 90- and 270-degree rotation (accessing on-chip memory and off-chip memory).	RW	0
19	VIDFIFOPRELOAD	Video preload value 0x0: H/W prefetches pixels up to the preload value defined in the preload register. 0x1: H/W prefetches pixels up to the high threshold value.	RW	0
18	VIDROWREPEAT ENABLE	Video Row Repeat (YUV case only when rotating 90 or 270-degree) 0x0: Row of VIDn won't be read twice. 0x1: The Row data are fetched twice to extract both the Y components	RW	0
17	VIDENDIANNESS	Video Endianness 0x0: Little endian operation is selected. 0x1: Big endian operation is selected.	RW	0
16	VIDCHANNELOUT	Video Channel Out configuration wr: Immediate 0x0: LCD output selected 0x1: 24 bit output selected	RW	0
15:14	VIDBURSTSIZE	Video DMA Burst Size 0x0: 4x32bit bursts 0x1: 8x32bit bursts 0x2: 16x32bit bursts 0x3: Reserved	RW	0x0
13:12	VIDROTATION	Video Rotation Flag 0x0: No rotation or VidFormat is RGB 0x1: Rotation by 90 degrees 0x2: Rotation by 180 degrees	RW	0x0

Bits	Field Name	Description	Type	Reset
		0x3: Rotation by 270 degrees		
11	VIDFULLRANGE	VidFullRange	RW	0
		0x0: Limited range selected: 16 subtracted from Y before color space conversion		
		0x1: Full range selected: Y is not modified before the color space conversion		
10	VIDREPLICATION ENABLE	VidReplicationEnable	RW	0
		0x0: Disable Video replication logic		
		0x1: Enable Video replication logic		
9	VIDCOLORCONV ENABLE	VidColorConvEnable	RW	0
		0x0: Disable Color Space Conversion CbYCr to RGB		
		0x1: Enable Color Space Conversion CbYCr to RGB		
8	VIDVRESIZECONF	Video Vertical Resize Configuration	RW	0
		0x0: Up-sampling selected		
		0x1: Down-sampling selected		
7	VIDHRESIZECONF	Video Horizontal Resize Configuration	RW	0
		0x0: Up-sampling selected		
		0x1: Down-sampling selected		
6:5	VIDRESIZEENABLE	Video Resize Enable	RW	0x0
		0x0: Disable the resize processing		
		0x1: Enable the horizontal resize processing		
		0x2: Enable the vertical resize processing		
		0x3: Enable both horizontal and vertical resize processing		
4:1	VIDFORMAT (Video 1 channel)	Video1 channel Format; Other enums: Reserved (all other values between 0x0 and 0x3, 0x5, 0x7, and between 0xC and 0xF)	RW	0x0
		0x4: RGB12 (16-bit container)		
		0x6: RGB 16		
		0x8: RGB 24 (unpacked in 32-bit container)		
		0x9: RGB 24 (packed in24-bit container)		
		0xA: YUV2 4:2:2 co-sited		
		0xB: UYVY 4:2:2 co-sited		
	VIDFORMAT (Video 2 channel)	Video2 channel Format; Other enums: Reserved (all other values: 0x0 and 0x3, 0x7, and 0xF)	RW	0x0
		0x4: RGB 12 (16-bit container)		
		0x5: ARGB 16		
		0x6: RGB 16		
		0x8: RGB 24 (un-packed in 32-bit container)		
		0x9: RGB 24 (packed in 24-bit container)		
		0xA: YUV2 4:2:2 co-sited		
		0xB: UYVY 4:2:2 co-sited		
		0xC: ARGB 32		
		0xD: RGBA 32		
		0xE: RGBx 32 (24-bit RGB aligned on MSB of the 32-bit container)		
0	VIDENABLE	VidEnable	RW	0
		0x0: Video disabled (video pipeline inactive and window not present)		
		0x1: Video enabled (video pipeline active and window present on the screen)		

Table 12-216. Register Call Summary for Register DISPC_VIDn_ATTRIBUTES

Display Subsystem Functional Description • Up-/Down-Sampling: [0] • Overlay Support: [1]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [2] • Video DMA Registers: [3] [4] [5] [6] [7] [8] • Video Configuration Register: [9] [10] • Video Window Attributes: [11] [12] [13] [14] [15] • Video Up-/Down-Sampling Configuration: [16] [17] [18] [19] [20] [21] [22] • Rotation/Mirroring Registers: [23] • Additional configuration when using YUV format: [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41]
Display Subsystem Use Cases and Tips • Vertical Filtering: [42] • Register List: [43] • Enabling: [44] [45] • Video1 Channel Configuration: [46]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [47] [48] • Display Controller Registers: [49]

Table 12-217. DISPC_VIDn_FIFO_THRESHOLD

Address Offset	0x0D0+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04D0+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the video FIFO associated with video pipeline #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				VIDFIFOHIGHTHRESHOLD												Reserved				VIDFIFOLOWTHRESHOLD											

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
27:16	VIDFIFOHIGH THRESHOLD	Video FIFO high threshold Number of bytes defining the threshold value	RW	0x3FF
15:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
11:0	VIDFIFOLOW THRESHOLD	Video FIFO low threshold Number of bytes defining the threshold value	RW	0x3C0

Table 12-218. Register Call Summary for Register DISPC_VIDn_FIFO_THRESHOLD

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Video DMA Registers: [1] [2] [3] [4] [5] [6] • Rotation/Mirroring Registers: [7]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [8] [9]

Table 12-219. DISPC_VIDn_FIFO_SIZE_STATUS

Address Offset	0x0D4+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04D4+((–1)* 0x90)	**Instance**	DISC
Description	The register defines the video FIFO size for video pipeline #n.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					VIDFIFOSIZE										

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x000000
10:0	VIDFIFOSIZE	Video FIFO Size Number of bytes defining the FIFO value	R	0x400

Table 12-220. Register Call Summary for Register DISPC_VIDn_FIFO_SIZE_STATUS

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [0] [1]

Table 12-221. DISPC_VIDn_ROW_INC

Address Offset	0x0D8+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04D8+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the number of bytes to increment at the end of the row for the buffer associated with video window #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VIDROWINC																															

Bits	Field Name	Description	Type	Reset
31:0	VIDROWINC	Number of bytes to increment at the end of the row Encoded signed value (from -2^{31}- 1 to 2^{31}) to specify the number of bytes to increment at the end of the row in the video buffer. The value 0 is invalid. The value 1 means next pixel. The value 1+n*BPP means increment of n pixels. The value 1- (n+1)*BPP means decrement of n pixels.	RW	0x00000001

Table 12-222. Register Call Summary for Register DISPC_VIDn_ROW_INC

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Video DMA Registers: [1]
- Video Window Attributes: [2] [3]
- Rotation/Mirroring Registers: [4]
- DMA Register Settings: [5] [6]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [7] [8]

Table 12-223. DISPC_VIDn_PIXEL_INC

Address Offset	0x0DC+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04DC+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the number of bytes to increment between two pixels for the buffer associated with video window #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VIDPIXELINC															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x0000
15:0	VIDPIXELINC	Number of bytes to increment at the end of the row Encoded signed value (from -2^{15}- 1 to 2^{15}) to specify the number of bytes between two pixels in the video buffer. The value 0 is invalid. The value 1 means next pixel. The value 1+n*BPP means increment of n pixels. The value 1- (n+1)*BPP means decrement of n pixels	RW	0x0001

Table 12-224. Register Call Summary for Register DISPC_VIDn_PIXEL_INC

Display Subsystem Basic Programming Model

- Display Controller Basic Programming Model: [0]
- Video DMA Registers: [1]
- Rotation/Mirroring Registers: [2]
- DMA Register Settings: [3] [4]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [5] [6]

Table 12-225. DISPC_VIDn_FIR

Address Offset	0x0E0+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04E0+((–1)* 0x90)	**Instance**	DISC
Description	The register configures the resize factors for horizontal and vertical up-/down-sampling of video window #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			VIDFIRVINC													Reserved			VIDFIRHINC												

Bits	Field Name	Description	Type	Reset
31:29	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
28:16	VIDFIRVINC	Vertical increment of the up-/down-sampling filter Encoded value (from 1 to 4096). The value 0 is invalid. Values greater than 4096 are invalid.	RW	0x0000
15:13	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
12:0	VIDFIRHINC	Horizontal increment of the up-/down-sampling filter Encoded value (from 1 to 4096). The value 0 is invalid. Values greater than 4096 are invalid.	RW	0x0000

Table 12-226. Register Call Summary for Register DISPC_VIDn_FIR

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Video Configuration Register: [1] • Video Up-/Down-Sampling Configuration: [2] [3]
Display Subsystem Use Cases and Tips • Register List: [4] • Factor: [5] [6] • Coefficients: [7] [8] [9] [10] • Video1 Channel Configuration: [11]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [12] [13]

Table 12-227. DISPC_VIDn_PICTURE_SIZE

Address Offset	0x0E4+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 04E4+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the size of the video picture associated with video layer #nbefore up-/down-scaling. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VIDORGSIZEY								Reserved								VIDORGSIZEX							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	VIDORGSIZEY	Number of lines of the video picture Encoded value (from 1 to 2048) to specify the number of lines of the video picture in memory (program to value minus one).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	VIDORGSIZEX	Number of pixels of the video picture Encoded value (from 1 to 2048) to specify the number of pixels of the video picture in memory (program to value minus one). The size is limited to the size of the line buffer of the vertical sampling block in case the video picture is processed by the vertical filtering unit.(1)	RW	0x000

(1) For 5-tap RGB16 and YUV422 picture formats, the width of the input picture must be a multiple of 2 pixels and more than 5 pixels. This leads to the following register configuration:
- DISPC_VIDn_ATTRIBUTES[21] VIDVERTICALTAPS is set to 1.
- DISPC_VIDn_PICTURE_SIZE[10:0] VIDORGSIZEX must be even and more than 4.

Table 12-228. Register Call Summary for Register DISPC_VIDn_PICTURE_SIZE

Display Subsystem Functional Description • Up-/Down-Sampling: [0]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [1] • Video DMA Registers: [2] • Video Configuration Register: [3] • Video Window Attributes: [4] [5] • Video Up-/Down-Sampling Configuration: [6]
Display Subsystem Use Cases and Tips • Vertical Filtering: [7]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [8] [9] • Display Controller Registers: [10]

Table 12-229. DISPC_VIDn_ACCUI

Address Offset	0x0E8+ ((–1)* 0x90)+ (I*0x04)	**Index**	n = 1 for VID1 or 2 for VID2 I = 0 to 1
Physical address	0x4805 04E8 + ((n-1)*0x90)+ (I*0x04)	**Instance**	DISC
Description	The register configures the resize accumulator init values for horizontal and vertical up-/down-sampling of video window #n (#I for ping-pong mechanism with external trigger, based on the field polarity) Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VIDVERTICALACCU								Reserved								VIDHORIZONTALACCU							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
25:16	VIDVERTICAL ACCU	Vertical initialization accu value. Encoded value (from 0 to 1023).	RW	0x000
15:10	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
9:0	VIDHORIZONTAL ACCU	Horizontal initialization accu value. Encoded value (from 0 to 1023).	RW	0x000

Table 12-230. Register Call Summary for Register DISPC_VIDn_ACCUI

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Video Up-/Down-Sampling Configuration: [1] [2]

Display Subsystem Use Cases and Tips
- Register List: [3]
- Initial Phase: [4] [5]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [6] [7]

Table 12-231. DISPC_VIDn_FIR_COEF_Hi

Address Offset	0x0F0+ ((–1)* 0x90) + (i* 0x08)	**Index**	n = 1 for VID1 or 2 for VID2 i = 0 to 7
Physical address	0x4805 04F0+ ((n-1)*0x90) + (i*0x08)	**Instance**	DISC
Description	The bank of registers configure the up-/down-scaling coefficients for the vertical and horizontal resize of the video picture associated with video window #n for the phases from 0 to 7. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VIDFIRHC3								VIDFIRHC2								VIDFIRHC1								VIDFIRHC0							

Bits	Field Name	Description	Type	Reset
31:24	VIDFIRHC3	Signed coefficient C3 for the horizontal up-/down-scaling with the phase n	RW	0x00
23:16	VIDFIRHC2	Unsigned coefficient C2 for the horizontal up-/down-scaling with the phase n	RW	0x00
15:8	VIDFIRHC1	Signed coefficient C1 for the horizontal up-/down-scaling with the phase n	RW	0x00
7:0	VIDFIRHC0	Signed coefficient C0 for the horizontal up-/down-scaling with the phase n	RW	0x00

Table 12-232. Register Call Summary for Register DISPC_VIDn_FIR_COEF_Hi

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Video Configuration Register: [1]
- Video Up-/Down-Sampling Configuration: [2] [3] [4]

Display Subsystem Use Cases and Tips
- Register List: [5] [6] [7] [8] [9]
- Coefficients: [10] [11] [12]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [13] [14]

Table 12-233. DISPC_VIDn_FIR_COEF_HVi

Address Offset	0x0F4+ ((–1)* 0x90) + (i* 0x08)	**Index**	n = 1 for VID1 or 2 for VID2 i = 0 to 7
Physical address	0x4805 04F4+ ((–1)* 0x90) + (i* 0x08)	**Instance**	DISC
Description	The bank of registers configure the down/up-/down-scaling coefficients for the vertical and horizontal resize of the video picture associated with video window #n for the phases from 0 to 7. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VIDFIRVC2								VIDFIRVC1								VIDFIRVC0								VIDFIRHC4							

Bits	Field Name	Description	Type	Reset
31:24	VIDFIRVC2	Signed coefficient C2 for the vertical up-/down-scaling with the phase n	RW	0x00
23:16	VIDFIRVC1	Unsigned coefficient C1 for the vertical up-/down-scaling with the phase n	RW	0x00
15:8	VIDFIRVC0	Signed coefficient C0 for the vertical up-/down-scaling with the phase n	RW	0x00
7:0	VIDFIRHC4	Signed coefficient C4 for the horizontal up-/down-scaling with the phase n	RW	0x00

Table 12-234. Register Call Summary for Register DISPC_VIDn_FIR_COEF_HVi

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Video Configuration Register: [1]
- Video Up-/Down-Sampling Configuration: [2] [3] [4] [5] [6] [7]

Display Subsystem Use Cases and Tips
- Register List: [8] [9] [10] [11] [12] [13] [14] [15]
- Coefficients: [16] [17] [18] [19] [20]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [21] [22]

Table 12-235. DISPC_VIDn_CONV_COEF0

Address Offset	0x130+((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 0530+((–1)* 0x90)	**Instance**	DISC
Description	The register configures the color space conversion matrix coefficients for video pipeline #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RCR								Reserved								RY							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	RCR	RCr Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	RY	RY Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000

Table 12-236. Register Call Summary for Register DISPC_VIDn_CONV_COEF0

Display Subsystem Basic Programming Model
- Rotation/Mirroring Registers: [0] [1]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [2] [3]

Table 12-237. DISPC_VIDn_CONV_COEF1

Address Offset	0x134+((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 0534+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the color space conversion matrix coefficients for video pipeline #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GY								Reserved								RCB							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	GY	GY Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	RCB	RCb Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000

Table 12-238. Register Call Summary for Register DISPC_VIDn_CONV_COEF1

Display Subsystem Basic Programming Model
- Rotation/Mirroring Registers: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1] [2]

Table 12-239. DISPC_VIDn_CONV_COEF2

Address Offset	0x138+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 0538+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the color space conversion matrix coefficients for video pipeline #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								GCB								Reserved								GCR							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	GCB	GCb Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	GCR	GCr Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000

Table 12-240. Register Call Summary for Register DISPC_VIDn_CONV_COEF2

Display Subsystem Basic Programming Model • Rotation/Mirroring Registers: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1] [2]

Table 12-241. DISPC_VIDn_CONV_COEF3

Address Offset	0x13C+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 053C+ ((–1)* 0x90)	**Instance**	DISC
Description	The register configures the color space conversion matrix coefficients for video pipeline #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BCR								Reserved								BY							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
26:16	BCR	BCr coefficient Encoded signed value (from -1024 to 1023).	RW	0x000
15:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x00
10:0	BY	BY coefficient Encoded signed value (from -1024 to 1023).	RW	0x000

Table 12-242. Register Call Summary for Register DISPC_VIDn_CONV_COEF3

Display Subsystem Basic Programming Model • Rotation/Mirroring Registers: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1] [2]

Table 12-243. DISPC_VIDn_CONV_COEF4

Address Offset	0x140+ ((–1)* 0x90)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 0540+ ((–1)* 0x90)	**Instance**	DISPC
Description	The register configures the color space conversion matrix coefficients for video pipeline #n. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					BCB										

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x000000
10:0	BCB	BCb Coefficient Encoded signed value (from -1024 to 1023).	RW	0x000

Table 12-244. Register Call Summary for Register DISPC_VIDn_CONV_COEF4

Display Subsystem Basic Programming Model • Rotation/Mirroring Registers: [0] [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2] [3]

Table 12-245. DISPC_DATA_CYCLEk

Address Offset	0x1D4 + (k* 0x04)	**Index**	k = 0 to 2
Physical address	0x4805 05D4+ (k * 0x04)	**Instance**	DISPC
Description	The control register configures the output data format for ith (1st, 2nd or 3rd) cycle. Shadow register, updated on VFP start period.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				BITALIGNMENTPIXEL2				Reserved			NBBITSPIXEL2					Reserved				BITALIGNMENTPIXEL1				Reserved			NBBITSPIXEL1				

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x0
27:24	BITALIGNMENT PIXEL2	Bit alignment Alignment of the bits from pixel#2 on the output interface	RW	0x0
23:21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
20:16	NBBITSPIXEL2	Number of bits Number of bits from the pixel #2 (value from 0 to 16 bits). The values from 17 to 31 are invalid.	RW	0x00
15:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
11:8	BITALIGNMENT PIXEL1	Bit alignment Alignment of the bits from pixel#1 on the output interface	RW	0x0
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4:0	NBBITSPIXEL1	Number of bits Number of bits from the pixel #1 (value from 0 to 16 bits). The values from 17 to 31 are invalid.	RW	0x00

Table 12-246. Register Call Summary for Register DISPC_DATA_CYCLEk

Display Subsystem Functional Description • Multiple Cycle Output Format: [0] [1] [2]
Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [3] • LCD-Specific Control Registers: [4] • LCD TDM: [5] [6] [7]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [8]

Table 12-247. DISPC_VIDn_FIR_COEF_Vi

Address Offset	0x1E0+ ((–1)* 0x20) + (i* 0x04)	**Index**	n = 1 for VID1 or 2 for VID2 i = 0 to 7
Physical address	0x4805 05E0+ ((n-1)*0x20) + (i* 0x04)	**Instance**	DISC
Description	This bank of registers configures the down/up/down-scaling coefficients for the vertical resize of the video picture associated with video window #n for phases 0 to 7. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VIDFIRVC22								VIDFIRVC00							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	VIDFIRVC22	Signed coefficient C22 for vertical up/down-scaling with phase n	RW	0x00
7:0	VIDFIRVC00	Signed coefficient C00 for vertical up/down-scaling with phase n	RW	0x00

Table 12-248. Register Call Summary for Register DISPC_VIDn_FIR_COEF_Vi

Display Subsystem Basic Programming Model • Display Controller Basic Programming Model: [0] • Video Up-/Down-Sampling Configuration: [1] [2]
Display Subsystem Use Cases and Tips • Register List: [3] [4] [5] • Coefficients: [6] • Up-Sampling: [7] • Down-Sampling: [8]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [9] [10]

Table 12-249. DISPC_CPR_COEF_R

Address Offset	0x220		
Physical address	0x4805 0620	**Instance**	DISC
Description	This register configures the color phase rotation matrix coefficients for the red component. Shadow register, updated on VFP start period.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RR										Reserved		RG									Reserved			RB							

Bits	Field Name	Description	Type	Reset
31:22	RR	RR coefficient Encoded signed value (from -512 to 511)	RW	0x000
21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
20:11	RG	RG coefficient Encoded signed value (from -512 to 511)	RW	0x000
10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
9:0	RB	RB coefficient Encoded signed value (from -512 to 511)	RW	0x000

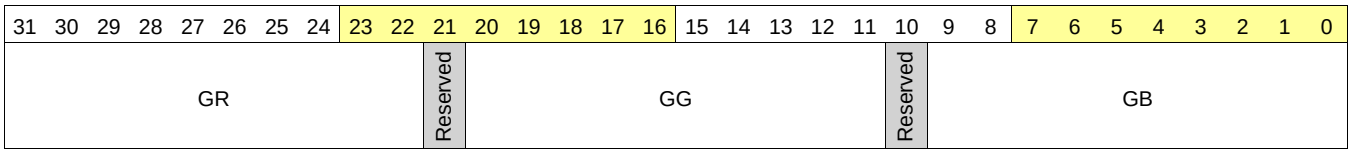
Table 12-250. Register Call Summary for Register DISPC_CPR_COEF_R

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- LCD-Specific Control Registers: [1]
- LCD Color Phase Rotation: [2] [3]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [4]

Table 12-251. DISPC_CPR_COEF_G

Address Offset	0x224		
Physical address	0x4805 0624	**Instance**	DISC
Description	This register configures the color phase rotation matrix coefficients for the green component. Shadow register, updated on VFP start period.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:22	GR	GR coefficient Encoded signed value (from -512 to 511)	RW	0x000
21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
20:11	GG	GG coefficient Encoded signed value (from -512 to 511)	RW	0x000
10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
9:0	GB	GB coefficient Encoded signed value (from -512 to 511)	RW	0x000

Table 12-252. Register Call Summary for Register DISPC_CPR_COEF_G

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- LCD-Specific Control Registers: [1]
- LCD Color Phase Rotation: [2] [3]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [4]

Table 12-253. DISPC_CPR_COEF_B

Address Offset	0x228		
Physical address	0x4805 0628	**Instance**	DISC
Description	This register configures the color phase rotation matrix coefficients for the blue component. Shadow register, updated on VFP start period.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR										Reserved		BG									Reserved			BB							

Bits	Field Name	Description	Type	Reset
31:22	BR	BR coefficient Encoded signed value (from -512 to 511)	RW	0x000
21	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
20:11	BG	BG coefficient Encoded signed value (from -512 to 511)	RW	0x000
10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
9:0	BB	BB coefficient Encoded signed value (from -512 to 511)	RW	0x000

Table 12-254. Register Call Summary for Register DISPC_CPR_COEF_B

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- LCD-Specific Control Registers: [1]
- LCD Color Phase Rotation: [2] [3]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [4]

Table 12-255. DISPC_GFX_PRELOAD

Address Offset	0x22C		
Physical address	0x4805 062C	**Instance**	DISC
Description	This register configures the graphics FIFO. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				PRELOAD											

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:0	PRELOAD	Graphics preload value: Number of bytes defining the preload value. Constraint: Maximum value is (FIFO size - DMA burst size - 8) bytes	RW	0x100

Table 12-256. Register Call Summary for Register DISPC_GFX_PRELOAD

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Graphics DMA Registers: [1] [2] [3] [4]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [5]

Table 12-257. DISPC_VIDn_PRELOAD

Address Offset	0x230+ ((–1)* 0x04)	**Index**	n = 1 for VID1 or 2 for VID2
Physical address	0x4805 0630+ ((–1)* 0x04)	**Instance**	DISC
Description	This register configures the video FIFO. Shadow register, updated on VFP start period or EVSYNC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				PRELOAD											

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00000
11:0	PRELOAD	Video preload value: Number of bytes defining the preload value. Constraint: Maximum value is (FIFO size - DMA burst size - 8) bytes	RW	0x100

Table 12-258. Register Call Summary for Register DISPC_VIDn_PRELOAD

Display Subsystem Basic Programming Model
- Display Controller Basic Programming Model: [0]
- Video DMA Registers: [1] [2] [3] [4]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [5] [6]

12.7.2.3 RFBI Registers

Table 12-259. RFBI_REVISION

Address Offset	0x00		
Physical address	0x4805 0800	**Instance**	RFBI
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision	R	TI internal data

Table 12-260. Register Call Summary for Register RFBI_REVISION

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [0]

Table 12-261. RFBI_SYSCONFIG

Address Offset	0x10		
Physical address	0x4805 0810	**Instance**	RFBI
Description	This register allows control of various parameters of the interconnect interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								Reserved	Reserved	SIDLEMODE		Reserved	SOFTRESET	AUTOIDLE	

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000000
6	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
4:3	SIDLEMODE	Slave interface power management, Idle req/ack control 00: Force-idle: Idle request is acknowledged unconditionally. 01: No idle: An idle request is never acknowledged 10: Smart idle: Idle request is acknowledged based on the internal activity of the module. 11: Reserved	RW	0x0
2	Reserved	Write 0s for future compatibility Read returns 0	RW	0
1	SOFTRESET	Software reset Sets this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware. During reads, it always returns 0. 0: Normal mode 1: The module is reset	RW	0
0	AUTOIDLE	Internal clock gating strategy (interconnectL4 and display controller clock) 0: Interconnect L4 clock and display controller clock are free-running. 1: Automatic clock gating strategy is applied for the interconnect L4 clock and display controller clock, based on the interconnect interface and internal activity.	RW	1

Table 12-262. Register Call Summary for Register RFBI_SYSCONFIG

Display Subsystem Integration • Software Reset: [0] • Autoidle Mode: [1] • Idle Mode: [2] [3] [4]
Display Subsystem Basic Programming Model • RFBI Configuration: [5]
Display Subsystem Use Cases and Tips • Autoidle: [6] • Smart-Idle: [7]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [8]

Table 12-263. RFBI_SYSSTATUS

Address Offset	0x14		
Physical address	0x4805 0814	**Instance**	RFBI
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						BUSYRFBIDATA	BUSY	RESERVED							RESETDONE

Bits	Field Name	Description	Type	Reset
31: 10	Reserved	Reserved. Read returns 0	R	0x000000
9	BUSYRFBIDATA	Data are pending to be processed from interconnect FIFO. Read 0x0: No data pending Read 0x1: Some data are pending	R	0
8	BUSY	L4 Interface busy status bit Read 0x0: The access to the following register is not stalled: RFBI_CMD, RFBI_DATA, RFBI_STATUS, RFBI_PARAM, RFBI_READ. Read 0x1: The access to any of the following registers is stalled: RFBI_CMD, RFBI_DATA, RFBI_STATUS, RFBI_PARAM, RFBI_READ.	R	0
7:1	Reserved	Reserved. Read returns 0	R	0x00
0	RESETDONE	Internal reset monitoring 0: Internal module reset is on-going 1: Reset completed	R	1

Table 12-264. Register Call Summary for Register RFBI_SYSSTATUS

Display Subsystem Basic Programming Model
- RFBI Configuration: [0]
- RFBI State-Machine: [1] [2] [3] [4] [5] [6] [7] [8] [9]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [10]

Table 12-265. RFBI_CONTROL

Address Offset	0x40		
Physical address	0x4805 0840	**Instance**	RFBI
Description	The control register allows configuration of the RFBI module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							SMART_DMA_REQ	DISABLE_DMA_REQ	HIGHTHRESHOLD		ITE	CONFIGSELECT		BYPASSMODE	ENABLE

Bits	Field Name	Description	Type	Reset
31: 9	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0000000
8	SMART_DMA_REQ	Smart DMA request 0x0: The dmareq is asserted and de-asserted depending on the interconnect FIFO space even if MIdlereq is high in smart idle/no-idle mode and the entire burst gets error responses from the module. 0x1: The dmareq is de-asserted after 2 clk cycles if it has been asserted for more than or equal to 2 clk cycles and MIdlereq is high in smart idle or no idle mode. No more burst requests will be given even if the space is available in the interconnect FIFO.	RW	0x0

Bits	Field Name	Description	Type	Reset
7	DISABLE_DMA_REQ	Disable DMA request 0x0: The dmareq is enabled and the signal is generated based on the space available and the request coming into the data register. 0x1: The dmareq is disabled and the signal is not generated at all based on space in the interconnect FIFO. It stays high until the DISABLE DMAREQ is high even if there is space in the interconnect FIFO to take requests.	RW	0x0
6:5	HIGHTHRESHOLD	Defines the interconnect FIFO high threshold used by HW to assert DMA request. Used only if data written to RFBI_DATA are sent using system DMA. 0x0: Size of the transfer of 4 words of 32-bit wide 0x1: Size of the transfer of 8 words of 32-bit wide 0x2: Size of the transfer of 16 words of 32-bit wide	RW	0x0
4	ITE	Internal Trigger 0: H/W waits for ITE bit to be set if in internal trigger mode for the configuration in use. 1: User sets the ITE bit to start the transfer, when H/W takes into account the bit, the H/W resets it.	RW	0
3:2	CONFIGSELECT	Select the CS and configuration 00: No CS selected 01: CS0 selected and configuration #0 10: CS1 selected and configuration #1 11: CS0 and CS1 both selected (only the configuration for CS0 is used)	RW	0x0
1	BYPASSMODE	Bypass Mode 0: The bypass mode not selected 1: The bypass mode is selected	RW	1
0	ENABLE	Enable/Disable flag 0: Disable the RFBI module 1: Enable the RFBI module	RW	0

Table 12-266. Register Call Summary for Register RFBI_CONTROL

Display Subsystem Environment
- Parallel Interface in RFBI Mode (MIPI DBI Protocol): [0] [1]

Display Subsystem Basic Programming Model
- RFBI Control Registers: [2]
- High Threshold: [3] [4] [5]
- Bypass Mode: [6]
- Enable: [7] [8] [9]
- Configuration Selection: [10]
- ITE Bit: [11] [12] [13] [14] [15]
- Number of Pixels to Transfer: [16] [17]
- RFBI Configuration: [18]
- Trigger Mode: [19]
- RFBI Timings: [20]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [21]

Table 12-267. RFBI_PIXEL_CNT

Address Offset	0x44		
Physical address	0x4805 0844	**Instance**	RFBI
Description	The control register configures the RFBI pixel count value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIXELCNT																															

Bits	Field Name	Description	Type	Reset
31:0	PIXELCNT	Pixel counter value The S/W indicates the number of pixels to transfer to the LCD panel frame buffer. The value is set when the module is disabled. During the transfer the HW decrements the register when a pixel has been sent to the RFB.	RW	0x00000000

Table 12-268. Register Call Summary for Register RFBI_PIXEL_CNT

Display Subsystem Basic Programming Model • RFBI Control Registers: [0] • Enable: [1] • Number of Pixels to Transfer: [2] [3] [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

Table 12-269. RFBI_LINE_NUMBER

Address Offset	0x48		
Physical address	0x4805 0848	**Instance**	RFBI
Description	The control register configures the number of lines to synchronize the beginning of the transfer.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					LINENUMBER										

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Write 0s for future compatibility Read returns 0	RW	0x000000
10:0	LINENUMBER	Programmable line number Line number from 0 to 2^{11}-1. Number of HSYNC after the VSYNC occurs before the beginning of the transfer.	RW	0x000

Table 12-270. Register Call Summary for Register RFBI_LINE_NUMBER

Display Subsystem Basic Programming Model • RFBI Control Registers: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1]

Table 12-271. RFBI_CMD

Address Offset	0x4C		
Physical address	0x4805 084C	**Instance**	RFBI
Description	The control register configures the RFBI command		
Type	W		
Write Latency	1		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																CMD															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility Read returns 0	W	0x0000
15:0	CMD	Command Value 8/9/12/16 bit value depending on the parallel mode [7:0] 8-bit DT [8:0] 9-bit Data type [11:0] 12-bit Data type [15:0] 16-bit Data type	W	0x0000

Table 12-272. Register Call Summary for Register RFBI_CMD

Display Subsystem Functional Description • Send Commands: [0]
Display Subsystem Basic Programming Model • Number of Pixels to Transfer: [1] [2] • RFBI State-Machine: [3] [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5] • RFBI Registers: [6] [7]

Table 12-273. RFBI_PARAM

Address Offset	0x50		
Physical address	0x4805 0850	**Instance**	RFBI
Description	The control register configures the RFBI parameter.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PARAM															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility Read returns 0	W	0x0000
15:0	PARAM	Param Value 8/9/12/16 bit value depending on the parallel mode [7:0] 8-bit Data type [8:0] 9-bit Data type [11:0] 12-bit Data type [15:0] 16-bit Data type	W	0x0000

Table 12-274. Register Call Summary for Register RFBI_PARAM

Display Subsystem Basic Programming Model • Number of Pixels to Transfer: [0] [1] • RFBI State-Machine: [2] [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4] • RFBI Registers: [5] [6]

Table 12-275. RFBI_DATA

Address Offset	0x54		
Physical address	0x4805 0854	**Instance**	RFBI
Description	The control register configures the RFBI data.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data value 12/16/18/24/2x16 bit value depending on the Data type [11:0] 12-bit Data type [15:0] 16-bit Data type [17:0] 18-bit Data type [23:0] 24-bit Data type [31:0] 2x16-bit Data type	W	0x00000000

Table 12-276. Register Call Summary for Register RFBI_DATA

Display Subsystem Overview
• Display Subsystem Overview: [0]
Display Subsystem Functional Description
• RFBI Interconnect FIFO: [1] [2]
Display Subsystem Basic Programming Model
• High Threshold: [3] [4] [5]
• RFBI State-Machine: [6] [7] [8] [9] [10]
Display Subsystem Register Manual
• Display Subsystem Register Mapping Summary: [11]
• RFBI Registers: [12] [13] [14]

Table 12-277. RFBI_READ

Address Offset	0x58		
Physical address	0x4805 0858	**Instance**	RFBI
Description	The control register configures the RFBI read		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																READ															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0000
15:0	READ	Read Value 8/9/12/16 bit value depending on the parallel mode [7:0] 8-bit Data type [8:0] 9-bit Data type [11:0] 12-bit Data type [15:0] 16-bit Data type	RW	0x0000

Table 12-278. Register Call Summary for Register RFBI_READ

Display Subsystem Basic Programming Model
• Number of Pixels to Transfer: [0] [1]
• RFBI State-Machine: [2] [3]
Display Subsystem Register Manual
• Display Subsystem Register Mapping Summary: [4]
• RFBI Registers: [5] [6]

Table 12-279. RFBI_STATUS

Address Offset	0x5C		
Physical address	0x4805 085C	**Instance**	RFBI
Description	The control register configures the RFBI status.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																STATUS															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0000
15:0	STATUS	Status value 8/9/12/16 bit value depending on the parallel mode [7:0] 8-bit Data type [8:0] 9-bit Data type [11:0] 12-bit Data type [15:0] 16-bit Data type	RW	0x0000

Table 12-280. Register Call Summary for Register RFBI_STATUS

Display Subsystem Basic Programming Model

- Number of Pixels to Transfer: [0] [1]
- RFBI State-Machine: [2] [3]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [4]
- RFBI Registers: [5] [6]

Table 12-281. RFBI_CONFIGi

Address Offset	0x60+ (i* 0x18)	**Index**	i = 0 to 1
Physical address	0x4805 0860+ (i* 0x18)	**Instance**	RFBI
Description	The control register allows configuration #I of the RFBI module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										HSYNCPOLARITY	TE_VSYNC_POLARITY	CSPOLARITY	WEPOLARITY	REPOLARITY	A0POLARITY	Reserved			UNUSEDBITS		CYCLEFORMAT		L4FORMAT		DATA TYPE		TIMEGRANULARITY		TRIGGERMODE		PARALLEL MODE

Bits	Field Name	Description	Type	Reset
31:22	Reserved	Write 0s for future compatibility Read returns 0	RW	0x000
21	HSYNCPOLARITY	HSYNC polarity 0: HSYNC active low 1: HSYNC active high	RW	1
20	TE_VSYNC_ POLARITY	TE or VSYNC Polarity 0: TE or VSYNC active low 1: TE or SYNC active high	RW	1

Bits	Field Name	Description	Type	Reset
19	CSPOLARITY	CS Polarity 0: CS active low defined at reset time 1: CS active high defined at reset time	RW	0
18	WEPOLARITY	WE Polarity 0: WE active low 1: WE active high	RW	0
17	REPOLARITY	RE Polarity 0: RE active low 1: RE active high	RW	0
16	A0POLARITY	A0 Polarity 0: A0 active low 1: A0 active high	RW	1
15:13	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
12:11	UNUSEDBITS	State of unused bits 00: Low level (0) 01: High level (1) 10: Unchanged from previous state 11: Reserved	RW	0x0
10:9	CYCLEFORMAT	Cycle format 00: 1 cycle for 1 pixel 01: 2 cycles for 1 pixel 10: 3 cycles for 1 pixel 11: 3 cycles for 2 pixels	RW	0x0
8:7	L4FORMAT	L4 Write Access format 00: 1 pixel per L4 access to the register data 01: Reserved 10: 2 pixels per L4 access to the register data with 1st pixel at the position [15:0] 11: 2 pixels per L4 access to the register data with 1st pixel at the position [31:16]	RW	0x0
6:5	DATA TYPE	Data type from the display controller and L4 00: 12-bit 01: 16-bit 10: 18-bit 11: 24-bit	RW	0x0
4	TIMEGRANULARITY	Multiplies signal timing latencies by two 0: x2 latencies disabled 1: x2 latencies enabled	RW	0
3:2	TRIGGERMODE	Trigger Mode 00: Internal trigger mode (ITE bit mode) 01: External trigger mode (TE signal) 10: External trigger mode (VSYNC/HSYNC signals) 11: Reserved	RW	0x0
1:0	PARALLELMODE	Parallel Mode 00: 8-bit parallel output interface selected 01: 9-bit parallel output interface selected 10: 12-bit parallel output interface selected 11: 16-bit parallel output interface selected	RW	0x0

Table 12-282. Register Call Summary for Register RFBI_CONFIGi

Display Subsystem Environment • Parallel Interface in RFBI Mode (MIPI DBI Protocol): [0]
Display Subsystem Functional Description • Output Parallel Modes: [1] • Read/Write: [2] [3]

Table 12-282. Register Call Summary for Register RFBI_CONFIGi (continued)

Display Subsystem Basic Programming Model • ITE Bit: [4] [5] • Number of Pixels to Transfer: [6] [7] [8] [9] [10] • Parallel Mode: [11] • Cycle Format: [12] • Unused Bits: [13] • RFBI Timings: [14] • RFBI State-Machine: [15] [16] • RFBI Configuration Flow Charts: [17] [18] [19]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [20]

Table 12-283. RFBI_ONOFF_TIMEi

Address Offset	0x64+ (i* 0x18)	**Index**	i = 0 to 1
Physical address	0x4805 0864+ (i* 0x18)	**Instance**	RFBI
Description	The control register allows configuration of the RFBI timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		REOFFTIME						REONTIME				WEOFFTIME				WEONTIME				CSOFFTIME								CSONTIME			

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
29:24	REOFFTIME	Read Enable deassertion time from start access time Number of L4Clk cycles	RW	0x00
23:20	REONTIME	Read Enable assertion time from start access time Number of L4Clk cycles	RW	0x0
19:14	WEOFFTIME	Write Enable deassertion time from start access time Number of L4Clk cycles	RW	0x00
13:10	WEONTIME	Write Enable assertion time from start access time Number of L4Clk cycles	RW	0x0
9:4	CSOFFTIME	CS deassertion time from start access time Number of L4Clk cycles	RW	0x00
3:0	CSONTIME	CS assertion time from start access time Number of L4Clk cycles	RW	0x0

Table 12-284. Register Call Summary for Register RFBI_ONOFF_TIMEi

Display Subsystem Environment • Transaction Timing Diagrams: [0] [1] [2]
Display Subsystem Basic Programming Model • RFBI Timings: [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [18]

Table 12-285. RFBI_CYCLE_TIMEi

Address Offset	0x68+ (i* 0x18)	**Index**	i = 0 to 1
Physical address	0x4805 0868+ (i* 0x18)	**Instance**	RFBI
Description	The control register allows configuration of the RFBI timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ACCESSTIME						WRENABLE	WWENABLE	RRENABLE	RWENABLE	CSPULSEWIDTH						RECYCLETIME				WECYCLETIME							

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
27:22	ACCESSTIME	Access Time Number of L4Clk cycles	RW	0x00
21	WRENABLE	Write to Read Pulse Width Enable (same CS) 0: CSPulseWidth does not apply on Write to Read access 1: CSPulseWidth applies on Write to Read access	RW	0
20	WWENABLE	Write to Write Pulse Width Enable (same CS) 0: CSPulseWidth does not apply on Write to Write access 1: CSPulseWidth applies on Write to Write access	RW	0
19	RRENABLE	Read to Read Pulse Width Enable (same CS) 0: CSPulseWidth does not apply on Read to Read access 1: CSPulseWidth applies on Read to Read access	RW	0
18	RWENABLE	Read to Write Pulse Width Enable (same CS) 0: CSPulseWidth does not apply on Read to Write access 1: CSPulseWidth applies on Read to Write access	RW	0
17:12	CSPULSEWIDTH	CS Pulse Width Number of L4Clk cycles	RW	0x00
11:6	RECYCLETIME	RE Cycle Time Number of L4Clk cycles	RW	0x00
5:0	WECYCLETIME	WE Cycle Time Number of L4Clk cycles	RW	0x00

Table 12-286. Register Call Summary for Register RFBI_CYCLE_TIMEi

Display Subsystem Environment
- Transaction Timing Diagrams: [0] [1] [2]

Display Subsystem Basic Programming Model
- RFBI Timings: [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [16]

Table 12-287. RFBI_DATA_CYCLE1_i

Address Offset	0x6C+ (i* 0x18)	**Index**	i = 0 to 1
Physical address	0x4805 086C+ (i* 0x18)	**Instance**	RFBI
Description	The control register configures the RFBI data format for 1st cycle.		
Type	RW		

31 30 29 28	27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	BITALIGNMENTPIXEL2	Reserved / NBBITSPIXEL2	Reserved	BITALIGNMENTPIXEL1	Reserved / NBBITSPIXEL1

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
27:24	BITALIGNMENTPIXEL2	Bit alignment Alignment of the bits from pixel#2 on the output interface	RW	0x0
23:21	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
20:16	NBBITSPIXEL2	Number of bits Number of bits from the pixel #2 (value from 0 to16 bits). The values from 17 to 31 are invalid.	RW	0x00
15:12	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
11:8	BITALIGNMENTPIXEL1	Bit alignment Alignment of the bits from pixel#1 on the output interface	RW	0x0
7:5	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
4:0	NBBITSPIXEL1	Number of bits Number of bits from the pixel #1 (value from 0 to16 bits). The values from 17 to 31 are invalid.	RW	0x00

Table 12-288. Register Call Summary for Register RFBI_DATA_CYCLE1_i

Display Subsystem Functional Description • Output Parallel Modes: [0]
Display Subsystem Basic Programming Model • Cycle Format: [1] [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

Table 12-289. RFBI_DATA_CYCLE2_i

Address Offset	0x70+ (i* 0x18)	**Index**	i = 0 to 1
Physical address	0x4805 0870+ (i* 0x18)	**Instance**	RFBI
Description	The control register configures the RFBI data format for 2nd cycle.		
Type	RW		

31 30 29 28	27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Reserved	BITALIGNMENTPIXEL2	Reserved / NBBITSPIXEL2	Reserved	BITALIGNMENTPIXEL1	Reserved / NBBITSPIXEL1

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
27:24	BITALIGNMENTPIXEL2	Bit alignment Alignment of the bits from pixel#2 on the output interface	RW	0x0
23:21	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
20:16	NBBITSPIXEL2	Number of bits Number of bits from the pixel #2 (value from 0 to16 bits). The values from 17 to 31 are invalid.	RW	0x00
15:12	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
11:8	BITALIGNMENTPIXEL1	Bit alignment Alignment of the bits from pixel#1 on the output interface	RW	0x0
7:5	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
4:0	NBBITSPIXEL1	Number of bits Number of bits from the pixel #1 (value from 0 to16 bits). The values from 17 to 31 are invalid.	RW	0x00

Table 12-290. Register Call Summary for Register RFBI_DATA_CYCLE2_i

Display Subsystem Functional Description • Output Parallel Modes: [0]
Display Subsystem Basic Programming Model • Cycle Format: [1] [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

Table 12-291. RFBI_DATA_CYCLE3_i

Address Offset	0x74+ (i* 0x18)	**Index**	i = 0 to 1
Physical address	0x4805 0874+ (i* 0x18)	**Instance**	RFBI
Description	The control register configures the RFBI data format for 3rd cycle.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				BITALIGNMENTPIXEL2				Reserved			NBBITSPIXEL2					Reserved				BITALIGNMENTPIXEL1				Reserved			NBBITSPIXEL1				

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
27:24	BITALIGNMENTPIXEL2	Bit alignment Alignment of the bits from pixel#2 on the output interface	RW	0x0
23:21	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
20:16	NBBITSPIXEL2	Number of bits Number of bits from the pixel #2 (value from 0 to16 bits). The values from 17 to 31 are invalid.	RW	0x00
15:12	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0

Bits	Field Name	Description	Type	Reset
11:8	BITALIGNMENTPIXEL1	Bit alignment Alignment of the bits from pixel#1 on the output interface	RW	0x0
7:5	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0
4:0	NBBITSPIXEL1	Number of bits Number of bits from the pixel #1 (value from 0 to16 bits). The values from 17 to 31 are invalid.	RW	0x00

Table 12-292. Register Call Summary for Register RFBI_DATA_CYCLE3_i

Display Subsystem Functional Description • Output Parallel Modes: [0]
Display Subsystem Basic Programming Model • Cycle Format: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-293. RFBI_VSYNC_WIDTH

Address Offset	0x90		
Physical address	0x4805 0890	**Instance**	RFBI
Description	The control register configures the RFBI VSYNC minimum pulse width		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MINVSYNCPULSEWIDTH															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0000
15:0	MINVSYNCPULSEWIDTH	Programmable min VSYNC pulse width Minimum VSYNC pulse width from 0 to 65535. Number of L4 clock cycles to determine when VSYNC pulse occurs. The values 0 and 1 are invalid.	RW	0x0000

Table 12-294. Register Call Summary for Register RFBI_VSYNC_WIDTH

Display Subsystem Environment • Description of the TE Pulse Signal: [0]
Display Subsystem Basic Programming Model • RFBI Configuration: [1] • VSYNC Pulse Width (Minimum Value): [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

Table 12-295. RFBI_HSYNC_WIDTH

Address Offset	0x94		
Physical address	0x4805 0894	**Instance**	RFBI
Description	The control register configures the RFBI HSYNC minimum pulse width.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MINHSYNCPULSEWIDTH															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility Read returns 0	RW	0x0000
15:0	MINHSYNC PULSEWIDTH	Programmable min HSYNC pulse width Minimum HSYNC pulse width from 0 to 65535. Number of L4 clock cycles to determine when HSYNC pulse occurs. The values 0 and 1 are invalid.	RW	0x0000

Table 12-296. Register Call Summary for Register RFBI_HSYNC_WIDTH

Display Subsystem Environment • Description of the TE Pulse Signal: [0]
Display Subsystem Basic Programming Model • RFBI Configuration: [1] • HSYNC Pulse Width (Minimum Value): [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

12.7.2.4 Video Encoder Registers

Table 12-297. VENC_REV_ID

Address Offset	0x00		
Physical address	0x4805 0C00	**Instance**	VENC
Description	Revision ID for the encoder		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV_ID							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved. Read returns 0s.	R	0x000000
7:0	REV_ID	This read-only register contains the revision ID for the encoder. The revision ID will identify different revisions of the IP.	R	TI internal data

Table 12-298. Register Call Summary for Register VENC_REV_ID

Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [0]

Table 12-299. VENC_STATUS

Address Offset	0x04		
Physical address	0x4805 0C04	**Instance**	VENC
Description	VENC_STATUS		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											CCE	CCO	FSQ			

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reserved. Read returns 0s.	R	0x0000000
4	CCE	Closed caption status for even Field. This bit is set immediately after the data in registers LINE21_E0 and LINE21_E1 have been encoded to closed caption. This bit is reset when both of these registers are written.	R	0
3	CCO	Closed Caption Status for Odd Field. This bit is set immediately after the data in registers LINE21_O0 and LINE21_O1 have been encoded to closed caption. This bit is reset when both of these registers are written.	R	0
2:0	FSQ	Field Sequence ID. For PAL, all three FSQ[2:0] are used whereas for NTSC only FSQ[1:0] is meaningful. Furthermore, FSQ[0] represents odd field when it is 0 and even field when it is 1. Read 0x0: Odd field Read 0x1: Even field	R	0x0

Table 12-300. Register Call Summary for Register VENC_STATUS

Display Subsystem Functional Description
- Closed Caption Encoding: [0] [1] [2] [3]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [4]
- Video Encoder Registers: [5]

Table 12-301. VENC_F_CONTROL

Address Offset	0x08		
Physical address	0x4805 0C08	**Instance**	VENC
Description	This register specifies the input video source and format		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							RESET	SVDS		RGBF	BCOLOR			FMT	

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Reserved. Read returns 0s.	RW	0x000000
8	RESET	RESET the encoder 0x0: No effect 0x1: Reset the encoder, after reset, this bit is automatically set to zero.	RW	0
7:6	SVDS	Select Video Data Source. 0x0: Use external video source 0x1: Use internal Color BAR 0x2: Use background color 0x3: Reserved	RW	0x2
5	RGBF	RGB/YCrCb input coding range 0x0: The input RGB data are in binary format with coding range 0-255 The input YCrCb data are in binary format with coding range 0-255 0x1: The input RGB data are in binary format with coding range 16-235 The input YCrCb data are in binary format conforming to ITU-601 standard	RW	0

Bits	Field Name	Description	Type	Reset
4:2	BCOLOR	Background color select 0x0: black 0x1: blue 0x2: red 0x3: magenta 0x4: green 0x5: cyan 0x6: yellow 0x7: white	RW	0x1
1:0	FMT	These two bits specify the video input data stream format and timing 0x0: 24-bit 4:4:4 RGB 0x1: 24-bit 4:4:4 0x2: 16-bit 4:2:2 0x3: 8-bit ITU-R 656 4:2:2	RW	0x3

Table 12-302. Register Call Summary for Register VENC_F_CONTROL

Display Subsystem Functional Description • Test Pattern Generation: [0]
Display Subsystem Basic Programming Model • Video Encoder Software Reset: [1] • Video Encoder Programming Sequence: [2] [3] • Video Encoder Register Settings: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

Table 12-303. VENC_VIDOUT_CTRL

Address Offset	0x10		
Physical address	0x4805 0C10	**Instance**	VENC
Description	Encoder output clock		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															27_54

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved. Read returns 0s.	RW	0x00000000
0	27_54	Encoder output clock 0x0: 54 MHz, 4x oversampling 0x1: 27 MHz, 2x oversampling, the last 2x oversampling filter bypassed	RW	0

Table 12-304. Register Call Summary for Register VENC_VIDOUT_CTRL

Display Subsystem Basic Programming Model • Video Encoder Register Settings: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1]

Table 12-305. VENC_SYNC_CTRL

Address Offset	0x14		
Physical address	0x4805 0C14	**Instance**	VENC
Description	Sync Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FREE	ESAV	IGNP	NBLNKS	VBLKM		HBLKM		Reserved	FID_POL	Reserved					

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Reserved. Read returns 0s.	RW	0x0000
15	FREE	Free running	RW	1
		0x0: Free running disabled		
		0x1: Free running enabled. HSYNC and VSYNC are ignored		
14	ESAV	Enable to detect F and V bits only on EAV in ITU-R 656 input mode	RW	0
		0x0: Detection of F and V bits on both EAV and SAV		
		0x1: Detection of F and V bits only on EAV		
13	IGNP	Ignore protection bits in ITU-R 656 input mode	RW	0
		0x0: Protection bits are not ignored		
		0x1: Protection bits are ignored		
12	NBLNKS	Blank shaping	RW	0
		0x0: Blank shaping enabled		
		0x1: Blank shaping disabled		
11:10	VBLKM	Vertical blanking mode	RW	0x0
		0x0: Internal default blanking		
		0x1: Internal default blanking AND internal programmable blanking defined by VENC_FLEN_FAL[24:16]FAL and VENC_LAL_PHASE_RESET[8:0] LAL bit fields. Note: in this mode, the VENC_LAL_PHASE_RESET[16] SBLANK bit must be '0b1' to active the VBLKM functionality.		
		0x2: Reserved		
		0x3: Reserved		
9:8	HBLKM	Horizontal blanking mode	RW	0x0
		0x0: Internal default blanking		
		0x1: Internal programmable blanking defined by VENC_SAVID_EAVID[26:16] SAVID and VENC_SAVID_EAVID[10:0] EAVID bit fields.		
		0x2: External blanking defined by VENC_AVID_START_STOP_X and VENC_AVID_START_STOP_Y registers.		
		0x3: Reserved		
7	Reserved	Reserved. Read returns 0.	RW	0
6	FID_POL	FID output polarity	RW	0
		0x0: Odd field = 0 Even field = 1		
		0x1: Odd field = 1 Even field = 0		
5:0	Reserved	Reserved. Read returns 0.	RW	0x00

Table 12-306. Register Call Summary for Register VENC_SYNC_CTRL

Display Subsystem Basic Programming Model • Video Encoder Programming Sequence: [0] • Video Encoder Register Settings: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2] • Video Encoder Registers: [3]

Table 12-307. VENC_LLEN

Address Offset	0x1C		
Physical address	0x4805 0C1C	**Instance**	VENC
Description	VENC_LLEN		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved					LLEN										

Bits	Field Name	Description	Type	Reset
31:15	Reserved	Reserved. Read returns 0s.	RW	0x0000
14:11	Reserved	Reserved. Read returns 0s.	RW	0x0
10:0	LLEN	LLEN[10:0] Line length or total number of pixels in a scan line including active video and blanking. Total number of pixels in a scan line = LLEN **NOTE:** A write on the LLEN[10] is illegal.	RW	0x359

Table 12-308. Register Call Summary for Register VENC_LLEN

Display Subsystem Basic Programming Model • Video Encoder Register Settings: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1] • Video Encoder Registers: [2] [3]

Table 12-309. VENC_FLENS

Address Offset	0x20		
Physical address	0x4805 0C20	**Instance**	VENC
Description	VENC_FLENS		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					FLENS										

Bits	Field Name	Description	Type	Reset
31:11	Reserved	Reserved. Read returns 0s.	RW	0x000000
10:0	FLENS	The frame length or total number of lines in a frame including active video and blanking from the source image. Total number of lines in a frame from the source image = FLENS + 1	RW	0x20C

Table 12-310. Register Call Summary for Register VENC_FLENS

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2]

Table 12-311. VENC_HFLTR_CTRL

Address Offset	0x24		
Physical address	0x4805 0C24	**Instance**	VENC
Description	VENC_HFLTR_CTRL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													CINTP		YINTP

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reserved. Read returns 0s.	RW	0x00000000
2:1	CINTP	Chrominance interpolation filter control	RW	0x0
		0x0: The chrominance interpolation filter is enabled		
		0x1: The first section of the chrominance interpolation filter is bypassed		
		0x2: The second section of the chrominance interpolation filter is bypassed		
		0x3: Both sections of the filter are bypassed		
0	YINTP	Luminance interpolation filter control	RW	0
		0x0: The luminance interpolation filter is enabled		
		0x1: The luminance interpolation filter is bypassed		

Table 12-312. Register Call Summary for Register VENC_HFLTR_CTRL

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2]

Table 12-313. VENC_CC_CARR_WSS_CARR

Address Offset	0x28		
Physical address	0x4805 0C28	**Instance**	VENC
Description	Frequency code control		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FWSS																FCC															

Bits	Field Name	Description	Type	Reset
31:16	FWSS	Wide screen signaling run-in code frequency control For common values for FWSS[15:0] bit field, refer to Table 12-44 Reset value is for NTSC-601standard	RW	0x043F
15:0	FCC	Close caption run-in code frequency control For common values for FCC[15:0] bit field refer to Table 12-42. Reset value is for NTSC-601 standard	RW	0x2631

Table 12-314. Register Call Summary for Register VENC_CC_CARR_WSS_CARR

Display Subsystem Functional Description • Closed Caption Encoding: [0] [1] [2] • Wide-Screen Signaling (WSS) Encoding: [3] [4]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [5]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [6]

Table 12-315. VENC_C_PHASE

Address Offset	0x2C		
Physical address	0x4805 0C2C	**Instance**	VENC
Description	VENC_C_PHASE		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CPHS							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved. Read returns 0.	RW	0x000000
7:0	CPHS	Phase of the encoded video color subcarrier (including the color burst) relative to H-sync. The adjustable step is 360/256 degrees.	RW	0x00

Table 12-316. Register Call Summary for Register VENC_C_PHASE

Display Subsystem Functional Description • Subcarrier and Burst Generation: [0] [1] [2]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4] • Video Encoder Registers: [5]

Table 12-317. VENC_GAIN_U

Ad**dress Offset**	0x30		
Physical address	0x4805 0C30	**Instance**	VENC
Description	Gain control for Cb signal		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								GU							

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Reserved. Read returns 0s.	RW	0x000000
8:0	GU	Gain control for Cb signal. Following are typical programming examples for NTSC and PAL standards. NTSC with 7.5 IRE pedestal: WHITE - BLACK = 92.5 IRE GU = 0x102 NTSC with no pedestal: WHITE - BLACK = 100 IRE GU = 0x117 PAL with no pedestal: WHITE - BLACK = 100 IRE GU = 0x111	RW	0x102

Table 12-318. Register Call Summary for Register VENC_GAIN_U

Display Subsystem Functional Description • Chroma Stage: [0]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-319. VENC_GAIN_V

Address Offset	0x34		
Physical address	0x4805 0C34	**Instance**	VENC
Description	Gain control of Cr signal		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								GV							

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Reserved. Read returns 0s.	RW	0x000000
8:0	GV	Gain control of Cr signal. Following are typical programming examples for NTSC and PAL standards. NTSC with 7.5 IRE pedestal: WHITE - BLACK = 92.5 IRE GV = 0x16C NTSC with no pedestal: WHITE - BLACK = 100 IRE GV = 0x189 PAL with no pedestal: WHITE - BLACK = 100 IRE GV = 0x181	RW	0x16C

Table 12-320. Register Call Summary for Register VENC_GAIN_V

Display Subsystem Functional Description • Chroma Stage: [0]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-321. VENC_GAIN_Y

Address Offset	0x38		
Physical address	0x4805 0C38	**Instance**	VENC
Description	Gain control of Y signal		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								GY							

Bits	Field Name	Description	Type	Reset
31:9	Reserved	Reserved. Read returns 0s.	RW	0x000000
8:0	GY	Gain control of Y signal. Following are typical programming examples for NTSC/PAL standards. NTSC with 7.5 IRE pedestal: WHITE - BLACK = 92.5 IRE GY = 0x12F NTSC with no pedestal: WHITE - BLACK = 100 IRE GY = 0x147 PAL with no pedestal: WHITE - BLACK = 100 IRE GY = 0x140	RW	0x12F

Table 12-322. Register Call Summary for Register VENC_GAIN_Y

Display Subsystem Functional Description • Luma Stage: [0]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-323. VENC_BLACK_LEVEL

Address Offset	0x3C		
Physical address	0x4805 0C3C	**Instance**	VENC
Description	Video Encoder BLACK LEVEL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BLACK							

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Reserved. Read returns 0.	RW	0x0000000
6:0	BLACK	Black level setting. Following are typical programming examples for NTSC/PAL standards. NTSC with 7.5 IRE pedestal: WHITE - BLACK = 92.5 IRE BLACK_LEVEL = 0x43 NTSC with no pedestal: WHITE - BLACK = 100 IRE BLACK_LEVEL = 0x38 PAL with no pedestal: WHITE - BLACK = 100 IRE BLACK_LEVEL = 0x3B	RW	0x43

Table 12-324. Register Call Summary for Register VENC_BLACK_LEVEL

Display Subsystem Functional Description • Luma Stage: [0]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-325. VENC_BLANK_LEVEL

Address Offset	0x40		
Physical address	0x4805 0C40	**Instance**	VENC
Description	Video Encoder BLANK LEVEL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								BLANK							

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Reserved. Read returns 0s.	RW	0x0000000
6:0	BLANK	Blank level setting. Following are typical programming examples for NTSC/PAL standards. NTSC with 7.5 IRE pedestal: WHITE - BLACK = 92.5 IRE BLANK_LEVEL = 0x38 NTSC with no pedestal: WHITE - BLACK = 100 IRE BLANK_LEVEL = 0x38 PAL with no pedestal: WHITE - BLACK = 100 IRE BLANK_LEVEL = 0x3B	RW	0x38

Table 12-326. Register Call Summary for Register VENC_BLANK_LEVEL

Display Subsystem Functional Description • Luma Stage: [0]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-327. VENC_X_COLOR

Address Offset	0x44		
Physical address	0x4805 0C44	**Instance**	VENC
Description	Cross-Color Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									XCE	Reserved		XCBW		LCD	

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Reserved. Read returns 0s.	RW	0x0000000
6	XCE	Cross color reduction enable for composite video output. Cross color does not affect S-video output 0x0: Cross color reduction is disabled 0x1: Cross color is enabled	RW	0
5	Reserved	Reserved. Read returns 0.	RW	0
4:3	XCBW	Cross color reduction filter selection 0x0: The notch is at 32.8 % of the frequency of the encoding pixel clock 0x1: The notch is at 26.5 % of the frequency of the encoding pixel clock 0x2: The notch is at 30.0 % of the frequency of the encoding pixel clock 0x3: The notch is at 29.2 % of the frequency of the encoding pixel clock	RW	0x0
2:0	LCD	These three bits can be used for chroma channel delay compensation. Delay on Luma channel. 0x0: 0 0x1: 0.5 pixel clock period 0x2: 1.0 pixel clock period 0x3: 1.5 pixel clock period 0x4: -2.0 pixel clock period 0x5: -1.5 pixel clock period 0x6: -1.0 pixel clock period 0x7: -0.5 pixel clock period	RW	0x0

Table 12-328. Register Call Summary for Register VENC_X_COLOR

Display Subsystem Basic Programming Model • Video Encoder Register Settings: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1]

Table 12-329. VENC_M_CONTROL

Address Offset	0x48		
Physical address	0x4805 0C48	**Instance**	VENC
Description	VENC_M_CONTROL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PALI	PALN	PALPHS	CBW			PAL	FFRQ

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved. Read returns 0s.	RW	0x0000000
7	PALI	PAL I enable 0x0: Normal operation 0x1: PAL I enable	RW	0
6	PALN	PAL N enable 0x0: Normal operation 0x1: PAL N enable	RW	0
5	PALPHS	PAL switch phase setting 0x0: PAL switch phase is nominal 0x1: PAL switch phase is inverted compared to nominal	RW	0
4:2	CBW	Chrominance lowpass filter bandwidth control 0x0: -6db at 21.8 % of encoding pixel clock frequency 0x1: -6db at 19.8 % of encoding pixel clock frequency 0x2: -6db at 18.0 % of encoding pixel clock frequency 0x3: Reserved 0x4: Reserved 0x5: -6db at 23.7 % of encoding pixel clock frequency 0x6: -6db at 26.8 % of encoding pixel clock frequency 0x7: Chrominance lowpass filter bypass	RW	0x0
1	PAL	Phase alternation line encoding selection 0x0: Phase alternation line encoding disabled 0x1: Phase alternation line encoding enabled	RW	0
0	FFRQ	The value of this field and the SQP bit in the VENC_BSTAMP_WSS_DATA[7] SQP bit control the number of horizontal pixels displayed per scan line Mode: Configuration: ITU-R 601 NTSC: SQP = 0, FFRQ = 1, Number of pixels by line = 858 Square pixel NTSC: SQP = 1, FFRQ = 1, Number of pixels by line = 780 ITU-R 601 PAL: SQP = 0, FFRQ = 0, Number of pixels by line = 864 Square pixel PAL: SQP = 1, FFRQ = 0, Number of pixels by line = 944	RW	1

Table 12-330. Register Call Summary for Register VENC_M_CONTROL

Display Subsystem Functional Description • Subcarrier and Burst Generation: [0] [1] [2]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4] • Video Encoder Registers: [5] [6]

Table 12-331. VENC_BSTAMP_WSS_DATA

Address Offset	0x4C		
Physical address	0x4805 0C4C	**Instance**	VENC
Description	VENC BSTAMP and WSS_DATA		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WSS_DATA																SQP	BSTAP						

Bits	Field Name	Description			Type	Reset
31:28	Reserved	Reserved. Read returns 0s.			RW	0x0
27:8	WSS_DATA	WSS data [19:0]: Wide Screen Signaling data			RW	0x00000
		NTSC:	WORD 0	WSS_D1, WSS_D0		
			WORD 1	WSS_D5, WSS_D4, WSS_D3, WSS_D2		
			WORD 2	WSS_D13, WSS_D12, WSS_D11, WSS_D10, WSS_D9, WSS_D8, WSS_D7, WSS_D6		
			CRC	WSS_D19, WSS_D18, WSS_D17, WSS_D16, WSS_D15, WSS_D14		
		PAL:	GROUP A	WSS_D3, WSS_D2, WSS_D1, WSS_D0		
			GROUP B	WSS_D7, WSS_D6, WSS_D5, WSS_D4		
			GROUP C	WSS_D10, WSS_D9, WSS_D8		
			GROUP D	WSS_D13, WSS_D12, WSS_D11		
7	SQP	Square-pixel sampling rate. Please refer to VENC_M_CONTROL[0] FFRQ bit description for programming information.			RW	0
		0x0: ITU-R 601 sampling rate				
		0x1: Square-pixel sampling rate				
6:0	BSTAP	Setting of amplitude of color burst.			RW	0x38

Table 12-332. Register Call Summary for Register VENC_BSTAMP_WSS_DATA

Display Subsystem Functional Description • Subcarrier and Burst Generation: [0] • Wide-Screen Signaling (WSS) Encoding: [1]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3] • Video Encoder Registers: [4]

Table 12-333. VENC_S_CARR

Address Offset	0x50		
Physical address	0x4805 0C50	**Instance**	VENC
Description	Color Subcarrier Frequency Registers.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSC																															

Bits	Field Name	Description	Type	Reset
31:0	FSC	These four bytes' data program the color subcarrier frequency and are determined by the following formula. S_CARR = ROUND((Fsc/Fclkenc) * 2^{32}) Where: Fsc = Frequency of the subcarrier Fclkenc = Frequency of the internal video encoding clock = 2*LLEN *Fh LLEN = Number of pixels in a scan line. For LLEN setting, please refer to the description of VENC_LLEN register (offset 0x1C). Fh = Line frequency For typical setting of the FSC field, refer to Table 12-40.	RW	0x21F07C1F

Table 12-334. Register Call Summary for Register VENC_S_CARR

Display Subsystem Functional Description • Subcarrier and Burst Generation: [0] [1] [2]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-335. VENC_LINE21

Address Offset	0x54		
Physical address	0x4805 0C54	**Instance**	VENC
Description	VENC LINE 21		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L21E																L21O															

Bits	Field Name	Description	Type	Reset
31:16	L21E	The two bytes of the closed caption data in the even field. For a complete field description, refer to CEA-608-x standard. For data stream content, see Table 12-41, *Closed-Caption Data Format.* [31:24] First byte of data [23:16] Second byte of data	RW	0x0000
15:0	L21O	The two bytes of the closed caption data in the odd field. For a complete field description, refer to CEA-608-x standard. For data stream content, see Table 12-41, *Closed-Caption Data Format.* [15:8] First byte of data [7:0] Second byte of data	RW	0x0000

Table 12-336. Register Call Summary for Register VENC_LINE21

Display Subsystem Functional Description • Closed Caption Encoding: [0] [1] [2] [3]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

Table 12-337. VENC_LN_SEL

Address Offset	0x58		
Physical address	0x4805 0C58	**Instance**	VENC
Description	VENC_LN_SEL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LN21_RUNIN								Reserved								SLINE							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	LN21_RUNIN	The two bytes of the closed caption run in code position from the HSYNC.	RW	0x10B
15:5	Reserved	Reserved. Read returns 0s.	RW	0x000
4:0	SLINE	Selects the line where closed caption or extended service data are encoded. The value of the SLINE[4:0] bit field depends on the video standard: PAL mode: Because there is a one-line offset, program the desired line number – 1. To activate the closed caption on line 21 (0x15), program the value 0x15 – 1 = 0x14. The default value is 0x15 + 1 = 0x16 (line 22). NTSC mode: Because there is a four-line offset, program the desired line number – 4. To activate the closed caption on line 21 (0x15), program the value 0x15 – 4 = 0x11. The default value is 0x15 + 4 = 0x19 (line 25).	RW	0x15

Table 12-338. Register Call Summary for Register VENC_LN_SEL

Display Subsystem Functional Description • Closed Caption Encoding: [0] [1] [2]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4] • Video Encoder Registers: [5]

Table 12-339. VENC_L21_WC_CTL

Address Offset	0x5C		
Physical address	0x4805 0C5C	**Instance**	VENC
Description	VENC L21 & WC_CTL registers		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																INV	EVEN_ODD_EN		LINE					Reserved							L21EN

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Reserved. Read returns 0s.	RW	0x0000
15	INV	WSS inverter	RW	0
		0x0: No effect		
		0x1: Invert WSS data		
14:13	EVEN_ODD_EN	This bit controls the WSS encoding.	RW	0x0
		0x0: WSS encoding OFF		
		0x1: Enables encoding in 1st field (odd field)		
		0x2: Enables encoding in 2nd field (even field)		
		0x3: Enables encoding in both fields		
12:8	LINE	Selects the line where WSS data are encoded. The LINE[12:8] bit field value depends on the video standard: PAL mode: There is an one line offset, so program the wanted line number - 1. The recommended value is line 0x16 + 1 = 0x17 (23rd line). The default value is 0x14 + 1 = 0x15 (line 21). NTSC mode: There is a four line offset, so program the wanted line number - 4. The recommended value is line 0x10 + 4 = 0x14 (20th line). The default value is 0x14 + 4 = 0x18 (line 24).	RW	0x14
7:2	Reserved	Reserved. Read returns 0s.	RW	0x00
1:0	L21EN	Those bits controls the Line21 closed caption encoding according to the mode.	RW	0x0
		0x0: Line21 encoding OFF		
		0x1: Enables encoding in 1st field (odd field)		
		0x2: Enables encoding in 2d field (even field)		
		0x3: Enables encoding in both fields		

Table 12-340. Register Call Summary for Register VENC_L21_WC_CTL

Display Subsystem Functional Description • Closed Caption Encoding: [0] [1] • Wide-Screen Signaling (WSS) Encoding: [2] [3]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

Table 12-341. VENC_HTRIGGER_VTRIGGER

Address Offset	0x60		
Physical address	0x4805 0C60	**Instance**	VENC
Description	VENC HTRIGGER and VTRIGGER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VTRIG								Reserved								HTRIG							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	VTRIG	Vertical trigger reference for VSYNC. These bits specify the phase between VSYNC input and the lines in a field. The VTRIG field is expressed in units of half-line.	RW	0x000
15:11	Reserved	Reserved. Read returns 0s.	RW	0x00
10:0	HTRIG	Horizontal trigger phase, which sets HSYNC. HTRIG is expressed in half-pixels or clk2x (27 MHz) periods	RW	0x000

Table 12-342. Register Call Summary for Register VENC_HTRIGGER_VTRIGGER

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-343. VENC_SAVID_EAVID

Address Offset	0x64		
Physical address	0x4805 0C64	**Instance**	VENC
Description	VENC SAVID and EAVID		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EAVID								Reserved								SAVID							

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved. Read returns 0s.	RW	0x00
26:16	EAVID	End of active video. These bits define the ending pixel position on a horizontal display line where active video will be displayed.	RW	0x693
15:11	Reserved	Reserved. Read returns 0s.	RW	0x00
10:0	SAVID	Start of active video. These bits define the starting pixel position on a horizontal line where active video will be displayed.	RW	0x0F4

Table 12-344. Register Call Summary for Register VENC_SAVID_EAVID

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2] [3]

Table 12-345. VENC_FLEN_FAL

Address Offset	0x68		
Physical address	0x4805 0C68	**Instance**	VENC
Description	VENC FLEN and FAL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FAL								Reserved								FLEN							

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Reserved. Read returns 0s.	RW	0x00
24:16	FAL	First Active Line of Field. These bits define the first active line of a field	RW	0x016
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	FLEN	Field length. These bits define the number of half_lines in each field. Length of field = (FLEN + 1) half_lines	RW	0x20C

Table 12-346. Register Call Summary for Register VENC_FLEN_FAL

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2]

Table 12-347. VENC_LAL_PHASE_RESET

Address Offset	0x6C		
Physical address	0x4805 0C6C	**Instance**	VENC
Description	VENC LAL and PHASE_RESET		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PRES		SBLANK	Reserved								LAL							

Bits	Field Name	Description	Type	Reset
31:19	Reserved	Reserved. Read returns 0s.	RW	0x0000
18:17	PRES	Phase reset mode. 0x0: No reset 0x1: Reset every two lines 0x2: Reset every eight fields. Color subcarrier phase is reset to VENC_CPHASE[7:0] CPHS field value (offset 0x2C) upon reset 0x3: Reset every four fields. Color subcarrier phase is reset to VENC_CPHASE[7:0] CPHS bit field value (offset 0x2C) upon reset	RW	0x3
16	SBLANK	Data output enable 0x0: No functionality 0x1: Enables the output of data when VENC_SYNC_CTRL[10] VBLMK bit is '0b1'.	RW	0
15:9	Reserved	Reserved. Read returns 0s.	RW	0x00
8:0	LAL	Last Active Line of Field. These bits define the last active line of a field. The LAL[8:0] bit field value must be set to a value lower than the active window.	RW	0x107

Table 12-348. Register Call Summary for Register VENC_LAL_PHASE_RESET

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2] [3]

Table 12-349. VENC_HS_INT_START_STOP_X

Address Offset	0x70		
Physical address	0x4805 0C70	**Instance**	VENC
Description	VENC_HS_INT_START_STOP_X		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								HS_INT_STOP_X								Reserved								HS_INT_START_X							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	HS_INT_STOP_X	HSYNC internal stop. These bits define HSYNC internal stop pixel value	RW	0x07E
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	HS_INT_START_X	HSYNC internal start. These bits define HSYNCI NTERNAL start pixel value	RW	0x34E

Table 12-350. Register Call Summary for Register VENC_HS_INT_START_STOP_X

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2]

Table 12-351. VENC_HS_EXT_START_STOP_X

Address Offset	0x74		
Physical address	0x4805 0C74	**Instance**	VENC
Description	VENC_HS_EXT_START_STOP_X		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								HS_EXT_STOP_X								Reserved								HS_EXT_START_X							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	HS_EXT_STOP_X	HSYNC external stop. These bits define HSYNC external stop pixel value	RW	0x00F
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	HS_EXT_START_X	HSYNC external start. These bits define HSYNC EXTERNAL start pixel value	RW	0x359

Table 12-352. Register Call Summary for Register VENC_HS_EXT_START_STOP_X

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2]

Table 12-353. VENC_VS_INT_START_X

Address Offset	0x78		
Physical address	0x4805 0C78	**Instance**	VENC
Description	VENC_VS_INT_START_X		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VS_INT_START_X								Reserved															

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	VS_INT_START_X	VSYNC internal start. These bits define VSYNC internal start pixel value.	RW	0x1A0
15:0	Reserved	Reserved. Read returns 0s.	RW	0x0000

Table 12-354. Register Call Summary for Register VENC_VS_INT_START_X

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2]

Table 12-355. VENC_VS_INT_STOP_X_VS_INT_START_Y

Address Offset	0x7C		
Physical address	0x4805 0C7C	**Instance**	VENC
Description	VENC VS_INT_STOP_X and VS_INT_START_Y		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VS_INT_START_Y								Reserved						VS_INT_STOP_X									

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	VS_INT_START_Y	VSYNC internal start. These bits define VSYNC INTERNAL start line value	RW	0x209
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	VS_INT_STOP_X	VSYNC internal stop. These bits define VSYNC internal stop pixel value	RW	0x1A0

Table 12-356. Register Call Summary for Register VENC_VS_INT_STOP_X_VS_INT_START_Y

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-357. VENC_VS_INT_STOP_Y_VS_EXT_START_X

Address Offset	0x80		
Physical address	0x4805 0C80	**Instance**	VENC
Description	VENC VS_INT_STOP_Y and VS_EXT_START_X		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VS_EXT_START_X								Reserved								VS_INT_STOP_Y							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	VS_EXT_START_X	VSYNC external start. These bits define VSYNC external start pixel value.	RW	0x1AC
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	VS_INT_STOP_Y	VSYNC internal stop. These bits define VSYNC INTERNAL stop line value.	RW	0x022

Table 12-358. Register Call Summary for Register VENC_VS_INT_STOP_Y_VS_EXT_START_X

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-359. VENC_VS_EXT_STOP_X_VS_EXT_START_Y

Address Offset	0x84		
Physical address	0x4805 0C84	**Instance**	VENC
Description	VENC VS_EXT_STOP_X and VS_EXT_START_Y		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VS_EXT_START_Y								Reserved								VS_EXT_STOP_X							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	VS_EXT_START_Y	VSYNC external start. These bits define VSYNC EXTERNAL start line value.	RW	0x20D
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	VS_EXT_STOP_X	VSYNC external stop. These bits define VSYNC EXTERNAL stop pixel value.	RW	0x1AC

Table 12-360. Register Call Summary for Register VENC_VS_EXT_STOP_X_VS_EXT_START_Y

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-361. VENC_VS_EXT_STOP_Y

Address Offset	0x88		
Physical address	0x4805 0C88	**Instance**	VENC
Description	VENC VS_EXT_STOP_Y		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						VS_EXT_STOP_Y									

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Reserved. Read returns 0s.	RW	0x000000
9:0	VS_EXT_STOP_Y	VSYNC external stop. These bits define VSYNC EXTERNAL stop line value.	RW	0x006

Table 12-362. Register Call Summary for Register VENC_VS_EXT_STOP_Y

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-363. VENC_AVID_START_STOP_X

Address Offset	0x90		
Physical address	0x4805 0C90	**Instance**	VENC
Description	VENC_AVID_START_STOP_X		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AVID_STOP_X								Reserved								AVID_START_X							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	AVID_STOP_X	AVID stop. These bits define AVID stop pixel value	RW	0x348
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	AVID_START_X	AVID start. These bits define AVID start pixel value	RW	0x078

Table 12-364. Register Call Summary for Register VENC_AVID_START_STOP_X

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2] [3]

Table 12-365. VENC_AVID_START_STOP_Y

Address Offset	0x94		
Physical address	0x4805 0C94	**Instance**	VENC
Description	VENC_AVID_START_STOP_Y		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AVID_STOP_Y								Reserved								AVID_START_Y							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	AVID_STOP_Y	AVID stop. These bits define AVID stop line value.	RW	0x206
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	AVID_START_Y	AVID start. These bits define AVID start line value	RW	0x026

Table 12-366. Register Call Summary for Register VENC_AVID_START_STOP_Y

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]
- Video Encoder Registers: [2] [3]

Table 12-367. VENC_FID_INT_START_X_FID_INT_START_Y

Address Offset	0xA0		
Physical address	0x4805 0CA0	**Instance**	VENC
Description	VENC_FID_INT_START_X and FID_INT_START_Y		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FID_INT_START_Y								Reserved								FID_INT_START_X							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	FID_INT_START_Y	FID internal start. These bits define FID internal start line value	RW	0x001
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	FID_INT_START_X	FID internal start. These bits define FID internal start pixel value	RW	0x08A

Table 12-368. Register Call Summary for Register VENC_FID_INT_START_X_FID_INT_START_Y

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-369. VENC_FID_INT_OFFSET_Y_FID_EXT_START_X

Address Offset	0xA4		
Physical address	0x4805 0CA4	**Instance**	VENC
Description	VENC FID_INT_OFFSET_Y and FID_EXT_START_X		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FID_EXT_START_X								Reserved								FID_INT_OFFSET_Y							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	FID_EXT_START_X	FID external start. These bits define FID external start pixel value	RW	0x1AC
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	FID_INT_OFFSET_Y	FID internal offset. These bits define FID internal offset linel value	RW	0x106

Table 12-370. Register Call Summary for Register VENC_FID_INT_OFFSET_Y_FID_EXT_START_X

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-371. VENC_FID_EXT_START_Y_FID_EXT_OFFSET_Y

Address Offset	0xA8		
Physical address	0x4805 0CA8	**Instance**	VENC
Description	VENC FID_EXT_START_Y and FID_EXT_OFFSET_Y		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FID_EXT_OFFSET_Y								Reserved								FID_EXT_START_Y							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	FID_EXT_OFFSET_Y	FID external offset. These bits define FID external offset line value	RW	0x106
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	FID_EXT_START_Y	FID external start. These bits define FID external start line value.	RW	0x006

Table 12-372. Register Call Summary for Register VENC_FID_EXT_START_Y_FID_EXT_OFFSET_Y

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [0]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [1]

Table 12-373. VENC_TVDETGP_INT_START_STOP_X

Address Offset	0xB0		
Physical address	0x4805 0CB0	**Instance**	VENC
Description	TV Detection Start and Stop pixel values		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TVDETGP_INT_STOP_X								Reserved								TVDETGP_INT_START_X							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	TVDETGP_INT_STOP_X	TVDETGP internal stop. These bits define TVDETGP internal stop pixel value.	RW	0x014
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	TVDETGP_INT_START_X	TVDETGP internal start. These bits define TVDETGP internal start pixel value	RW	0x001

Table 12-374. Register Call Summary for Register VENC_TVDETGP_INT_START_STOP_X

Display Subsystem Functional Description
- TV Detection/Disconnection Pulse Generation: [0]
- TV Detection Procedure: [1] [2] [3]
- TV Disconnection Procedure: [4] [5] [6]
- Recommended TV Detection/Disconnection Pulse Waveform: [7]

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [8]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [9]

Table 12-375. VENC_TVDETGP_INT_START_STOP_Y

Address Offset	0xB4		
Physical address	0x4805 0CB4	**Instance**	VENC
Description	TV detection Start and Stop line values		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TVDETGP_INT_STOP_Y								Reserved								TVDETGP_INT_START_Y							

Bits	Field Name	Description	Type	Reset
31:26	Reserved	Reserved. Read returns 0s.	RW	0x00
25:16	TVDETGP_INT_STOP_Y	TVDETGP internal stop. These bits define TVDETGP internal stop line value.	RW	0x001
15:10	Reserved	Reserved. Read returns 0s.	RW	0x00
9:0	TVDETGP_INT_START_Y	TVDETGP internal start. These bits define TVDETGP internal start line value.	RW	0x001

Table 12-376. Register Call Summary for Register VENC_TVDETGP_INT_START_STOP_Y

Display Subsystem Functional Description
- TV Detection/Disconnection Pulse Generation: [0]
- TV Detection Procedure: [1] [2] [3]
- TV Disconnection Procedure: [4] [5] [6]

Display Subsystem Basic Programming Model
- Video Encoder Register Settings: [7]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [8]

Table 12-377. VENC_GEN_CTRL

Address Offset	0xB8		
Physical address	0x4805 0CB8	**Instance**	VENC
Description	TVDETGP enable and SYNC_POLARITY and UVPHASE_POL		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					MS	656	CBAR	HIP	VIP	HEP	VEP	AVIDP	FIP	FEP	TVDP	Reserved															EN

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved. Read returns 0s.	RW	0x0000
26	MS	UVPHASE_POL MS mode UV phase	RW	0
		0x0: CbCr		
		0x1: CrCb		
25	656	UVPHASE_POL 656 input mode UV phase	RW	0
		0x0: CbCr		
		0x1: CrCb		
24	CBAR	UVPHASE_POL CBAR mode UV phase	RW	0
		0x0: CbCr		
		0x1: CrCb		
23	HIP	HSYNC internal polarity	RW	1
		0x0: Active low		
		0x1: Active high		
22	VIP	VSYNC internal polarity	RW	1
		0x0: Active low		
		0x1: Active high		
21	HEP	HSYNC external polarity	RW	1
		0x0: Active low		
		0x1: Active high		
20	VEP	VSYNC external polarity	RW	1
		0x0: Active low		
		0x1: Active high		
19	AVIDP	AVID polarity	RW	1
		0x0: Active low		
		0x1: Active high		
18	FIP	FID internal polarity	RW	1
		0x0: Active low		
		0x1: Active high		
17	FEP	FID external polarity	RW	1
		0x0: Active low		
		0x1: Active high		
16	TVDP	TVDETGP polarity	RW	1
		0x0: Active low		
		0x1: Active high		
15:1	Reserved	Reserved. Read returns 0s.	RW	0x00
0	EN	TVDETGP generation enable	RW	0
		0x0: Disabled		
		0x1: Enabled		

Table 12-378. Register Call Summary for Register VENC_GEN_CTRL

Display Subsystem Functional Description • TV Detection/Disconnection Pulse Generation: [0] [1] • TV Detection Procedure: [2] [3] • TV Disconnection Procedure: [4] [5]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [6]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [7]

Table 12-379. VENC_OUTPUT_CONTROL

Address Offset	0x0000 00C4		
Physical Address	0x4805 0CC4	**Instance**	VENC
Description	Output channel control register Also contains some test control features		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								LUMA_TEST								RESERVED								CHROMA_SOURCE	COMPOSITE_SOURCE	LUMA_SOURCE	TEST_MODE	VIDEO_INVERT	CHROMA_ENABLE	COMPOSITE_ENABLE	LUMA_ENABLE

Bits	Field Name	Description	Type	Reset
31:26	RESERVED	Reserved. Read returns 0s.	RW	0x00
25:16	LUMA_TEST	In test mode, DAC 1 input value (if s-video video mode is selected)	RW	0x000
15:8	RESERVED	Reserved. Read returns 0s.	RW	0x00
7	CHROMA_SOURCE	Source of chroma video data in test mode 0x0: Chroma test data comes from internal register OUTPUT_TEST[25:16] 0x1: Chroma test data comes from display controller video port G[1:0], B[7:0]	RW	0x0
6	COMPOSITE_ SOURCE	Source of composite video data in test mode 0x0: Composite test data comes from internal register OUTPUT_TEST[9:0] 0x1: Composite test data comes from display controller video port G[1:0], B[7:0]	RW	0x0
5	LUMA_SOURCE	Source of luminance video data in test mode 0x0: Luma test data comes from internal register OUTPUT_CONTROL[25:16] 0x1: Luma test data comes from display controller video port G[1:0], B[7:0]	RW	0x0
4	TEST_MODE	This enables the video DACs to be tested. The values sent to the DACs comes from a register for each output channel (Luma, Composite or Chroma) or from the display controller video port bits G[1:0], B[7:0], depending on the setting of the Source bits 0x0: Video outputs are in normal operation 0x1: Test mode. Video outputs are directly connected to either internal registers or the display controller video port.	RW	0x0
3	VIDEO_INVERT	Controls the video output polarity. This may be used to correct for inversion in an external video amplifier. 0x0: Video outputs are inverted 0x1: Video outputs are normal polarity	RW	0x1
2	CHROMA_ENABLE	Enable the Chrominance output channel 0x0: Chroma output is disabled 0x1: Chroma output is enabled	RW	0x0
1	COMPOSITE_ ENABLE	Enable the Composite output channel 0x0: Composite output is disabled 0x1: Composite output is enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
0	LUMA_ENABLE	Enable the Luminance output channel 0x0: Luma output is disabled 0x1: Luma output is enabled	RW	0x0

Table 12-380. Register Call Summary for Register VENC_OUTPUT_CONTROL

Display Subsystem Environment • TV Display Support: [0] [1]
Display Subsystem Functional Description • TV Detection Procedure: [2] [3] • TV Disconnection Procedure: [4] [5] • Video Dual-DAC Test Mode: [6] [7] [8] [9] [10]
Display Subsystem Basic Programming Model • Video DAC Settings: [11] [12] [13] • Video Encoder Register Settings: [14]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [15]

Table 12-381. VENC_OUTPUT_TEST

Address Offset	0x0000 00C8		
Physical Address	0x4805 0CC8	**Instance**	VENC
Description	Test values for the Luma/Composite Video DAC1 (if composite video is selected) and the Chroma Video DAC2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CHROMA_TEST								RESERVED								COMPOSITE_TEST							

Bits	Field Name	Description	Type	Reset
31:26	RESERVED	Reserved. Read returns 0s.	RW	0x00
25:16	CHROMA_TEST	In test mode, DAC 2 input value	RW	0x000
15:10	RESERVED	Reserved. Read returns 0s.	RW	0x00
9:0	COMPOSITE_TEST	In test mode, DAC 1 input value (if composite video is selected)	RW	0x000

Table 12-382. Register Call Summary for Register VENC_OUTPUT_TEST

Display Subsystem Functional Description • Video Dual-DAC Test Mode: [0] [1]
Display Subsystem Basic Programming Model • Video Encoder Register Settings: [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

12.7.2.5 DSI Protocol Engine Registers

Table 12-383. DSI_REVISION

Address Offset	0x0000 0000		
Physical Address	0x4804 FC00	**Instance**	DSI_PROTOCOL_ENGINE
Description	MODULE REVISION This register contains the IP revision code in binary coded digital. For example, we have: 0x01 = revision 0.1 and 0x21 = revision 2.1		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Write 0s for future compatibility. Reads returns 0.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision	R	TI internal data

Table 12-384. Register Call Summary for Register DSI_REVISION

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [0]

Table 12-385. DSI_SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x4804 FC10	**Instance**	DSI_PROTOCOL_ENGINE
Description	SYSTEM CONFIGURATION REGISTER This register is the system configuration register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		RESERVED				CLOCKACTIVITY		RESERVED			SIDLEMODE		ENWAKEUP	SOFT_RESET	AUTO_IDLE

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00000
13:10	RESERVED	Write 0s for future compatibility.	RW	0x0
9:8	CLOCKACTIVITY	Clocks activity during wake up mode period 0x0: Interface and Functional clocks can be switched off 0x1: Functional clocks can be switched off and Interface clocks are maintained during wake up period 0x2: Interface clocks can be switched off and Functional clocks are maintained during wake up period 0x3: Interface and Functional clocks are maintained during wake up period	RW	0x0
7:5	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
4:3	SIDLEMODE	Slave interface power management, Idle req/ack control 0x0: Force-idle. An idle request is acknowledged unconditionally 0x1: No-idle. An idle request is never acknowledged 0x2: Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module. 0x3: Reserved	RW	0x2
2	ENWAKEUP	Wake-up mode enable bit 0x0: Wakeup is disabled. 0x1: Wakeup is enabled,	RW	0x0

Bits	Field Name	Description	Type	Reset
1	SOFT_RESET	Software reset. Set the bit to 1 to trigger a module reset. The bit is automatically reset by the hw. During reads return 0. 0x0: Normal mode. 0x1: The module is reset	RW	0x0
0	AUTO_IDLE	Internal interface clock gating strategy 0x0: Interface clock is free-running. 0x1: Automatic Interface clock gating strategy is applied based on the module interface activity.	RW	0x1

Table 12-386. Register Call Summary for Register DSI_SYSCONFIG

Display Subsystem Integration
- Software Reset: [0]
- Clock Activity Mode: [1] [2] [3]
- Autoidle Mode: [4]
- Idle Mode: [5] [6] [7]

Display Subsystem Basic Programming Model
- Software Reset: [8]
- Power Management: [9] [10]
- Software Reset: [11]

Display Subsystem Use Cases and Tips
- Reset DSI Modules: [12]
- Reset DSI Modules: [13] [14] [15]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [16]

Table 12-387. DSI_SYSSTATUS

Address Offset	0x0000 0014		
Physical Address	0x4804 FC14	**Instance**	DSI_PROTOCOL_ENGINE
Description	SYSTEM STATUS REGISTER This register provides status information about the module, excluding the interrupt status register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															RESET_DONE

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reads returns 0.	R	0x00000000
0	RESET_DONE	Internal reset monitoring 0x0: Internal module reset is on going. 0x1: Reset completed.	R	0x1

Table 12-388. Register Call Summary for Register DSI_SYSSTATUS

Display Subsystem Basic Programming Model • Software Reset: [0] [1]
Display Subsystem Use Cases and Tips • Reset DSI Modules: [2] • Set Up DSI Control Registers: [3] • Reset DSI Modules: [4] • Configure DSI Protocol Engine: [5]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [6]

Table 12-389. DSI_IRQSTATUS

Address Offset	0x0000 0018		
Physical Address	0x4804 FC18	**Instance**	DSI_PROTOCOL_ENGINE
Description	INTERRUPT STATUS REGISTER - All VCs + complex I/O + PLL This register associates one bit for each VC to determine which VC has generated the interrupt. The VC should be enabled for events to be generated on that VC. If the VC is disabled, the interrupt is not generated.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											TA_TO_IRQ	LDO_POWER_GOOD_IRQ	SYNC_LOST_IRQ	ACK_TRIGGER_IRQ	TE_TRIGGER_IRQ	LP_RX_TO_IRQ	HS_TX_TO_IRQ	RESERVED	RESERVED		COMPLEXIO_ERR_IRQ	PLL_RECAL_IRQ	PLL_UNLOCK_IRQ	PLL_LOCK_IRQ	RESERVED	RESYNCHRONIZATION_IRQ	WAKEUP_IRQ	VIRTUAL_CHANNEL3_IRQ	VIRTUAL_CHANNEL2_IRQ	VIRTUAL_CHANNEL1_IRQ	VIRTUAL_CHANNEL0_IRQ

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x000
20	TA_TO_IRQ	Turn-around Time out. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
19	LDO_POWER_GOOD_IRQ	Transition of the status signal LDOPWRGOOD from the DSI_PHY indicating a state change for the supply VDDALDODSIPLL from up to down or down to up. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
18	SYNC_LOST_IRQ	Synchronization with Video port is lost (Video mode only) 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Bits	Field Name	Description	Type	Reset
17	ACK_TRIGGER_IRQ	Acknowledge Trigger 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
16	TE_TRIGGER_IRQ	Tearing Effect Trigger 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
15	LP_RX_TO_IRQ	Interrupt for Low Power Rx Time out 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
14	HS_TX_TO_IRQ	Interrupt for high-speed Tx Time out. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
13	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
12:11	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
10	COMPLEXIO_ ERR_IRQ	Error signaling from complex I/O: status of the complex I/O errors received from the complex I/O(events are defined in DSI_COMPLEXIO_IRQSTATUS). 0x0: READS: Event is false. 0x1: READS: Event is true (pending).	R	0x0
9	PLL_RECAL_IRQ	PLL recalibration event (assertion of recalibration signal from the DSI PLL Control module) 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
8	PLL_UNLOCK_IRQ	PLL un-lock event (deassertion of lock signal from the DSI PLL Control module) 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
7	PLL_LOCK_IRQ	PLL lock event (assertion of lock signal from the DSI PLL Control module) 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
6	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
5	RESYNCHRONIZATION_ IRQ	Video mode resynchronization Indicates that the video port works but the configuration of the timings for the display controller (DISPC) and for DSI protocol engine may have to be modified to avoid the resynchronization to occur. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	WAKEUP_IRQ	Wakeup 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
3	VIRTUAL_CHANNEL3_IRQ	Virtual channel #3 Error signaling from DSI Virtual Channel3: Status of DSI Virtual Channel3 errors received from DSI Virtual Channel3 (events are defined in DSI_VC3_IRQSTATUS). 0x0: READS: Event is false. 0x1: READS: Event is true (pending).	R	0x0
2	VIRTUAL_CHANNEL2_IRQ	Virtual channel #2 Error signaling from DSI Virtual Channel2: Status of DSI Virtual Channel2 errors received from DSI Virtual Channel2 (events are defined in DSI_VC2_IRQSTATUS). 0x0: READS: Event is false. 0x1: READS: Event is true (pending).	R	0x0
1	VIRTUAL_CHANNEL1_IRQ	Virtual channel #1 Error signaling from DSI Virtual Channel1: Status of DSI Virtual Channel1 errors received from DSI Virtual Channel1 (events are defined in DSI_VC1_IRQSTATUS). 0x0: READS: Event is false. 0x1: READS: Event is true (pending).	R	0x0
0	VIRTUAL_CHANNEL0_IRQ	Virtual channel #0 Error signaling from DSI Virtual Channel0: Status of the DSI Virtual Channel0 errors received from DSI Virtual Channel0 (events are defined in DSI_VC0_IRQSTATUS). 0x0: READS: Event is false. 0x1: READS: Event is true (pending).	R	0x0

Table 12-390. Register Call Summary for Register DSI_IRQSTATUS

Display Subsystem Integration • DSI Interrupt Request: [0]
Display Subsystem Functional Description • HS TX Timer: [1] • LP RX Timer: [2] • Tearing Effect: [3] • Acknowledge: [4] • Error Handling: [5] [6] [7]
Display Subsystem Basic Programming Model • Interrupts: [8] [9] • Video Mode: [10] [11] • DSI PLL Error Handling: [12] [13] [14] [15] [16] [17] • Error Handling: [18]
Display Subsystem Use Cases and Tips • Reset DSI Modules: [19] • Reset DSI Modules: [20]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [21] • Display Subsystem and SDI Registers: [22]

Table 12-391. DSI_IRQENABLE

Address Offset	0x0000 001C		
Physical Address	0x4804 FC1C	**Instance**	DSI_PROTOCOL_ENGINE
Description	INTERRUPT ENABLE REGISTER - This register associates one bit for each VC to enable/disable each VC individually.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											TA_TO_IRQ_EN	LDO_POWER_GOOD_IRQ_EN	SYNC_LOST_IRQ_EN	ACK_TRIGGER_IRQ_EN	TE_TRIGGER_IRQ_EN	LP_RX_TO_IRQ_EN	HS_TX_TO_IRQ_EN	RESERVED	RESERVED	RESERVED	PLL_RECAL_IRQ_EN	PLL_UNLOCK_IRQ_EN	PLL_LOCK_IRQ_EN	RESERVED	RESYNCHRONIZATION_IRQ_EN	WAKEUP_IRQ_EN	RESERVED				

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x000
20	TA_TO_IRQ_EN	Turn-around Time out. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
19	LDO_POWER_GOOD_ IRQ_EN	Transition of the status signal LDOPWRGOOD from the DSI_PHY indicating a state change for the supply VDDALDODSIPLL from up to down or down to up. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
18	SYNC_LOST_IRQ_EN	Synchronization with Video port is lost (Video mode only) 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
17	ACK_TRIGGER_IRQ_EN	Acknowledge trigger 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
16	TE_TRIGGER_IRQ_EN	Tearing Effect trigger 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
15	LP_RX_TO_IRQ_EN	Interrupt for Low Power Rx Time out. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
14	HS_TX_TO_IRQ_EN	Interrupt for high-speed Tx Time out. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
13	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
12:11	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
10	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
9	PLL_RECAL_IRQ_EN	PLL recalibration event (assertion of recalibration signal from the DSI PLL Control module) 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
8	PLL_UNLOCK_IRQ_EN	PLL un-lock event (deassertion of lock signal from the DSI PLL Control module) 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
7	PLL_LOCK_IRQ_EN	PLL lock event (assertion of lock signal from the DSI PLL Control module) 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
6	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
5	RESYNCHRONIZATION_IRQ_EN	Resynchronization 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
4	WAKEUP_IRQ_EN	Wakeup 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
3:0	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0

Table 12-392. Register Call Summary for Register DSI_IRQENABLE

Display Subsystem Functional Description • Tearing Effect: [0] • Acknowledge: [1]
Display Subsystem Basic Programming Model • Interrupts: [2]
Display Subsystem Use Cases and Tips • Set Up DSI Control Registers: [3] • Configure DSI Protocol Engine: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

Table 12-393. DSI_CTRL

Address Offset	0x0000 0040		
Physical Address	0x4804 FC40	**Instance**	DSI_PROTOCOL_ENGINE
Description	GLOBAL CONTROL REGISTER This register controls the DSI Protocol Engine module. This register should not be modified dynamically (except IF_EN bit field).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					RGB565_ORDER	DCS_CMD_CODE	DCS_CMD_ENABLE	HSA_BLANKING_MODE	HBP_BLANKING_MODE	HFP_BLANKING_MODE	BLANKING_MODE	EOT_ENABLE	VP_HSYNC_END	VP_HSYNC_START	VP_VSYNC_END	VP_VSYNC_START	TRIGGER_RESET_MODE	LINE_BUFFER		VP_VSYNC_POL	VP_HSYNC_POL	VP_DE_POL	VP_CLK_POL	VP_DATA_BUS_WIDTH		TRIGGER_RESET	VP_CLK_RATIO	TX_FIFO_ARBITRATION	ECC_RX_EN	CS_RX_EN	IF_EN

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00
26	RGB565_ORDER	Byte order for RBG565 command mode from video port 0x0: 0x1: Byte order as for video mode	RW	0
25	DCS_CMD_CODE	DCS command code value to insert between header and video port data when enabled by DCS_CMD_ENABLE 0x0: DCS write memory continue code is inserted. 0x1: DCS write memory start code is inserted.	RW	0
24	DCS_CMD_ENABLE	Enables automatic insertion of DCS command codes when data is sourced by the video port. 0x0: DCS command code is not inserted when command mode traffic is coming from the Video Port. 0x1: DCS command code is inserted automatically when command mode traffic is coming from the Video Port.	RW	0
23	HSA_BLANKING_MODE	Blanking mode 0x0: Packets in TX FIFO are sent during HSA blanking period of video mode or LPS is used. 0x1: LONG BLANKING PACKETS only are used during HSA blanking period of video mode.	RW	0x0
22	HBP_BLANKING_MODE	Blanking mode 0x0: Packets in TX FIFO are sent during HBP blanking period of video mode or LPS is used. 0x1: LONG BLANKING PACKETS only are used during HBP blanking period of video mode.	RW	0x0
21	HFP_BLANKING_MODE	Blanking mode 0x0: Packets in TX FIFO are sent during HFP blanking period of video mode or LPS is used. 0x1: LONG BLANKING PACKETS only are used during HFP blanking period of video mode.	RW	0x0
20	BLANKING_MODE	Blanking mode 0x0: LPS is used during blanking periods of video mode (except HSA, HBP, HFP defined in HSA_BLANKING_MODE, HBP_BLANKING_MODE and HFP_BLANKING_MODE bit fields respectively) when there is no command mode data in TX FIFO ready to be sent. So blanking periods can be different during the frame depending on the TX FIFO. 0x1: LONG BLANKING PACKETS are used during blanking periods of video mode (except HSA, HBP, HFP defined in HSA_BLANKING_MODE, HBP_BLANKING_MODE and HFP_BLANKING_MODE bit fields respectively) regardless of the packets present in the TX FIFO ready to be sent	RW	0x0
19	EOT_ENABLE	Enable EOT packets at the end of HS transmission. 0x0: No EOT packets 0x1: EOT packet is sent at all HS to LP transitions	RW	0
18	VP_HSYNC_END	HSYNC end pulse. 0x0: Disabled. No HSYNC END short packet is generated. 0x1: Enabled. While the HSYNC END pulse is detected, the associated short packet HSYNC END is generated.	RW	0x0
17	VP_HSYNC_START	HSYNC start pulse. 0x0: Disabled. No HSYNC START short packet is generated. 0x1: Enabled. While the HSYNC start pulse is detected, the associated short packet HSYNC START is generated.	RW	0x0

Bits	Field Name	Description	Type	Reset
16	VP_VSYNC_END	VSYNC end pulse. 0x0: Disabled. No VSYNC END short packet is generated. 0x1: Enabled. While the VSYNC END pulse is detected, the associated short packet VSYNC END is generated.	RW	0x0
15	VP_VSYNC_START	VSYNC start pulse. 0x0: Disabled. No VSYNC START short packet is generated. 0x1: Enabled. While the VSYNC START pulse is detected, the associated short packet VSYNC START is generated.	RW	0x0
14	TRIGGER_RESET_MODE	Selection of the trigger reset mode 0x0: Synchronized: the mode is only valid if there is VC using the video mode and it is active. The principle is to wait for the current video frame to be transferred on the link. Any data received after the VSYNC are ignored. 0x1: Immediate: all pending requests in TX FIFO are taken into account for transfer scheduling, the RX FIFO is ignored, and the data from video port are ignored as soon as possible. Only the current transfer on DSI link and already scheduled ones are transmitted. All the other transfers are discarded.	RW	0x0
13:12	LINE_BUFFER	Number of line buffers to be used while receiving data on the video port. 0x0: No line buffer 0x1: 1 line buffer 0x2: 2 line buffers	RW	0x0
11	VP_VSYNC_POL	VP vertical synchronization signal polarity 0x0: VSYNC signal on the video port is active low. 0x1: VSYNC signal on the video port is active high.	RW	0x0
10	VP_HSYNC_POL	VP horizontal synchronization signal polarity 0x0: HSYNC signal on the video port is active low. 0x1: HSYNC signal on the video port is active high.	RW	0x0
9	VP_DE_POL	VP data enable signal polarity 0x0: DE signal on the video port is active low. 0x1: DE signal on the video port is active high.	RW	0x0
8	VP_CLK_POL	VP pixel clock polarity 0x0: The DSI Protocol Engine module captures the data on the VP on the pixel clock falling edge. The module connected to the VP must drive the data on the pixel clock rising edge. 0x1: The DSI Protocol Engine module captures the data on the VP on the pixel clock raising edge. The module connected to the VP must drive the data on the pixel clock falling edge.	RW	0x1
7:6	VP_DATA_BUS_WIDTH	Defines the size of the video port data bus 0x0: 16-bits data width (LSB of the 24-bit video port data bus) 0x1: 18-bits data width (LSB of the 24-bit video port data bus) 0x2: 24-bits data width (LSB of the 24-bit video port data bus)	RW	0x0
5	TRIGGER_RESET	Send the reset trigger to the peripheral. 0x0: READS: Reset trigger generation is completed. It is reset by HW when it is completed. WRITES: Cancellation of the request for Reset trigger generation (maybe too late since it is already on going) 0x1: READS: Generation of the reset trigger has been requested by user (could be on going but not completed yet). WRITES: Request for Reset trigger to be sent to the peripheral.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	VP_CLK_RATIO	This bit indicates the clock ratio between VP_CLK and VP_PCLK. The clock VP_PCLK is generated from VP_CLK. It is divided down. The information is only used when the video port is used to provide data in command mode. In the case of video mode, it is not used. 0x0: The clock VP_PCLK is the clock VP_CLK divided by 2. The duty cycle of VP_PCLK is 50/50. 0x1: The clock VP_PCLK is the clock VP_CLK divided by 3 or more. The duty cycle of VP_PCLK is not 50/50 for odd ratio numbers (3,5,7,...).	RW	0x0
3	TX_FIFO_ ARBITRATION	Defines the arbitration scheme for granting the VC pending ready requests in the TX FIFO 0x0: Round-Robin Scheme is used 0x1: Sequential Scheme is used	RW	0x0
2	ECC_RX_EN	Enables the ECC check for the received header (short and long packets for all VC IDs). 0x0: Disabled 0x1: Enabled	RW	0x0
1	CS_RX_EN	Enables the checksum check for the received payload (long packet only for all VC IDs). 0x0: Disabled 0x1: Enabled	RW	0x0
0	IF_EN	Enables the module. When the module is disabled the signals from the complex I/O are gated (no updates of the interrupt status register). It is not possible to change the bit fields in the DSI_CTRL register, except IF_EN when it is enabled. All the other registers can be changed except the ones that require DSI_VCn_CTRL[0] VC_EN to be equal to 0 to be modified. 0x0: The interface is disabled. If one of the VC uses the video mode with the video port to receive the data, the DSI protocol engines is disabled when the next VSYNC is received and all the data in the FIFO for the other VCs in command mode are sent to the peripherals (if BTA_EN bit is enabled, the DSI protocol engine needs to wait for the response and BTA from the peripheral before disabling all the internal logic since an acknowledge is requested). 0x1: The interface is enabled immediately, the data acquisition on the video port starts on the next VSYNC (video mode) or first data received in the Slave port FIFO (command mode).	RW	0x0

Table 12-394. Register Call Summary for Register DSI_CTRL

Display Subsystem Environment

- Data/Clock Configuration: [0] [1] [2]
- Video Port (VP) Interface: [3] [4] [5] [6]
- Video Port Used for Video Mode: [7] [8] [9]
- Video Port Used on Command Mode: [10] [11] [12]

Display Subsystem Functional Description

- Clock Requirements: [13]
- Command Mode: [14] [15] [16] [17]
- DSI-PLL Power FSM: [18] [19] [20] [21]
- TurnRequest FSM: [22] [23]
- HS TX Timer: [24] [25]
- LP RX Timer: [26] [27]
- Bus Turnaround: [28] [29] [30] [31]
- Reset: [32] [33] [34]
- Tearing Effect: [35]
- ECC Generation: [36]
- Checksum Generation for Long Packet Payloads: [37]
- End of Transfer Packet: [38]

Table 12-394. Register Call Summary for Register DSI_CTRL (continued)

Display Subsystem Basic Programming Model

- Global Register Controls: [39] [40] [41] [42] [43] [44] [45]
- Packets: [46] [47] [48] [49] [50] [51]
- Video Mode: [52] [53] [54] [55]
- Video Port Data Bus: [56]
- Command Mode TX FIFO: [57] [58]
- Video Mode Transfer: [59]
- Command Mode Transfer Example 1: [60]
- Command Mode Transfer Example 2: [61]

Display Subsystem Use Cases and Tips

- Set Up DSI Control Registers: [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75]
- Enable Video Mode Using the DISPC Video Port: [76] [77]
- Configure DSI Protocol Engine: [78] [79] [80] [81] [82] [83] [84] [85] [86]
- Enable Command Mode Using DISPC Video Port: [87] [88] [89] [90]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [91]
- DSI Protocol Engine Registers: [92] [93] [94] [95] [96] [97] [98] [99]

Table 12-395. DSI_COMPLEXIO_CFG1

Address Offset	0x0000 0048		
Physical Address	0x4804 FC48	**Instance**	DSI_PROTOCOL_ENGINE
Description	COMPLEXIO CONFIGURATION REGISTER for the complex I/O This register contains the lane configuration for the order and position of the lanes (clock and data) and the polarity order for the control of the PHY differential signals in addition to the control bit for the power FSM.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHADOWING	GOBIT	RESET_DONE	PWR_CMD		PWR_STATUS		RESERVED			LDO_POWER_GOOD_STATE	USE_LDO_EXTERNAL	RESERVED		RESERVED		RESERVED		RESERVED		DATA2_POL		DATA2_POSITION		DATA1_POL		DATA1_POSITION		CLOCK_POL		CLOCK_POSITION	

Bits	Field Name	Description	Type	Reset
31	SHADOWING	Shadowing configuration. 0x0: Disabled. The writes to the DSIPHY_CFG0 and DSIPHY_CFG1 registers are done like the other SCP registers. 0x1: Enabled. The writes to the DSIPHY_CFG0 and DSIPHY_CFG1 registers are done only when the GO bit is set and when the signal DISPC_UPDATE_SYNC from the display controller module is active.	RW	0x0

Bits	Field Name	Description	Type	Reset
30	GOBIT	Allows the synchronized update of the shadow registers when the signal DISPC_UPDATE_SYNC is active. 0x0: Resets the Gobit. The hardware has finished the update of the shadow SCP registers. The bit is reset by Hardware. The software can reset the bit in case users decide to abort it. There is no guarantee that the software reset is done before the transfer of the values to the complex I/O. 0x1: Set the Gobit. Only when the transfer of the new values for the two first registers is completed (2, 1, or 0 transfers are performed based on the number of registers to update), the GObit is reset. The DISPC_UPDATE_SYNC signal is used to synchronize the update. The bit must be set only when it is in reset state.	RW	0
29	RESET_DONE	Internal reset monitoring of the power domain using TxByteClkHS from the complex I/O 0x0: Internal module reset is on going. 0x1: Reset completed.	R	1
28:27	PWR_CMD	Command for power control of the complex I/O 0x0: Command to change to OFF state 0x1: Command to change to ON state 0x2: Command to change to ultralow-power state	RW	0x0
26:25	PWR_STATUS	Status of the power control of the complex I/O 0x0: complex I/O in OFF state 0x1: complex I/O in ON state 0x2: complex I/O in ultralow-power state	R	0x0
24:22	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
21	LDO_POWER_GOOD_STATE	Indicates the state of the signal LDOPWRGOOD. VDDALDODSIPLL: 1.2-V power supply for the PLL. The voltage is supplied by the internal or external LDO. The interrupt LDO_POWER_GOOD_IRQ is generated when a transition is detected on the signal LDOPWRGOOD from the DSI_PHY. 0x0: VDDALDODSIPLL power supply is down 0x1: VDDALDODSIPLL power supply is up	R	0x0
20	USE_LDO_EXTERNAL	Select the external LDO for the DSI_PHY. 0x0: DSI_PHY internal LDO is used. 0x1: External LDO is used. DSI_PHY LDO is tri-stated.	RW	0x0
19	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
18:16	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
15	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
14:12	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
11	DATA2_POL	+/- differential pin order of DATA lane 2. 0x0: +/- pin order (dsi_dx=+ and dsi_dy=-) 0x1: -/+ pin order (dsi_dx=- and dsi_dy=+)	RW	0x0
10:8	DATA2_POSITION	Position and order of the DATA lane 2. 0x0: Not used/connected 0x1: Data lane 2 is at the position 1 (line 1). 0x2: Data lane 2 is at the position 2 (line 2). 0x3: Data lane 2 is at the position 3 (line 3). Other values: reserved	RW	0x0

Bits	Field Name	Description	Type	Reset
7	DATA1_POL	+/- differential pin order of DATA lane 1 0x0: +/- pin order (dsi_dx=+ and dsi_dy=-) 0x1: -/+ pin order (dsi_dx=- and dsi_dy=+)	RW	0x0
6:4	DATA1_POSITION	Position and order of the DATA lane 1. 0x1: Data lane 1 is at the position 1 (line 1). 0x2: Data lane 1 is at the position 2 (line 2). 0x3: Data lane 1 is at the position 3 (line 3). Other values: reserved	RW	0x0
3	CLOCK_POL	+/- differential pin order of CLOCK lane. 0x0: +/- pin order (dsi_dx=+ and dsi_dy=-) 0x1: -/+ pin order (dsi_dx=- and dsi_dy=+)	RW	0x0
2:0	CLOCK_POSITION	Position and order of the CLOCK lane. The clock lane is always present. 0x1: Clock lane is at the position 1 (line 1). 0x2: Clock lane is at the position 2 (line 2). 0x3: Clock lane is at the position 3 (line 3). Other values: reserved	RW	0x0

Table 12-396. Register Call Summary for Register DSI_COMPLEXIO_CFG1

Display Subsystem Environment • Data/Clock Configuration: [0] [1]
Display Subsystem Functional Description • DSI Protocol Architecture: [2] • Shadowing Register: [3] [4] [5] [6] [7] [8] [9] • Complex I/O Power FSM: [10] [11]
Display Subsystem Basic Programming Model • Exiting ULPS: [12] [13] • Software Reset: [14] • Pad Configuration: [15] [16] [17]
Display Subsystem Use Cases and Tips • Set Up DSI Control Registers: [18] [19] [20] • Configure DSI Protocol Engine: [21] [22] [23] [24] [25]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [26]

Table 12-397. DSI_COMPLEXIO_IRQSTATUS

Address Offset	0x0000 004C		
Physical Address	0x4804 FC4C	**Instance**	DSI_PROTOCOL_ENGINE
Description	INTERRUPT STATUS REGISTER - All errors from complex I/O		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ULPSACTIVENOT_ALL1_IRQ	ULPSACTIVENOT_ALL0_IRQ	RESERVED	RESERVED	RESERVED	RESERVED	ERRCONTENTIONLP1_3_IRQ	ERRCONTENTIONLP0_3_IRQ	ERRCONTENTIONLP1_2_IRQ	ERRCONTENTIONLP0_2_IRQ	ERRCONTENTIONLP1_1_IRQ	ERRCONTENTIONLP0_1_IRQ	RESERVED	RESERVED	STATEULPS3_IRQ	STATEULPS2_IRQ	STATEULPS1_IRQ	RESERVED	RESERVED	ERRCONTROL3_IRQ	ERRCONTROL2_IRQ	ERRCONTROL1_IRQ	RESERVED	RESERVED	ERRESC3_IRQ	ERRESC2_IRQ	ERRESC1_IRQ	RESERVED	RESERVED	ERRSYNCESC3_IRQ	ERRSYNCESC2_IRQ	ERRSYNCESC1_IRQ

Bits	Field Name	Description	Type	Reset
31	ULPSACTIVENOT_ ALL1_IRQ	All the ULPSActiveNOT signals corresponding to the lanes with TXULPSExit being high are high. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
30	ULPSACTIVENOT_ ALL0_IRQ	All signals ULPSActiveNOT are 0 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
29	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
28	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
27	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
26	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
25	ERRCONTENTIONLP1_ 3_IRQ	Contention LP1 error for lane #3 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
24	ERRCONTENTIONLP0_ 3_IRQ	Contention LP0 error for lane #3 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
23	ERRCONTENTIONLP1_ 2_IRQ	Contention LP1 error for lane #2 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
22	ERRCONTENTIONLP0_ 2_IRQ	Contention LP0 error for lane #2 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
21	ERRCONTENTIONLP1_ 1_IRQ	Contention LP1 error for lane #1 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Bits	Field Name	Description	Type	Reset
20	ERRCONTENTIONLP0_1_IRQ	Contention LP0 error for lane #1 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
19	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
18	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
17	STATEULPS3_IRQ	Lane #3 in ultralow-power state 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
16	STATEULPS2_IRQ	Lane #2 in ultralow-power state 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
15	STATEULPS1_IRQ	Lane #1 in ultralow-power state 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
14	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
13	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
12	ERRCONTROL3_IRQ	Control error for lane #3 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
11	ERRCONTROL2_IRQ	Control error for lane #2 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
10	ERRCONTROL1_IRQ	Control error for lane #1 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
9	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
8	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
7	ERRESC3_IRQ	Escape entry error for lane #3 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
6	ERRESC2_IRQ	Escape entry error for lane #2 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Bits	Field Name	Description	Type	Reset
5	ERRESC1_IRQ	Escape entry error for lane #1 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
4	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
3	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
2	ERRSYNCESC3_IRQ	Low power Data transmission synchronization error for lane #3 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
1	ERRSYNCESC2_IRQ	Low power Data transmission synchronization error for lane #2 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
0	ERRSYNCESC1_IRQ	Low power Data transmission synchronization error for lane #1 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Table 12-398. Register Call Summary for Register DSI_COMPLEXIO_IRQSTATUS

Display Subsystem Integration • DSI Interrupt Request: [0] [1]
Display Subsystem Functional Description • Twakeup Timer: [2]
Display Subsystem Basic Programming Model • Exiting ULPS: [3] [4] • Error Handling: [5] [6] [7] [8] [9] [10] [11] [12] [13]
Display Subsystem Use Cases and Tips • Set Up DSI Control Registers: [14] • Configure DSI Protocol Engine: [15]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [16] • DSI Protocol Engine Registers: [17]

Table 12-399. DSI_COMPLEXIO_IRQENABLE

Address Offset	0x0000 0050		
Physical Address	0x4804 FC50	**Instance**	DSI_PROTOCOL_ENGINE
Description	INTERRUPT ENABLE REGISTER - All errors from complex I/O		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ULPSACTIVENOT_ALL1_IRQ_EN	ULPSACTIVENOT_ALL0_IRQ_EN	RESERVED	RESERVED	RESERVED	RESERVED	ERRCONTENTIONLP1_3_IRQ_EN	ERRCONTENTIONLP0_3_IRQ_EN	ERRCONTENTIONLP1_2_IRQ_EN	ERRCONTENTIONLP0_2_IRQ_EN	ERRCONTENTIONLP1_1_IRQ_EN	ERRCONTENTIONLP0_1_IRQ_EN	RESERVED	RESERVED	STATEULPS3_IRQ_EN	STATEULPS2_IRQ_EN	STATEULPS1_IRQ_EN	RESERVED	RESERVED	ERRCONTROL3_IRQ_EN	ERRCONTROL2_IRQ_EN	ERRCONTROL1_IRQ_EN	RESERVED	RESERVED	ERRESC3_IRQ_EN	ERRESC2_IRQ_EN	ERRESC1_IRQ_EN	RESERVED	RESERVED	ERRSYNCESC3_IRQ_EN	ERRSYNCESC2_IRQ_EN	ERRSYNCESC1_IRQ_EN

Bits	Field Name	Description	Type	Reset
31	ULPSACTIVENOT_ALL1_IRQ_EN	All the ULPSActiveNOT signals corresponding to the lanes with TXULPSExit being high are high. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
30	ULPSACTIVENOT_ALL0_IRQ_EN	All signals ULPSActiveNOT are 0 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
29	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
28	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
27	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
26	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
25	ERRCONTENTIONLP1_3_IRQ_EN	Contention LP1 error for lane #3 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
24	ERRCONTENTIONLP0_3_IRQ_EN	Contention LP0 error for lane #3 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
23	ERRCONTENTIONLP1_2_IRQ_EN	Contention LP1 error for lane #2 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
22	ERRCONTENTIONLP0_2_IRQ_EN	Contention LP0 error for lane #2 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
21	ERRCONTENTIONLP1_1_IRQ_EN	Contention LP1 error for lane #1 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
20	ERRCONTENTIONLP0_1_IRQ_EN	Contention LP0 error for lane #1 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
19	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
18	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
17	STATEULPS3_IRQ_EN	Lane #3 in ultralow-power state 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
16	STATEULPS2_IRQ_EN	Lane #2 in ultralow-power state 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
15	STATEULPS1_IRQ_EN	Lane #1 in ultralow-power state 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
14	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
13	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
12	ERRCONTROL3_IRQ_EN	Control error for lane #3 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
11	ERRCONTROL2_IRQ_EN	Control error for lane #2 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
10	ERRCONTROL1_IRQ_EN	Control error for lane #1 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
9	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
8	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
7	ERRESC3_IRQ_EN	Escape entry error for lane #3 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
6	ERRESC2_IRQ_EN	Escape entry error for lane #2 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
5	ERRESC1_IRQ_EN	Escape entry error for lane #1 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
4	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
3	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
2	ERRSYNCESC3_IRQ_EN	Low power Data transmission synchronization error for lane #3 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
1	ERRSYNCESC2_IRQ_EN	Low power Data transmission synchronization error for lane #2 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
0	ERRSYNCESC1_IRQ_EN	Low power Data transmission synchronization error for lane #1 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0

Table 12-400. Register Call Summary for Register DSI_COMPLEXIO_IRQENABLE

Display Subsystem Use Cases and Tips

- Set Up DSI Control Registers: [0]
- Configure DSI Protocol Engine: [1]

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [2]

Table 12-401. DSI_CLK_CTRL

Address Offset	0x0000 0054		
Physical Address	0x4804 FC54	**Instance**	DSI_PROTOCOL_ENGINE
Description	CLOCK CONTROL This register controls the CLOCK GENERATION. The register can be modified only when IF_EN is reset.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL_PWR_CMD		PLL_PWR_STATUS		RESERVED						LP_RX_SYNCHRO_ENABLE	LP_CLK_ENABLE	HS_MANUAL_STOP_CTRL	HS_AUTO_STOP_ENABLE	LP_CLK_NULL_PACKET_SIZE		LP_CLK_NULL_PACKET_ENABLE	CIO_CLK_ICG	DDR_CLK_ALWAYS_ON	LP_CLK_DIVISOR										

Bits	Field Name	Description	Type	Reset
31:30	PLL_PWR_CMD	Command for power control of the DSI PLL Control module 0x0: Command to change to OFF state 0x1: Command to change to ON state for PLL only (HSDIVISER is OFF) 0x2: Command to change to ON state for both PLL and HSDIVISER 0x3: Command to change to ON state for both PLL and HSDIVISER (no clock output to the DSI complex I/O)	RW	0x0
29:28	PLL_PWR_STATUS	Status of the power control of the DSI PLL Control module 0x0: DSI PLL Control module in OFF state 0x1: DSI PLL Control module in ON state for PLL only (HSDIVISER is OFF) 0x2: DSI PLL Control module in ON state for both PLL and HSDIVISER 0x3: DSI PLL Control module in ON state for both PLL and HSDIVISER (no clock output to the DSI complex I/O)	R	0x0
27:22	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00
21	LP_RX_SYNCHRO_ ENABLE	Defines if the DSI functional clock is higher or lower than 30 MHz. The information is used to define synchronization to be used for RxValidEsc. 0x0: The DSI functional clock is equal or slower than 30 MHz. The synchronization is falling/rising. 0x1: The DSI functional clock is higher than 30 MHz. The synchronization is rising/rising.	RW	0x0

Bits	Field Name	Description	Type	Reset
20	LP_CLK_ENABLE	Controls the gating of the TXCLKESC clock. 0x0: Disabled. The clock is not generated. The value of LP_CLK_DIVISOR[12:0] bit field is not used and does not have to be programmed. 0x1: Enabled. The clock is generated. The value of LP_CLK_DIVISOR[12:0] bit field is used and must be programmed.	RW	0x0
19	HS_MANUAL_STOP_CTRL	In case HS_AUTO_STOP_ENABLE bit is set to 0 (reset value), the bit field allows manual control of the assertion/deassertion of the signal DSIStopClk by users. 0x0: DSIStopClk deassertion unconditionally. 0x1: DSIStopClk assertion unconditionally.	RW	0x0
18	HS_AUTO_STOP_ENABLE	Enables the automatic assertion/deassertion of DSIStopClk signal. 0x0: Auto mode disabled. 0x1: Auto mode enabled.	RW	0x0
17:16	LP_CLK_NULL_PACKET_SIZE	Indicates the size of LP NULL Packets to be sent automatically when after the last LP packet transfer. It is used by the receiver to drain its internal pipeline. The valid values are from 0 to 3 bytes for the payload size.	RW	0x0
15	LP_CLK_NULL_PACKET_ENABLE	Enables the generation of NULL packet in low speed. 0x0: Disabled. The NULL packet is not sent in LP mode after the last LP packet. 0x1: Enabled. The NULL packet is sent in LP mode after the last LP packet.	RW	0x0
14	CIO_CLK_ICG	Controls the signal for gating the L3_ICLK clock provided to the complex I/O 0x0: Disabled. The L3_ICLK clock to the DSI complex I/O is gated. 0x1: Enabled. The L3_ICLK clock to the DSI complex I/O is not gated.	RW	0x0
13	DDR_CLK_ALWAYS_ON	Defines if the DDR clock is also sent when there is no HS packets sent to the peripheral (low power mode). So TXRequest for the clock lane is not de-asserted. 0x0: Disabled. The DDR clock is only provided when HS packets are sent. 0x1: Enabled. The DDR clock is always sent to the peripheral regardless of the state of the data lanes (HS or LP mode).	RW	0x0
12:0	LP_CLK_DIVISOR	Defines the ratio to be used for the generation of the low-power mode clock from DSI functional clock. The supported values are from 1 to 8191(the value 0 is invalid). The output frequency must be in the range between 20 MHz and 32 kHz.	RW	0x0001

Table 12-402. Register Call Summary for Register DSI_CLK_CTRL

Display Subsystem Environment • Data/Clock Configuration: [0] [1]
Display Subsystem Functional Description • Clock Requirements: [2] [3] [4] [5] • Extra LP Transitions: [6] [7] [8] • Power Management: [9] • DSI-PLL Power FSM: [10] [11] [12] [13] [14] [15]
Display Subsystem Basic Programming Model • Entering ULPS: [16] • Turn-Around Request in Transmit Mode: [17]

Table 12-402. Register Call Summary for Register DSI_CLK_CTRL (continued)

Display Subsystem Use Cases and Tips • Set Up DSI DPLL: [18] [19] [20] • Set Up DSI Control Registers: [21] • Configure DSI PLL: [22] [23] [24] [25] [26] [27] [28] • Configure DSI Protocol Engine: [29]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [30] • DSI Protocol Engine Registers: [31] [32]

Table 12-403. DSI_TIMING1

Address Offset	0x0000 0058		
Physical Address	0x4804 FC58	**Instance**	DSI_PROTOCOL_ENGINE
Description	TIMING1 REGISTER This register controls the DSI Protocol Engine module timers. Any bit field can be modified while DSI_CTRL.IF_EN is set to 1. It is used to indicate the number of DSI_FCLK clock cycles for the timers FORCE_TX_STOP_TIMER and TA_TO_TIMER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA_TO	TA_TO_X16	TA_TO_X8	TA_TO_COUNTER													FORCE_TX_STOP_MODE_IO	STOP_STATE_X16_IO	STOP_STATE_X4_IO	STOP_STATE_COUNTER_IO												

Bits	Field Name	Description	Type	Reset
31	TA_TO	Enables the turn-around timer 0x0: Turn-around counter is disabled. 0x1: Turn-around counter is enabled (required to receive TA interrupt in case the turn-around procedure is not successful).	RW	0x0
30	TA_TO_X16	Multiplication factor for the number of DSI_FCLK clock cycles defined in TA_TO_COUNTER bit field 0x0: The number of DSI_FCLK clock cycles defined in TA_TO_COUNTER is multiplied by 1x 0x1: The number of DSI_FCLK clock cycles defined in TA_TO_COUNTER is multiplied by 16x	RW	0x1
29	TA_TO_X8	Multiplication factor for the number of DSI_FCLK clock cycles defined in TA_TO_COUNTER bit field 0x0: The number of DSI_FCLK clock cycles defined in TA_TO_COUNTER is multiplied by 1x 0x1: The number of DSI_FCLK clock cycles defined in TA_TO_COUNTER is multiplied by 8x	RW	0x1
28:16	TA_TO_COUNTER	Turn around counter. It indicates the number of DSI_FCLK clock cycles to wait for the change of the Direction PPI signal according to the TurnRequest signal The value is from 0 to 8191.	RW	0x1FFF

Bits	Field Name	Description	Type	Reset
15	FORCE_TX_STOP_MODE_IO	Control of ForceTxStopMode signal 0x0: Deassertion of ForceTxStopMode. The hardware reset the bit at the end of the ForceTXStopMode assertion. The SW can reset the bit to stop the assertion of the ForceTXStopMode signal prior to the completion of the period. 0x1: Assertion of ForceTxStopMode	RW	0x0
14	STOP_STATE_X16_IO	Multiplication factor for the number of DSI_FCLK clock cycles defined in STOP_STATE_COUNTER_IO bit field 0x0: The number of DSI_FCLK clock cycles defined in STOP_STATE _COUNTER_IO is multiplied by 1x 0x1: The number of DSI_FCLK clock cycles defined in STOP_STATE _COUNTER_IO is multiplied by 16x	RW	0x1
13	STOP_STATE_X4_IO	Multiplication factor for the number of DSI_FCLK clock cycles defined in STOP_STATE_COUNTER_IO bit field 0x0: The number of DSI_FCLK clock cycles defined in STOP_STATE _COUNTER_IO is multiplied by 1x 0x1: The number of DSI_FCLK clock cycles defined in STOP_STATE _COUNTER_IO is multiplied by 4x	RW	0x1
12:0	STOP_STATE_COUNTER_IO	Stop state counter. It indicates the number of DSI_FCLK clock cycles to assert ForceTXStopMode signal. The value is from 0 to 8191.	RW	0x1FFF

Table 12-404. Register Call Summary for Register DSI_TIMING1

Display Subsystem Functional Description
- ForceTxStopMode FSM: [0] [1] [2] [3] [4]
- TurnRequest FSM: [5] [6] [7] [8] [9]
- LP RX Timer: [10]

Display Subsystem Basic Programming Model
- Video Mode Transfer: [11] [12]
- Command Mode Transfer Example 1: [13] [14]
- Command Mode Transfer Example 2: [15] [16]

Display Subsystem Use Cases and Tips
- Configure DSI Timing and Virtual Channels: [17]
- Configure DSI Protocol Engine: [18] [19] [20] [21]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [22]

Table 12-405. DSI_TIMING2

Address Offset	0x0000 005C		
Physical Address	0x4804 FC5C	**Instance**	DSI_PROTOCOL_ENGINE
Description	TIMING2 REGISTER This register controls the DSI Protocol Engine module timers. Any bit field can be modified while DSI_CTRL.IF_EN is set to 1. It is used to indicate the number of TxByteClkHS clock cycles for the timers HS_TX_TIMER and LP_RX_TIMER		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HS_TX_TO	HS_TX_TO_X16	HS_TX_TO_X8	HS_TX_TO_COUNTER													LP_RX_TO	LP_RX_TO_X16	LP_RX_TO_X4	LP_RX_TO_COUNTER												

Bits	Field Name	Description	Type	Reset
31	HS_TX_TO	Enables the HS TX timer. 0x0: Turn-around counter is disabled. 0x1: Turn-around counter is enabled (required to receive TA interrupt in case the turn-around procedure is not successful).	RW	0x0
30	HS_TX_TO_X16	Multiplication factor for the number of TxByteClkHS functional clock cycles defined in HS_TX_COUNTER bit field 0x0: The number of TxByteClkHS functional clock cycles defined in HS_TX_TO_COUNTER is multiplied by 1x 0x1: The number of TxByteClkHS functional clock cycles defined in HS_TX_TO_COUNTER is multiplied by 16x	RW	0x1
29	HS_TX_TO_X8	Multiplication factor for the number of TxByteClkHS functional clock cycles defined in HS_TX_COUNTER bit 0x0: The number of TxByteClkHS functional clock cycles defined in HS_TX_TO_COUNTER is multiplied by 1x 0x1: The number of TxByteClkHS functional clock cycles defined in HS_TX_TO_COUNTER is multiplied by 8x	RW	0x1
28:16	HS_TX_TO_ COUNTER	HS_TX_TIMER counter. It indicates the number of TxByteClkHS function clock cycles for the HS TX timer. The value is from 0 to 8191.	RW	0x1FFF
15	LP_RX_TO	Enables the LP RX timer. 0x0: Turn-around counter is disabled. 0x1: Turn-around counter is enabled (required to receive TA interrupt in case the turn-around procedure is not successful).	RW	0x0
14	LP_RX_TO_X16	Multiplication factor for the number of DSI_FCLK clock cycles defined in LP_RX_COUNTER bit field 0x0: The number of DSI_FCLK clock cycles defined in LP_RX_TO_COUNTER is multiplied by 1x 0x1: The number of DSI_FCLK clock cycles defined in LP_RX_TO_COUNTER is multiplied by 16x	RW	0x1
13	LP_RX_TO_X4	Multiplication factor for the number of DSI_FCLK clock cycles defined in LP_RX_COUNTER bit 0x0: The number of DSI_FCLK clock cycles defined in LP_RX_TO_COUNTER is multiplied by 1x 0x1: The number of DSI_FCLK clock cycles defined in LP_RX_TO_COUNTER is multiplied by 4x	RW	0x1
12:0	LP_RX_TO_ COUNTER	LP_RX_TIMER counter. It indicates the number of DSI_FCLK clock cycles for the LP RX timer. The value is from 0 to 8191.	RW	0x1FFF

Table 12-406. Register Call Summary for Register DSI_TIMING2

Display Subsystem Functional Description
- HS TX Timer: [0]
- LP RX Timer: [1] [2] [3] [4] [5]

Display Subsystem Use Cases and Tips
- Configure DSI Timing and Virtual Channels: [6]
- Configure DSI Protocol Engine: [7] [8] [9] [10] [11] [12]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [13]

Table 12-407. DSI_VM_TIMING1

Address Offset	0x0000 0060		
Physical Address	0x4804 FC60	**Instance**	DSI_PROTOCOL_ENGINE
Description	VIDEO MODE TIMING REGISTER This register defines the video mode timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSA								HFP								HBP															

Bits	Field Name	Description	Type	Reset
31:24	HSA	Defines the horizontal Sync active period used in video mode in number of byte clock cycles (TxByteClkHS) The supported values are from 0 to 255.	RW	0x00
23:12	HFP	Defines the horizontal front porch used in video mode in number of byte clock cycles (TxByteClkHS) The supported values are from 0 to 255	RW	0x000
11:0	HBP	Defines the horizontal back porch used in video mode in number of byte clock cycles (TxByteClkHS) The supported values are from 0 to 255	RW	0x000

Table 12-408. Register Call Summary for Register DSI_VM_TIMING1

Display Subsystem Basic Programming Model
- Video Mode: [0]
- Video Mode Transfer: [1]

Display Subsystem Use Cases and Tips
- Configure DSI Timing and Virtual Channels: [2]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [3]

Table 12-409. DSI_VM_TIMING2

Address Offset	0x0000 0064		
Physical Address	0x4804 FC64	**Instance**	DSI_PROTOCOL_ENGINE
Description	VIDEO MODE TIMING REGISTER This register defines the video mode timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				WINDOW_SYNC				VSA								VFP								VBP							

Bits	Field Name	Description	Type	Reset
31:28	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
27:24	WINDOW_SYNC	Number of TxByteClkHS clock cycles for the synchronization window. An interrupt for synchronization lost is generated when the received synchronization on video port is not inside the window. The DSI protocol engine does not change its own timings if the synch is inside the window. The valid values are from 0 to 15.	RW	0x0
23:16	VSA	Defines the vertical Sync active period used in video mode in number of lines. The supported values are from 0 to 255 It is used to generate the short packet for End of Vertical synchronization.	RW	0x00
15:8	VFP	Defines the vertical front porch used in video mode in number of lines. The supported values are from 0 to 255	RW	0x00
7:0	VBP	Defines the vertical back porch used in video mode in number of lines. The supported values are from 0 to 255	RW	0x00

Table 12-410. Register Call Summary for Register DSI_VM_TIMING2

Display Subsystem Environment • Video Port Used for Video Mode: [0]
Display Subsystem Basic Programming Model • Video Mode: [1] [2]
Display Subsystem Use Cases and Tips • Configure DSI Timing and Virtual Channels: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-411. DSI_VM_TIMING3

Address Offset	0x0000 0068		
Physical Address	0x4804 FC68	**Instance**	DSI_PROTOCOL_ENGINE
Description	VIDEO MODE TIMING REGISTER This register defines the video mode timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TL																VACT															

Bits	Field Name	Description	Type	Reset
31:16	TL	Defines the number of length of the line in video mode in number of byte clock cycles (TxByteClkHS) The supported values are from 0 to 8192. The values from 8193 to 65535 are not supported.	RW	0x0000
15:0	VACT	Defines the number of active lines used in video mode. The supported values are from 0 to 65535	RW	0x0000

Table 12-412. Register Call Summary for Register DSI_VM_TIMING3

Display Subsystem Environment • Video Port Used for Video Mode: [0]
Display Subsystem Basic Programming Model • Video Mode: [1]
Display Subsystem Use Cases and Tips • Configure DSI Timing and Virtual Channels: [2]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [3]

Table 12-413. DSI_CLK_TIMING

Address Offset	0x0000 006C		
Physical Address	0x4804 FC6C	**Instance**	DSI_PROTOCOL_ENGINE
Description	CLOCK TIMING REGISTER This register controls the DSI Protocol Engine module timers. This register should not be modified while DSI_CTRL.IF_EN is set to 1.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																DDR_CLK_PRE								DDR_CLK_POST							

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0000
15:8	DDR_CLK_PRE	Indicates the number of TxByteClkHS cycles between the start of the DDR clock and the assertion of the data request signal. The values from 1 to 255 are valid. The value 0 is reserved. The value is not used if DSI_CLK_CTRL[13] DDR_CLK_ALWAYS_ON is set to 1 since the DDR clock is always present.	RW	0x01
7:0	DDR_CLK_POST	Indicates the number of TxByteClkHS cycles after the deassertion of the data request signal and the stop of the DDR clock. The values from 1 to 255 are valid. The value 0 is reserved. The value is not used if DSI_CLK_CTRL[13] DDR_CLK_ALWAYS_ON is set to 1 since the DDR clock is always present.	RW	0x01

Table 12-414. Register Call Summary for Register DSI_CLK_TIMING

Display Subsystem Functional Description • Timing Parameters for an LP to HS Transaction: [0] • Timing Parameters for an HS to LP Transaction: [1]
Display Subsystem Use Cases and Tips • Configure DSI Timing and Virtual Channels: [2] • Configure DSI Protocol Engine: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-415. DSI_TX_FIFO_VC_SIZE

Address Offset	0x0000 0070		
Physical Address	0x4804 FC70	**Instance**	DSI_PROTOCOL_ENGINE
Description	Defines the corresponding memory entries allocated for each VC. The VC must be disabled to allocate/un-allocate some entries in the TX FIFO.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VC3_FIFO_SIZE				RESERVED	VC3_FIFO_ADD			VC2_FIFO_SIZE				RESERVED	VC2_FIFO_ADD			VC1_FIFO_SIZE				RESERVED	VC1_FIFO_ADD			VC0_FIFO_SIZE				RESERVED	VC0_FIFO_ADD		

Bits	Field Name	Description	Type	Reset
31:28	VC3_FIFO_SIZE	Size of the FIFO allocated for VC 3. For a complete description, refer to Table 12-64.	RW	0x0
27	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
26:24	VC3_FIFO_ADD	Address of the space allocated in the FIFO for VC 3. For a complete description, refer to Table 12-65.	RW	0x0
23:20	VC2_FIFO_SIZE	Size of the FIFO allocated for VC 2. For a complete description, refer to Table 12-64.	RW	0x0
19	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
18:16	VC2_FIFO_ADD	Address of the space allocated in the FIFO for VC 2. For a complete description, refer to Table 12-65.	RW	0x0
15:12	VC1_FIFO_SIZE	Size of the FIFO allocated for VC 1. For a complete description, refer to Table 12-64.	RW	0x0
11	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
10:8	VC1_FIFO_ADD	Address of the space allocated in the FIFO for VC 1.For a complete description, refer to Table 12-65.	RW	0x0
7:4	VC0_FIFO_SIZE	Size of the FIFO allocated for VC 0. For a complete description, refer to Table 12-64.	RW	0x0
3	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
2:0	VC0_FIFO_ADD	Address of the space allocated in the FIFO for VC 0. For a complete description, refer to Table 12-65.	RW	0x0

Table 12-416. Register Call Summary for Register DSI_TX_FIFO_VC_SIZE

Display Subsystem Basic Programming Model
- Command Mode TX FIFO: [0] [1] [2] [3] [4]

Display Subsystem Use Cases and Tips
- Configure DSI Protocol Engine: [5]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [6]

Table 12-417. DSI_RX_FIFO_VC_SIZE

Address Offset	0x0000 0074		
Physical Address	0x4804 FC74	**Instance**	DSI_PROTOCOL_ENGINE
Description	Defines the corresponding memory entries allocated for each VC and the addresses. The VC must be disabled to allocate/un-allocate some entries in the RX FIFO.		
Type	RW		

31 30 29 28	27	26 25 24	23 22 21 20	19	18 17 16	15 14 13 12	11	10 9 8	7 6 5 4	3	2 1 0
VC3_FIFO_SIZE	RESERVED	VC3_FIFO_ADD	VC2_FIFO_SIZE	RESERVED	VC2_FIFO_ADD	VC1_FIFO_SIZE	RESERVED	VC1_FIFO_ADD	VC0_FIFO_SIZE	RESERVED	VC0_FIFO_ADD

Bits	Field Name	Description	Type	Reset
31:28	VC3_FIFO_SIZE	Size of the FIFO allocated for VC 3. For a complete description, refer to Table 12-64.	RW	0x0
27	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
26:24	VC3_FIFO_ADD	Address of the space allocated in the FIFO for VC 3.For a complete description, refer to Table 12-65.	RW	0x0
23:20	VC2_FIFO_SIZE	Size of the FIFO allocated for VC 2. For a complete description, refer to Table 12-64.	RW	0x0
19	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
18:16	VC2_FIFO_ADD	Address of the space allocated in the FIFO for VC 2. For a complete description, refer to Table 12-65.	RW	0x0
15:12	VC1_FIFO_SIZE	Size of the FIFO allocated for VC 1. For a complete description, refer to Table 12-64.	RW	0x0
11	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
10:8	VC1_FIFO_ADD	Address of the space allocated in the FIFO for VC 1. For a complete description, refer to Table 12-65.	RW	0x0
7:4	VC0_FIFO_SIZE	Size of the FIFO allocated for VC 0. For a complete description, refer to Table 12-64.	RW	0x0

Bits	Field Name	Description	Type	Reset
3	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
2:0	VC0_FIFO_ADD	Address of the space allocated in the FIFO for VC 0. For a complete description, refer to Table 12-65.	RW	0x0

Table 12-418. Register Call Summary for Register DSI_RX_FIFO_VC_SIZE

Display Subsystem Basic Programming Model • Command Mode RX FIFO: [0] [1] [2]
Display Subsystem Use Cases and Tips • Configure DSI Protocol Engine: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-419. DSI_COMPLEXIO_CFG2

Address Offset	0x0000 0078		
Physical Address	0x4804 FC78	**Instance**	DSI_PROTOCOL_ENGINE
Description	COMPLEXIO CONFIGURATION REGISTER for the complex I/O This register contains the lane configuration for the ULPS for each lane.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														LP_BUSY	HS_BUSY	RESERVED						RESERVED	RESERVED	LANE3_ULPS_SIG2	LANE2_ULPS_SIG2	LANE1_ULPS_SIG2	RESERVED	RESERVED	LANE3_ULPS_SIG1	LANE2_ULPS_SIG1	LANE1_ULPS_SIG1

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0000
17	LP_BUSY	Indicates when there are still pending operations for VCs configured for LP mode. Forced to 1 when at least one VC is enabled and configured for LP mode. Read 0x0: LP logic is idle Read 0x1: LP logic is active	R	0x0
16	HS_BUSY	Indicates when there are still pending operations for VCs configured for HS mode. Forced to 1 when at least one VC is enabled and configured for HS mode Read 0x0: HS logic is idle Read 0x1: HS logic is active	R	0x0
15:10	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00
9	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
8	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0

Bits	Field Name	Description	Type	Reset
7	LANE3_ULPS_SIG2	Enables the ULPS for the lane #3. The HW should change the state of the lane to ULPS only when it is in stop state and there is no data pending inside the DSI protocol engine and the DSI protocol engine has control of the bus (BTA has not been sent). The state of the signal TxRequestEsc is change if lane #3 is a data lane. The state of the signal TxUlpsClk is change if lane #3 is a clock lane. There will be a latency depending on the frequency of TxClkExc. This bit should be read back to confirm a write has been effective. 0x0: READ: Inactive state effective WRITE: Request to change to inactive state 0x1: READ: Active state effective WRITE: Change request to active. If the lane is a data lane, TxRequestEsc is asserted and synchronously TxUlpsEsc is asserted for one period of TxClkEsc.	RW	0x0
6	LANE2_ULPS_SIG2	Enables the ULPS for the lane #2. The HW should change the state of the lane to ULPS only when it is in stop state and there is no data pending inside the DSI protocol engine and the DSI protocol engine has control of the bus (BTA has not been sent). The state of the signal TxRequestEsc is change if lane #2 is a data lane. The state of the signal TxUlpsClk is change if lane #2 is a clock lane. There will be a latency depending on the frequency of TxClkExc. This bit should be read back to confirm a write has been effective. 0x0: READ: Inactive state effective WRITE: Request to change to inactive state 0x1: READ: ACTIVE state effective WRITE: Change request to active. If the lane is a data lane, TxRequestEsc is asserted and synchronously TxUlpsEsc is asserted for one period of TxClkEsc.	RW	0x0
5	LANE1_ULPS_SIG2	Enables the ULPS for the lane #1. The HW should change the state of the lane to ULPS only when it is in stop state and there is no data pending inside the DSI protocol engine and the DSI protocol engine has control of the bus (BTA has not been sent). The state of the signal TxRequestEsc is change if lane #1 is a data lane. The state of the signal TxUlpsClk is change if lane #1 is a clock lane. There will be a latency depending on the frequency of TxClkExc. This bit should be read back to confirm a write has been effective. 0x0: READ: Inactive state effective WRITE: Request to change to inactive state 0x1: READ: ACTIVE state effective WRITE: Change request to active. If the lane is a data lane, TxRequestEsc is asserted and synchronously TxUlpsEsc is asserted for one period of TxClkEsc.	RW	0x0
4	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
3	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
2	LANE3_ULPS_SIG1	Enables the ULPS for the lane #3. The HW should change the state of the lane to ULPS only when it is in stop state and there is no data pending inside the DSI protocol engine and the DSI protocol engine has control of the bus (BTA has not been sent). The state of the signal TxULPSExit is changed if lane #3 is a clock lane. There will be a latency depending on the frequency of TxClkExc. This bit should be read back to confirm a write has been effective. 0x0: READ: Inactive state effective WRITE: Request to change to inactive state 0x1: READ: ACTIVE state effective WRITE: Change request to active. If the lane is a data lane, TxRequestEsc is asserted and synchronously TxUlpsEsc is asserted for one period of TxClkEsc.	RW	0x0

Bits	Field Name	Description	Type	Reset
1	LANE2_ULPS_SIG1	Enables the ULPS for the lane #2. The HW must change the state of the lane to ULPS only when it is in stop state and there is no data pending inside the DSI protocol engine and the DSI protocol engine has control of the bus (BTA has not been sent). The state of the signal TxULPSExit is changed if lane #3 is a clock lane. There will be a latency depending on the frequency of TxClkExc. This bit should be read back to confirm a write has been effective. 0x0: READ: Inactive state effective WRITE: Request to change to inactive state 0x1: READ: ACTIVE state effective WRITE: Change request to active. If the lane is a data lane, TxRequestEsc is asserted and synchronously TxUlpsEsc is asserted for one period of TxClkEsc.	RW	0x0
0	LANE1_ULPS_SIG1	Enables the ULPS for the lane #1. The HW must change the state of the lane to ULPS only when it is in stop state and there is no data pending inside the DSI protocol engine and the DSI protocol engine has control of the bus (BTA has not been sent). The state of the signal TxULPSExit is changed if lane #3 is a clock lane. There will be a latency depending on the frequency of TxClkExc. This bit should be read back to confirm a write has been effective. 0x0: READ: Inactive state effective WRITE: Request to change to inactive state 0x1: READ: ACTIVE state effective WRITE: Change request to active. If the lane is a data lane, TxRequestEsc is asserted and synchronously TxUlpsEsc is asserted for one period of TxClkEsc.	RW	0x0

Table 12-420. Register Call Summary for Register DSI_COMPLEXIO_CFG2

Display Subsystem Environment
- ULPS: [0]

Display Subsystem Basic Programming Model
- Ultra-Low Power State: [1]
- Entering ULPS: [2] [3] [4] [5] [6] [7] [8] [9]
- Exiting ULPS: [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [28]

Table 12-421. DSI_RX_FIFO_VC_FULLNESS

Address Offset	0x0000 007C		
Physical Address	0x4804 FC7C	**Instance**	DSI_PROTOCOL_ENGINE
Description	Defines the fullness of each space allocated for each VC.		
Type	R		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
VC3_FIFO_FULLNESS	VC2_FIFO_FULLNESS	VC1_FIFO_FULLNESS	VC0_FIFO_FULLNESS

Bits	Field Name	Description	Type	Reset
31:24	VC3_FIFO_FULLNESS	Fullness of the FIFO allocated for VC 3.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00
23:16	VC2_FIFO_FULLNESS	Fullness of the FIFO allocated for VC 2.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00
15:8	VC1_FIFO_FULLNESS	Fullness of the FIFO allocated for VC 1.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00
7:0	VC0_FIFO_FULLNESS	Fullness of the FIFO allocated for VC 0.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00

Table 12-422. Register Call Summary for Register DSI_RX_FIFO_VC_FULLNESS

Display Subsystem Basic Programming Model • Command Mode DMA Requests: [0]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [1]

Table 12-423. DSI_VM_TIMING4

Address Offset	0x0000 0080		
Physical Address	0x4804 FC80	**Instance**	DSI_PROTOCOL_ENGINE
Description	VIDEO MODE TIMING REGISTER This register defines the video mode timing.		
Type	RW		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
RESERVED	HSA_HS_INTERLEAVING	HFP_HS_INTERLEAVING	HBP_HS_INTERLEAVING

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00
23:16	HSA_HS_INTERLEAVING	Defines the number of TxByteClkHS cycles that can be used for interleaving high-speed command mode packet into Video Mode stream during HSA blanking period. The supported values are from 0 to 255.	RW	0x00
15:8	HFP_HS_INTERLEAVING	Defines the number of TxByteClkHS cycles that can be used for interleaving high-speed command mode packet into Video Mode stream during HFP blanking period. The supported values are from 0 to 255	RW	0x00
7:0	HBP_HS_INTERLEAVING	Defines the number of TxByteClkHS cycles that can be used for interleaving high-speed command mode packet into Video Mode stream during HBP blanking period. The supported values are from 0 to 255	RW	0x00

Table 12-424. Register Call Summary for Register DSI_VM_TIMING4

Display Subsystem Functional Description • HS Command Mode Interleaving Programming Model: [0] [1] [2]
Display Subsystem Basic Programming Model • Video Mode: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-425. DSI_TX_FIFO_VC_EMPTINESS

Address Offset	0x0000 0084		
Physical Address	0x4804 FC84	**Instance**	DSI_PROTOCOL_ENGINE
Description	Defines the emptiness of each space allocated for each VC.		
Type	R		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
VC3_FIFO_EMPTINESS	VC2_FIFO_EMPTINESS	VC1_FIFO_EMPTINESS	VC0_FIFO_EMPTINESS

Bits	Field Name	Description	Type	Reset
31:24	VC3_FIFO_EMPTINESS	Emptiness of the FIFO allocated for VC 3.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00
23:16	VC2_FIFO_EMPTINESS	Emptiness of the FIFO allocated for VC 2.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00
15:8	VC1_FIFO_EMPTINESS	Emptiness of the FIFO allocated for VC 1.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00
7:0	VC0_FIFO_EMPTINESS	Emptiness of the FIFO allocated for VC 0.The valid values are from 0 to 127 corresponding to 1x33-bit,...up to 128x33-bit.	R	0x00

Table 12-426. Register Call Summary for Register DSI_TX_FIFO_VC_EMPTINESS

Display Subsystem Basic Programming Model • Command Mode TX FIFO: [0] [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-427. DSI_VM_TIMING5

Address Offset	0x0000 0088		
Physical Address	0x4804 FC88	**Instance**	DSI_PROTOCOL_ENGINE
Description	VIDEO MODE TIMING REGISTER This register defines the video mode timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								HSA_LP_INTERLEAVING								HFP_LP_INTERLEAVING								HBP_LP_INTERLEAVING							

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00
23:16	HSA_LP_INTERLEAVING	Defines the number of bytes for Low Power command mode packets that can be sent on PPI link during HSA blanking period. The supported values are from 0 to 255.	RW	0x00
15:8	HFP_LP_INTERLEAVING	Defines the number of bytes for Low Power command mode packets that can be sent on PPI link during HFP blanking period. The supported values are from 0 to 255	RW	0x00
7:0	HBP_LP_INTERLEAVING	Defines the number of bytes for Low Power command mode packets that can be sent on PPI link during HBP blanking period. The supported values are from 0 to 255	RW	0x00

Table 12-428. Register Call Summary for Register DSI_VM_TIMING5

Display Subsystem Functional Description • LP Command Mode Interleaving Programming Model: [0] [1] [2]
Display Subsystem Basic Programming Model • Video Mode: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-429. DSI_VM_TIMING6

Address Offset	0x0000 008C		
Physical Address	0x4804 FC8C	**Instance**	DSI_PROTOCOL_ENGINE
Description	VIDEO MODE TIMING REGISTER This register defines the video mode timing.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BL_HS_INTERLEAVING																BL_LP_INTERLEAVING															

Bits	Field Name	Description	Type	Reset
31:16	BL_HS_INTERLEAVING	Defines the number of TxByteClkHS clock cycles that can be used for interleaving high-speed command mode packet into Video Mode stream during blanking periods during VSA, VBP, VFP periods inside one video frame on PPI link. The supported values are from 0 to 65535.	RW	0x0000
15:0	BL_LP_INTERLEAVING	Defines the maximum number of bytes for Low Power command mode packets that can be sent on PPI link during blanking periods during VSA, VBP or VFP periods inside one video frame on PPI link. The supported values are from 0 to 65535	RW	0x0000

Table 12-430. Register Call Summary for Register DSI_VM_TIMING6

Display Subsystem Environment • Video Port Used for Video Mode: [0]
Display Subsystem Functional Description • HS Command Mode Interleaving Programming Model: [1] • LP Command Mode Interleaving Programming Model: [2]
Display Subsystem Basic Programming Model • Video Mode: [3]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [4]

Table 12-431. DSI_VM_TIMING7

Address Offset	0x0000 0090		
Physical Address	0x4804 FC90	**Instance**	DSI_PROTOCOL_ENGINE
Description	Defines the maximum number of bytes of Low Power command mode packets that can be sent on PPI link during blanking periods during VSA, VBP or VFP periods inside one video frame on PPI link. The supported values are from 0 to 65535		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENTER_HS_MODE_LATENCY																EXIT_HS_MODE_LATENCY															

Bits	Field Name	Description	Type	Reset
31:16	ENTER_HS_MODE_ LATENCY	Defines the number of TxByteClkHS clock cycles necessary for entering to HS mode. It corresponds to the delay in number of HS clock cycles from assertion of TxRequestHS signal to 1 until assertion of TxReadyHS signal to 1. The supported values are from 0 to 65535 .	RW	0x0000

Bits	Field Name	Description	Type	Reset
15:0	EXIT_HS_MODE_ LATENCY	Defines the number of TxByteClkHS clock cycles necessary for exiting from HS mode. It corresponds to the maximum delay in number of TxByteClkHS clock cycles from deassertion of TxRequestHS signal until PPI link is in LP-11 state from which a new entrance to HS mode can be initiated which does not take more than ENTER_HS_MODE_LATENCY clock cycles. The supported values are from 0 to 65535	RW	0x0000

Table 12-432. Register Call Summary for Register DSI_VM_TIMING7

Display Subsystem Functional Description • Timing Parameters for an LP to HS Transaction: [0] • Timing Parameters for an HS to LP Transaction: [1]
Display Subsystem Basic Programming Model • Video Mode: [2] • Video Mode Transfer: [3]
Display Subsystem Use Cases and Tips • Configure DSI Timing and Virtual Channels: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

Table 12-433. DSI_STOPCLK_TIMING

Address Offset	0x0000 0094		
Physical Address	0x4804 FC94	**Instance**	DSI_PROTOCOL_ENGINE
Description	Number of functional clock cycles to wait for TxByteClock to stop/start after change in DSIStopClk signal		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DSI_STOPCLK_LATENCY							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Write 0s for future compatibility. Reads returns 0.	R	0x000000
7:0	DSI_STOPCLK_ LATENCY	Clock gating latency from DSI Protocol engine to TxByteClkHS	RW	0x80

Table 12-434. Register Call Summary for Register DSI_STOPCLK_TIMING

Display Subsystem Functional Description • DSI-PLL Power FSM: [0] [1]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [2]

Table 12-435. DSI_VCn_CTRL

Address Offset	0x0000 0100+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD00+ (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	CONTROL REGISTER - Virtual channel This register controls the VC.		
Type	RW		

31 30	29 28 27	26 25 24	23 22 21 20	19 18 17 16	15	14	13 12 11 10	9 8	7	6	5	4	3	2	1	0
RESERVED	DMA_RX_REQ_NB	DMA_RX_THRESHOLD	DMA_TX_REQ_NB	DMA_TX_THRESHOLD	RX_FIFO_NOT_EMPTY	TX_FIFO_FULL	VC_BUSY	PP_BUSY	RESERVED	MODE_SPEED	ECC_TX_EN	CS_TX_EN	BTA_EN	TX_FIFO_NOT_EMPTY	MODE	BTA_LONG_EN	BTA_SHORT_EN	SOURCE	VC_EN

Bits	Field Name	Description	Type	Reset
31:30	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0
29:27	DMA_RX_REQ_NB	Selection of the use of the DMA request (associated to the RX FIFO) 0x0: DSI_DMA_REQ0 is selected 0x1: DSI_DMA_REQ1 is selected 0x2: DSI_DMA_REQ2 is selected 0x3: DSI_DMA_REQ3 is selected 0x4: No DMA req selected	RW	0x0
26:24	DMA_RX_THRESHOLD	Defines the threshold value for the DMA request (associated to the RX FIFO) 0x0: 1x 32 bits 0x1: 2 x 32 bits 0x2: 4 x 32 bits 0x3: 8 x 32 bits 0x4: 16 x 32 bits 0x5: 32 x 32 bits	RW	0x0
23:21	DMA_TX_REQ_NB	Selection of the use of the DMA request (associated to the TX FIFO) 0x0: DSI_DMA_REQ0 is selected 0x1: DSI_DMA_REQ1 is selected 0x2: DSI_DMA_REQ2 is selected 0x3: DSI_DMA_REQ3 is selected 0x4: No DMA req selected	RW	0x0
20	RX_FIFO_NOT_EMPTY	FIFO status in command mode. Otherwise, this bit can be ignored. 0x0: The RX FIFO is empty (the FIFO does not contain any data for the VC) 0x1: The RX FIFO is not empty (the FIFO contains at least one byte for the VC)	R	0x0
19:17	DMA_TX_THRESHOLD	Defines the threshold value for the DMA request (associated to the TX FIFO) 0x0: 1x 32 bits 0x1: 2 x 32 bits 0x2: 4 x 32 bits 0x3: 8 x 32 bits 0x4: 16 x 32 bits 0x5: 32 x 32 bits	RW	0x0
16	TX_FIFO_FULL	FIFO status in command mode. Otherwise, this bit can be ignored. 0x0: The TX FIFO is not full (the FIFO can accept at least one more 32-bit value) 0x1: The TX FIFO is full	R	0x0

Bits	Field Name	Description	Type	Reset
15	VC_BUSY	Indicates if previously scheduled activities (packets, BTA) are still being processed. Forced to 1 by hardware if VC is enabled. Software should check this bit is 0 before changing channel configuration. 0x0: No pending operations for this VC 0x1: Pending operations for this VC	R	0x0
14	PP_BUSY	Line buffer busy status. 0x0: Software is permitted to write a new header for VP command mode traffic. 0x1: Software is NOT permitted to write a new header for VP command mode traffic.	R	0
13:10	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x00
9	MODE_SPEED	Selection of the mode. This bit is ignored by hardware when video mode is selected. 0x0: Low-power mode (CMOS) is used to send short and long packets to the peripheral. 0x1: High speed mode (SLVS) is used to send short and long packets to the peripheral.	RW	0x0
8	ECC_TX_EN	Enables the ECC generation for the transmit header (short and long packets). 0x0: Disabled 0x1: Enabled	RW	0x0
7	CS_TX_EN	Enables the checksum generation for the transmit payload (long packet only). 0x0: Disabled. The value 0x00 is used. 0x1: Enabled. The Check-sum value is calculated by HW.	RW	0x0
6	BTA_EN	Send the bus turn around to the peripheral. It can be used when the automatic mode is enabled (BTA_SHORT_EN=1 or/and BTA_LONG_EN=1). In that case only one BTA is sent to the peripheral. The manual mode allows users to define for which packets, the turn around is required for example getting acknowledge from the peripheral. 0x0: READS: BTA generation is completed. It is reset by HW when it is completed. WRITES: Cancellation of the BTA generation (not guarantee since it could already on going, must not be used). 0x1: READS: BTA generation has been requested by user (it could be on going but not completed). WRITES: Request for BTA generation.	RW	0x0
5	TX_FIFO_NOT_EMPTY	FIFO status 0x0: The TX FIFO is empty (the FIFO does not contain any data for the VC) 0x1: The TX FIFO is not empty (the FIFO contains at least one byte for the VC)	R	0x0
4	MODE	Selection of the mode 0x0: Command mode. 0x1: Video mode.	RW	0x0
3	BTA_LONG_EN	Enables the automatic bus turn-around after completion of each long packet transmission. 0x0: Disabled 0x1: Enabled	RW	0x0
2	BTA_SHORT_EN	Enables the automatic bus turn-around after completion of each short packet transmission. 0x0: Disabled 0x1: Enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
1	SOURCE	Selection of the source between L4 interconnect slave port and video port. This bit is ignored by hardware when video mode is selected. 0x0: All the data are provided by the L4 interconnect slave port. Any transfer on the video port is ignored for this VC. 0x1: If MODE=VIDEO_MODE. any data received on the video port (pixels and enabled synchronization events using DSI_CTRL[17] VP_HSYNC_START, DSI_CTRL[18] VP_HSYNC_END, DSI_CTRL[15] VP_VSYNC_START, DSI_CTRL[16] VP_VSYNC_END,) are sent on the VC (only one VC can be associated with the video port, the software must ensure that no more than one VC is enabled with the video port as the main source for data). If MODE=COMMAND_MODE, the VP_STALL signal is used by the protocol engine to indicate when new data are required. The synchronization signals are not generated by the display controller. Regardless of the MODE, no data can be provided on the L4 interconnect slave port.	RW	0x0
0	VC_EN	Enables the VC. 0x0: Disabled. The VC must be disabled for any register change in the DSI_VCn_XXX registers the corresponding VC ID. 0x1: Enabled. No change is allowed to the VC registers (except for setting the bit fields/registers: DSI_VCn_CTRL[6] BTA_EN, DSI_VCn_TE[15:0] TE_SIZE, DSI_VCn_TE[31] TE_START, DSI_VCn_LONG_XXX, DSI_VCn_SHORT_XXX, DSI_VCn_IRQXXX registers).	RW	0x0

Table 12-436. Register Call Summary for Register DSI_VCn_CTRL

Display Subsystem Environment
- Blanking: [0] [1] [2] [3]

Display Subsystem Functional Description
- Command Mode: [4] [5] [6]
- DSI-PLL Power FSM: [7] [8]
- TurnRequest FSM: [9]
- LP RX Timer: [10]
- Bus Turnaround: [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27]
- Tearing Effect: [28] [29] [30]
- ECC Generation: [31]
- Checksum Generation for Long Packet Payloads: [32]

Display Subsystem Basic Programming Model
- Global Register Controls: [33]
- Virtual Channels: [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44]
- Packets: [45] [46]
- Video Mode: [47] [48] [49] [50] [51] [52] [53]
- Command Mode TX FIFO: [54] [55] [56] [57] [58]
- Command Mode RX FIFO: [59]
- Command Mode DMA Requests: [60] [61] [62] [63] [64] [65]
- DSI Programming Sequence Example: [66] [67]
- Video Mode Transfer: [68] [69]
- Command Mode Transfer Example 1: [70] [71] [72] [73] [74]
- Command Mode Transfer Example 2: [75] [76] [77] [78] [79]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [80]
- DSI Protocol Engine Registers: [81] [82]

Table 12-437. DSI_VCn_TE

Address Offset	0x0000 0104+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD04+ (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	CONTROL REGISTER - Virtual channel This register controls the tearing effect logic. It defines the size of the transfer when TE occurs and enables the automatic TE mode.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TE_START	TE_EN	RESERVED						TE_SIZE																							

Bits	Field Name	Description	Type	Reset
31	TE_START	Manual control of the start of the transfer. Users can use the TE interrupt to determine that the TE trigger has been received before setting the TE_START bit field. It is not mandatory to use the TE interrupt. 0x0: Indicates the end of the transfer. The bit can be used to cancel the transfer if not already started. The FIFO must be flushed by software to ensure it contains no remaining data. 0x1: Starts the transfer of the data. The size is defined in TE_SIZE. The bit field is set until the transfer completes. It is reset by hardware when the transfer completes.	RW	0
30	TE_EN	Tearing effect control 0x0: Disables the automatic transfer of the data using the TE trigger as a synchronization event. The interruption is used to know when the TE trigger is received. The hardware resets the bit field when the transfer completes(TE_SIZE=0). 0x1: Enables the automatic transfer of the data using the TE trigger as a synchronization event.	RW	0
29:24	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x0000
23:0	TE_SIZE	Defines the number of bytes (payload data excluding the check-sum) to be sent. Users must perform the write into the DSI_VCn_LONG_PACKET_HEADER register before sending data from the DSI_VCn_LONG_PACKET_PAYLOAD register. The register value is decremented for every byte of the DSI link that is sent. At the end of the transfer (TE_SIZE = 0), the TE_EN bit field is reset by hardware. The DMA_request is asserted when the trigger is received in order to receive data in the TX FIFO. It must not be used until all data (TE_SIZE) have been received in the FIFO.	RW	0x000000

Table 12-438. Register Call Summary for Register DSI_VCn_TE

Display Subsystem Functional Description
- Tearing Effect: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [14]
- DSI Protocol Engine Registers: [15] [16]

Table 12-439. DSI_VCn_LONG_PACKET_HEADER

Address Offset	0x0000 0108+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD08+ (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	LONG PACKET HEADER INFORMATION–virtual channel. This register sets the 32-bit DATA_ID + Word count + ECC. The ECC is computed if ECC_TX_EN is set to 1. DATA_ID is located at bit[7:0]. WC is located at bit[23:8]. ECC is located at bit[31:24] (least-significant byte first and least significant bit first).		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEADER																															

Bits	Field Name	Description	Type	Reset
31:0	HEADER	Packet header information: DATA ID + DATA FIELD +ECC	W	0x00000000

Table 12-440. Register Call Summary for Register DSI_VCn_LONG_PACKET_HEADER

Display Subsystem Environment
- Video Port Used on Command Mode: [0] [1]
- Virtual Channel ID - VC Field, DI[7:6]: [2]

Display Subsystem Functional Description
- Video Mode: [3]
- Command Mode: [4] [5] [6] [7] [8] [9]
- Bus Turnaround: [10]
- Tearing Effect: [11] [12] [13]

Display Subsystem Basic Programming Model
- Global Register Controls: [14]
- Virtual Channels: [15]
- Packets: [16] [17] [18] [19] [20] [21]
- Video Mode: [22] [23]
- Command Mode TX FIFO: [24] [25] [26] [27] [28] [29]
- Command Mode RX FIFO: [30]
- Command Mode DMA Requests: [31]
- Command Mode Transfer Example 1: [32] [33] [34]
- Command Mode Transfer Example 2: [35]

Display Subsystem Register Manual
- Display Subsystem Register Manual: [36]
- Display Subsystem Register Mapping Summary: [37]
- DSI Protocol Engine Registers: [38] [39]

Table 12-441. DSI_VCn_LONG_PACKET_PAYLOAD

Address Offset	0x0000 010C+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD0C+ (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	LONG PACKET PAYLOAD INFORMATION–virtual channel. This register sets the payload information (excluding Check-sum). Hardware must capture the word count in the packet header (in DSI_VCn_LONG_PACKET_HEADER register) to determine the last valid data (the VC ID can be different from VC). Byte1 is bit[7:0]; Byte2 is bit[15:8]; Byte3 is bit[23:16]; Byte4 is bit[31:24]; and Byten is sent before Byten+1 (least-significant byte first and least significant bit first).		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAYLOAD																															

Bits	Field Name	Description	Type	Reset
31:0	PAYLOAD	Packet payload information (excluding check-sum)	W	0x00000000

Table 12-442. Register Call Summary for Register DSI_VCn_LONG_PACKET_PAYLOAD

Display Subsystem Functional Description • Command Mode: [0] [1] [2] [3] [4] • Bus Turnaround: [5] • Tearing Effect: [6] [7]
Display Subsystem Basic Programming Model • Global Register Controls: [8] • Packets: [9] [10] [11] • Command Mode TX FIFO: [12] [13] [14] • Command Mode RX FIFO: [15] • Command Mode Transfer Example 1: [16] [17]
Display Subsystem Register Manual • Display Subsystem Register Manual: [18] • Display Subsystem Register Mapping Summary: [19] • DSI Protocol Engine Registers: [20]

Table 12-443. DSI_VCn_SHORT_PACKET_HEADER

Address Offset	0x0000 0110+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD10+ (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	SHORT PACKET HEADER INFORMATION -Virtual channel This register sets the 24-bit DATA_ID + Short packet data field + ECC (the VC ID can be different than VC) DATA_ID is located at bit[7:0] short packet data field is located at bit[23:8] ECC is located at bit[31:24] (least-significant byte first and least significant bit first)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEADER																															

Bits	Field Name	Description	Type	Reset
31:0	HEADER	WRITES: Packet header information: DATA ID + DATA FIELD +ECC written into the TX FIFO READS: 32-bit values read from the RX FIFO	RW	0x00000000

Table 12-444. Register Call Summary for Register DSI_VCn_SHORT_PACKET_HEADER

Display Subsystem Environment • Virtual Channel ID - VC Field, DI[7:6]: [0]
Display Subsystem Functional Description • Command Mode: [1] [2] [3] • Tearing Effect: [4]
Display Subsystem Basic Programming Model • Global Register Controls: [5] • Virtual Channels: [6] • Packets: [7] • Command Mode TX FIFO: [8] [9] [10] • Command Mode RX FIFO: [11] • Command Mode DMA Requests: [12] • Command Mode Transfer Example 1: [13] [14]

Table 12-444. Register Call Summary for Register DSI_VCn_SHORT_PACKET_HEADER (continued)

Display Subsystem Register Manual

- Display Subsystem Register Manual: [15]
- Display Subsystem Register Mapping Summary: [16]

Table 12-445. DSI_VCn_IRQSTATUS

Address Offset	0x0000 0118+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD18+ (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	INTERRUPT STATUS REGISTER - Virtual channel This register regroups all the events related to the VC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							PP_BUSY_CHANGE_IRQ	FIFO_TX_UDF_IRQ	ECC_NO_CORRECTION_IRQ	BTA_IRQ	FIFO_RX_OVF_IRQ	FIFO_TX_OVF_IRQ	PACKET_SENT_IRQ	ECC_CORRECTION_IRQ	CS_IRQ

Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x000000
8	PP_BUSY_CHANGE_IRQ	Video port ping-pong buffer busy status. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0
7	FIFO_TX_UDF_IRQ	FIFO underflow status. The FIFO used on the slave port for buffering the data received on the OCP slave port for the VC has underflowed which means that the data for the current packet have not been received in time since the transfer of the packet are already started (transfer started since the packet size is bigger than space allocated in the FIFO). 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
6	ECC_NO_CORRECTION_IRQ	ECC error status (short and long packets). No correction of the header because of more than 1-bit error. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
5	BTA_IRQ	Virtual channel - BTA status. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Bits	Field Name	Description	Type	Reset
4	FIFO_RX_OVF_IRQ	FIFO overflow error status. The FIFO used on the slave port for buffering the data received on the DSI link for the VC has overflowed. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
3	FIFO_TX_OVF_IRQ	FIFO overflow error status. The FIFO used on the slave port for buffering the data received on the OCP slave port for the VC has overflowed. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
2	PACKET_SENT_IRQ	Indicates that a packet has been sent. It is used when BTA manual mode is used. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
1	ECC_CORRECTION_IRQ	Virtual channel - ECC has been used to do the correction of the only 1-bit error status (short and long packet only). 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0
0	CS_IRQ	Virtual channel - Check-Sum mismatch status. 0x0: READS: Event is false. WRITES: Status bit unchanged. 0x1: READS: Event is true (pending). WRITES: Status bit is reset.	RW	0x0

Table 12-446. Register Call Summary for Register DSI_VCn_IRQSTATUS

Display Subsystem Integration • DSI Interrupt Request: [0]
Display Subsystem Functional Description • Command Mode: [1] • Bus Turnaround: [2] [3]
Display Subsystem Basic Programming Model • Interrupts: [4] • Command Mode TX FIFO: [5] [6] • Command Mode DMA Requests: [7] • Command Mode Transfer Example 1: [8] • Command Mode Transfer Example 2: [9]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [10]

Table 12-447. DSI_VCn_IRQENABLE

Address Offset	0x0000 011C+ (n* 0x20)	**Index**	n = 0 to 3
Physical Address	0x4804 FD1C + (n* 0x20)	**Instance**	DSI_PROTOCOL_ENGINE
Description	INTERRUPT ENABLE REGISTER - Virtual channel This register regroups all the events related to VC.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							PP_BUSY_CHANGE_IRQ_EN	FIFO_TX_UDF_IRQ_EN	ECC_NO_CORRECTION_IRQ_EN	BTA_IRQ_EN	FIFO_RX_OVF_IRQ_EN	FIFO_TX_OVF_IRQ_EN	PACKET_SENT_IRQ_EN	ECC_CORRECTION_IRQ_EN	CS_IRQ_EN

Bits	Field Name	Description	Type	Reset
31:9	RESERVED	Write 0s for future compatibility. Reads returns 0.	RW	0x000000
8	PP_BUSY_CHANGE_IRQ_EN	Video port ping-pong buffer busy. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0
7	FIFO_TX_UDF_IRQ_EN	FIFO underflow enable. The FIFO used on the slave port for buffering the data received on the OCP slave port for the VC has underflowed which means that the data for the current packet have not been received in time since the transfer of the packet are already started (transfer started since the packet size is bigger than space allocated in the FIFO). 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
6	ECC_NO_CORRECTION_IRQ_EN	ECC error (short and long packets). No correction of the header because of more than 1-bit error. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
5	BTA_IRQ_EN	Virtual channel -Bus turn around reception 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
4	FIFO_RX_OVF_IRQ_EN	FIFO overflow enable. The FIFO used on the slave port for buffering the data received on the DSI link for the VC has overflowed. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
3	FIFO_TX_OVF_IRQ_EN	FIFO overflow enable. The FIFO used on the slave port for buffering the data received on the OCP slave port for the VC has overflowed. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
2	PACKET_SENT_IRQ_EN	Indicates that a packet has been sent. It is used when BTA manual mode is used. 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0
1	ECC_CORRECTION_IRQ_EN	Virtual channel - ECC has been used to correct the only 1-bit error (short and long packet). 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0

Bits	Field Name	Description	Type	Reset
0	CS_IRQ_EN	Virtual channel - Check-Sum of the payload mismatch detection 0x0: Event is masked 0x1: Event generates an interrupt when it occurs	RW	0x0

Table 12-448. Register Call Summary for Register DSI_VCn_IRQENABLE

Display Subsystem Functional Description • Command Mode: [0] • Bus Turnaround: [1]
Display Subsystem Basic Programming Model • Interrupts: [2] • Command Mode Transfer Example 1: [3] • Command Mode Transfer Example 2: [4]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [5]

12.7.2.6 DSI complex I/O Registers

Table 12-449. DSI_PHY_CFG0

Address Offset	0x0000 0000		
Physical Address	0x4804 FE00	**Instance**	DSI_PHY_SCP
Description	Configuration register for HS mode timings		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THS_PREPARE								THS_PREPARE_THS_ZERO								THS_TRAIL								THS_EXIT							

Bits	Field Name	Description	Type	Reset
31:24	THS_PREPARE	THS-PREPARE timing parameter in multiples of DDR clock period. DDR clock = CLKIN4DDR/4. Programmed value = CEIL(70 ns/DDR clock period) + 2	RW	0x1A
23:16	THS_PREPARE_ THS_ZERO	THS-PREPARE + THS-ZERO timing parameter in multiples of DDR clock period. DDR clock = CLKIN4DDR/4. Programmed value = ceil(175 ns/DDR clock period) + 2	RW	0x3c
15:8	THS_TRAIL	THS-TRAIL timing parameter in multiples of DDR clock period. DDR clock = CLKIN4DDR/4. Programmed value = ceil(60 ns/DDR clock period) + 5	RW	0x1A
7:0	THS_EXIT	Ths-exit timing parameter in multiples of DDR clock period. DDR clock = CLKIN4DDR/4). Programmed value = ceil(145 ns/DDR clock period).	RW	0x28

Table 12-450. Register Call Summary for Register DSI_PHY_CFG0

Display Subsystem Functional Description • Timing Parameters for an LP to HS Transaction: [0] [1] [2] • Timing Parameters for an HS to LP Transaction: [3] [4] • Shadowing Register: [5]
Display Subsystem Basic Programming Model • High-Speed Clock Transmission: [6] [7] • High-Speed Data Transmission: [8] [9] [10]
Display Subsystem Use Cases and Tips • Configure DSI_PHY: [11] [12] [13] [14] • Configure DSI_PHY Timing: [15] [16] [17] [18]

Table 12-450. Register Call Summary for Register DSI_PHY_CFG0 (continued)

Display Subsystem Register Manual

- Display Subsystem Register Mapping Summary: [19]

Table 12-451. DSI_PHY_CFG1

Address Offset	0x0000 0004		
Physical Address	0x4804 FE04	**Instance**	DSI_PHY_SCP
Description	Configuration register for LP mode and HS mode timings		
Type	RW		

31 30 29	28 27	26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
TTA_GO	TTA_SURE	TTA_GET	RESERVED	TLPX_HALF	TCLK_TRAIL	TCLK_ZERO

Bits	Field Name	Description	Type	Reset
31:29	TTA_GO	TTA-GO timing in terms of number of TXCLKESC clocks 0x0: 2 cycles 0x1: 3 cycles 0x2: 4 cycles 0x3: 5 cycles 0x4: 6 cycles 0x5: 7 cycles 0x6: 8 cycles 0x7: 9 cycles Default value: 4 cycles	RW	0x2
28:27	TTA_SURE	TTA-SURE timing in terms of number of TXCLKESC clocks 0x0: 2 cycles 0x1: 1 cycle 0x2: 3 cycles 0x3: 4 cycles Default value: 2 cycles	RW	0x0
26:24	TTA_GET	TTA-GET timing in terms of number of TXCLKESC clocks 0x0: 3 cycles 0x1: 4 cycles 0x2: 5 cycles 0x3: 6 cycles 0x4: 7 cycles 0x5: 8 cycles 0x6: 9 cycles 0x7: 10 cycles Default value: 5 cycles	RW	0x2
23:21	RESERVED	Reserved. Read returns zero. Write only zero for future compatibility	RW	0x0
20:16	TLPX_HALF	(TLPX)/2 timing parameter in multiples of DDR clock frequency. DDR clock = CLKIN4DDR/4. Programmed value = ceil(25ns/DDR clock period) Default value : 10 (for 400 MHz) Note: TLPX is used to define the length of LP-01 state in HS Start of Transmission sequences on clock and data lanes. For all other purposes TLPX is defined by the period of TxLPEsc clock.	RW	0x0A

Bits	Field Name	Description	Type	Reset
15:8	TCLK_TRAIL	TCLK-TRAIL timing parameter in multiples of DDR clock frequency. DDR clock = CLKIN4DDR/4. Programmed value = ceil(60 ns/DDR clock period) + 2	RW	0x18
7:0	TCLK_ZERO	TCLK-ZERO timing parameter in multiples of DDR clock period. DDR clock = CLKIN4DDR/4. Programmed value = ceil(260 ns/DDR clock period)	RW	0x75

Table 12-452. Register Call Summary for Register DSI_PHY_CFG1

Display Subsystem Functional Description • Timing Parameters for an LP to HS Transaction: [0] [1] [2] • Timing Parameters for an HS to LP Transaction: [3] • Shadowing Register: [4]
Display Subsystem Basic Programming Model • High-Speed Clock Transmission: [5] [6] [7] • High-Speed Data Transmission: [8] • Turn-Around Request in Transmit Mode: [9] • Turn-Around Request in Receive Mode: [10]
Display Subsystem Use Cases and Tips • Configure DSI_PHY: [11] [12] [13] [14] • Configure DSI_PHY Timing: [15] [16] [17] [18]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [19]

Table 12-453. DSI_PHY_CFG2

Address Offset	0x0000 0008		
Physical Address	0x4804 FE08	**Instance**	DSI_PHY_SCP
Description	Sync pattern and reserved bits		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HS_SYNC								RESERVED																TCLK_PREPARE							

Bits	Field Name	Description	Type	Reset
31:24	HS_SYNC	HS sync pattern. Default = 184 (0xB8)	RW	0xB8
23:8	RESERVED	Reserved. Do not modify the reset value.	RW	0xB80000
7:0	TCLK_PREPARE	TCLK-PREPARE timing parameter in multiples of DDR clock period. DDR clock = CLKIN4DDR/4. Programmed value = ceil(65 ns/DDR clock period)	RW	0x1B

Table 12-454. Register Call Summary for Register DSI_PHY_CFG2

Display Subsystem Functional Description • Timing Parameters for an LP to HS Transaction: [0]
Display Subsystem Basic Programming Model • High-Speed Data Transmission: [1]
Display Subsystem Use Cases and Tips • Configure DSI_PHY: [2] [3] • Configure DSI_PHY Timing: [4] [5]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [6]

Table 12-455. DSI_PHY_CFG3

Address Offset	0x0000 000C		
Physical Address	0x4804 FE0C	**Instance**	DSI_PHY_SCP
Description	TX trigger patterns		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTRIGGERESC3								TXTRIGGERESC2								TXTRIGGERESC1								TXTRIGGERESC0							

Bits	Field Name	Description	Type	Reset
31:24	TXTRIGGERESC3	Transmitted pattern when TXTRIGGERESC3 is asserted (first bit transmitted to last bit transmitted) Default: 01100010	RW	0x62
23:16	TXTRIGGERESC2	Default: 01011101	RW	0x5D
15:8	TXTRIGGERESC1	Default: 00100001	RW	0x21
7:0	TXTRIGGERESC0	Default: 10100000	RW	0xA0

Table 12-456. Register Call Summary for Register DSI_PHY_CFG3

Display Subsystem Functional Description
- PHY Triggers: [0]
- Reset: [1]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [2]

Table 12-457. DSI_PHY_CFG4

Address Offset	0x0000 0010		
Physical Address	0x4804 FE10	**Instance**	DSI_PHY_SCP
Description	RX trigger patterns		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXTRIGGERESC3								RXTRIGGERESC2								RXTRIGGERESC1								RXTRIGGERESC0							

Bits	Field Name	Description	Type	Reset
31:24	RXTRIGGERESC3	Received pattern for which RXTRIGGERESC3 is asserted (first bit transmitted to last bit transmitted) Default: 01100010	RW	0x62
23:16	RXTRIGGERESC2	Default: 01011101	RW	0x5D
15:8	RXTRIGGERESC1	Default: 00100001	RW	0x21
7:0	RXTRIGGERESC0	Default: 10100000	RW	0xA0

Table 12-458. Register Call Summary for Register DSI_PHY_CFG4

Display Subsystem Functional Description
- PHY Triggers: [0]
- Tearing Effect: [1]
- Acknowledge: [2]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [3]

Table 12-459. DSI_PHY_CFG5

Address Offset	0x0000 0014		
Physical Address	0x4804 FE14	**Instance**	DSI_PHY_SCP
Description	Reset done bits		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESETDONETXBYTECLK	RESETDONESCPCLK	RESETDONEPWRCLK	RESETDONETXCLKESC0	RESETDONETXCLKESC1	RESETDONETXCLKESC2	RESERVED																									

Bits	Field Name	Description	Type	Reset
31	RESETDONETXBYTECLK	RESETDONETXBYTECLK 0x0: No reset 0x1: Reset done for the TXBYTECLK domain	R	0x0
30	RESETDONESCPCLK	RESETDONESCPCLK 0x0: No reset 0x1: Reset done for the SCP clock domain	R	0x0
29	RESETDONEPWRCLK	RESETDONEPWRCLK 0x0: No reset 0x1: Reset done for the PWR clock domain	R	0x0
28	RESETDONETXCLKESC0	RESETDONETXCLKESC0 0x0: No reset 0x1: Reset done for the TXCLKESC domain for lane 0	R	0x0
27	RESETDONETXCLKESC1	RESETDONETXCLKESC1 0x0: No reset 0x1: Reset done for the TXCLKESC domain for lane 1	R	0x0
26	RESETDONETXCLKESC2	RESETDONETXCLKESC2 0x0: No reset 0x1: Reset done for the TXCLKESC domain for lane 2	R	0x0
25:0	RESERVED	Read-Only register. Read returns zero	R	0x0000000

Table 12-460. Register Call Summary for Register DSI_PHY_CFG5

Display Subsystem Basic Programming Model • Reset-Done Bits: [0] [1] [2] [3] [4] [5] [6]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [7]

12.7.2.7 DSI PLL Control Module Registers

Table 12-461. DSI_PLL_CONTROL

Address Offset	0x0000 0000		
Physical Address	0x4804 FF00	**Instance**	DSI_PLL_CTRL_SCP
Description	This register controls the PLL reset/power and modes		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											DSI_HSDIV_SYSRESET	DSI_PLL_SYSRESET	DSI_PLL_HALTMODE	DSI_PLL_GATEMODE	DSI_PLL_AUTOMODE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Reserved. Write only zero for future compatibility. Reads return zero.	R	0x0000000
4	DSI_HSDIV_SYSRESET	Force HSDIVIDER SYSRESET 0x0: HSDIVIDER SYSRESET controlled by power FSM 0x1: HSDIVIDER SYSRESET forced active	RW	0x0
3	DSI_PLL_SYSRESET	Force ADPLLV2 SYSRESET 0x0: PLL SYSRESET controlled by power FSM 0x1: PLL SYSRESET forced active	RW	0x0
2	DSI_PLL_HALTMODE	Allow PLL to be halted if no activity 0x0: PLL will not be halted 0x1: PLL will be halted based on activity	RW	0x0
1	DSI_PLL_GATEMODE	Allow PLL clock gating for power saving 0x0: CLKIN4DDR on 0x1: CLKIN4DDR gated by DSI Protocol Engine activity	RW	0x0
0	DSI_PLL_AUTOMODE	Automatic update mode. If this bit is set then the configuration updates will be synchronized to DISPC_UPDATE_SYNC. If this bit is clear configuration updates will be done immediately. 0x0: Manual mode 0x1: Automatic mode	RW	0x0

Table 12-462. Register Call Summary for Register DSI_PLL_CONTROL

Display Subsystem Functional Description
- DSI-PLL Power FSM: [0] [1]

Display Subsystem Basic Programming Model
- DSI PLL Go Sequence: [2] [3] [4] [5]

Display Subsystem Use Cases and Tips
- Set Up DSI DPLL: [6]
- Configure DSI PLL: [7]

Display Subsystem Register Manual
- Display Subsystem Register Mapping Summary: [8]

Table 12-463. DSI_PLL_STATUS

Address Offset	0x0000 0004		
Physical Address	0x4804 FF04	**Instance**	DSI_PLL_CTRL_SCP
Description	This register contains the status information		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						DSI_BYPASSACKZ	DSIPROTO_CLOCK_ACK	DSS_CLOCK_ACK	DSI_PLL_BYPASS	DSI_PLL_HIGHJITTER	DSI_PLL_LIMP	DSI_PLL_LOSSREF	DSI_PLL_RECAL	DSI_PLL_LOCK	DSI_PLLCTRL_RESET_DONE

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Reserved. Reads return zero.	R	0x000000
9	DSI_BYPASSACKZ	State of bypass mode on PHY and HSDIVIDER 0x0: DSI_PHY and HSDIVIDER have switched to using the bypass clocks. 0x1: PLL outputs are still being used by DSI_PHY or HSDIVIDER	R	0x0
8	DSIPROTO_ CLOCK_ACK	Acknowledge for enable of DSI Protocol Engine clock Verify the status before selecting this source in the DSI Protocol Engine clock mux 0x0: DSI Protocol Engine clock inactive 0x1: DSI Protocol Engine clock active	R	0x0
7	DSS_CLOCK_ACK	Acknowledge for enable of DSS clock Verify the status before selecting this source in the DSS clock multiplexer 0x0: DSS clock inactive 0x1: DSS clock active	R	0x0
6	DSI_PLL_BYPASS	DSI PLL Bypass status 0x0: PLL not bypassing 0x1: PLL bypass	R	0x0
5	DSI_PLL_HIGHJITTER	DSI PLL High Jitter status 0x0: PLL in normal jitter condition 0x1: PLL in high jitter condition: Phase error > 24% (TIGHTPHASELOCK = 0) Phase error > 12% (TIGHTPHASELOCK = 1)	R	0x0
4	DSI_PLL_LIMP	DSI PLL Limp status 0x0: LIMP mode inactive 0x1: LIMP mode active	R	0x0
3	DSI_PLL_LOSSREF	DSI PLL Reference Loss status 0x0: Reference input active 0x1: Reference input inactive	R	0x0
2	DSI_PLL_RECAL	DSI PLL re-calibration status If this bit is active, the PLL must be recalibrated 0x0: Recalibration is not required 0x1: Recalibration is required	R	0x0

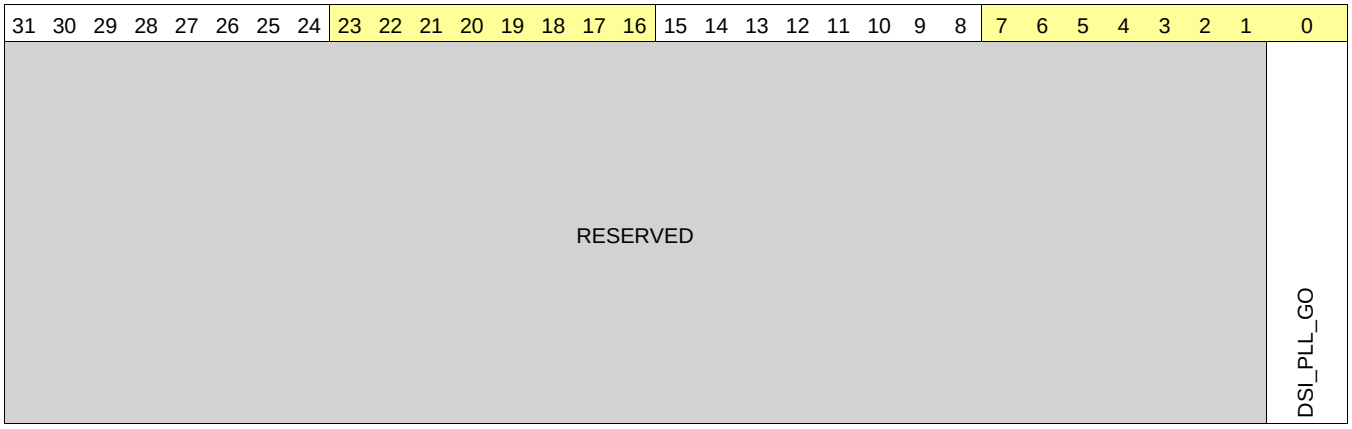
Bits	Field Name	Description	Type	Reset
1	DSI_PLL_LOCK	DSI PLL Lock status See the programming guide for the use of this bit 0x0: PLL is not locked 0x1: PLL is locked	R	0x0
0	DSI_PLLCTRL_RESET_DONE	DSI PLL Controller reset done status 0x0: Reset is in progress 0x1: Reset has completed	R	0x0

Table 12-464. Register Call Summary for Register DSI_PLL_STATUS

Display Subsystem Functional Description • Error Handling: [0] [1]
Display Subsystem Basic Programming Model • Software Reset: [2] • DSI PLL Clock Gating Sequence: [3] • DSI PLL Error Handling: [4] [5] [6] [7]
Display Subsystem Use Cases and Tips • Set Up DSI DPLL: [8] • Configure DSI PLL: [9]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [10]

Table 12-465. DSI_PLL_GO

Address Offset	0x0000 0008		
Physical Address	0x4804 FF08	**Instance**	DSI_PLL_CTRL_SCP
Description	This register contains the GO bit		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved. Write only zero for future compatibility. Reads return zero.	R	0x00000000
0	DSI_PLL_GO	Request (re-)locking sequence of the PLL. If the AutoMode bit is set, then this will be deferred until DISPC_UPDATE_SYNC goes active 0x0: No pending action 0x1: Request PLL (re-)locking/locking pending	RW	0x0

Table 12-466. Register Call Summary for Register DSI_PLL_GO

Display Subsystem Basic Programming Model • DSI PLL Go Sequence: [0] [1] • DSI PLL Clock Gating Sequence: [2] [3] • DSI PLL Recommended Values: [4]
Display Subsystem Use Cases and Tips • Set Up DSI DPLL: [5] [6] [7] [8] [9] • Configure DSI PLL: [10] [11] [12] [13] [14]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [15] • DSI PLL Control Module Registers: [16]

Table 12-467. DSI_PLL_CONFIGURATION1

Address Offset	0x0000 000C		
Physical Address	0x4804 FF0C	**Instance**	DSI_PLL_CTRL_SCP
Description	This register contains the latched PLL and HSDIVDER configuration bits		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					DSIPROTO_CLOCK_DIV			DSS_CLOCK_DIV					DSI_PLL_REGM												DSI_PLL_REGN							DSI_PLL_STOPMODE

Bits	Field Name	Description	Type	Reset
31:27	RESERVED	Reserved. Write only zero for future compatibility. Reads return zero.	R	0x00
26:23	DSIPROTO_CLOCK_DIV	Divider value for DSI Protocol Engine clock source REGM4	RW	0x0
22:19	DSS_CLOCK_DIV	Divider value for DSS clock source REGM3	RW	0x0
18:8	DSI_PLL_REGM	M Divider for PLL	RW	0x000
7:1	DSI_PLL_REGN	N Divider for PLL (Reference)	RW	0x00
0	DSI_PLL_STOPMODE	DSI PLL STOPMODE 0x0: STOPMODE is not selected 0x1: STOPMODE is selected	RW	0x0

Table 12-468. Register Call Summary for Register DSI_PLL_CONFIGURATION1

Display Subsystem Functional Description • DSI PLL Operations: [0] [1] [2]
Display Subsystem Basic Programming Model • DSI PLL Lock Sequence: [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] • DSI PLL Recommended Values: [15]
Display Subsystem Use Cases and Tips • Set Up DSI DPLL: [16] [17] [18] [19] [20] • Configure DSI PLL: [21] [22] [23] [24] [25]
Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [26]

Table 12-469. DSI_PLL_CONFIGURATION2

Address Offset	0x0000 0010		
Physical Address	0x4804 FF10	**Instance**	DSI_PLL_CTRL_SCP
Description	This register contains the unlatched PLL and HSDIVDER configuration bits These bits are "shadowed" when automatic mode is selected		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											DSI_HSDIVBYPASS	DSI_PROTO_CLOCK_PWDN	DSI_PROTO_CLOCK_EN	DSS_CLOCK_PWDN	DSS_CLOCK_EN	DSI_BYPASSEN	DSI_PHY_CLKINEN	DSI_PLL_REFEN	DSI_PLL_HIGHFREQ	DSI_PLL_CLKSEL	DSI_PLL_LOCKSEL	DSI_PLL_DRIFTGUARDEN	DSI_PLL_TIGHTPHASELOCK	DSI_PLL_LOWCURRSTBY	DSI_PLL_PLLLPMODE	DSI_PLL_FREQSEL				DSI_PLL_IDLE

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved. Write only zero for future compatibility. Reads return zero.	R	0x000
20	DSI_HSDIVBYPASS	Forces HSDIVIDER to bypass mode 0x0: HSDIVIDER in normal operation. Bypass controlled by PLL. 0x1: HSDIVIDER is forced to bypass mode.	RW	0x0
19	DSI_PROTO_CLOCK_PWDN	Power down for DSI Protocol Engine clock source 0x0: DSI Protocol Engine clock divider is active. 0x1: DSI Protocol Engine clock divider is powered down.	RW	0x0
18	DSI_PROTO_CLOCK_EN	Enable for DSI Protocol Engine clock source 0x0: DSI Protocol Engine clock divider is disabled. 0x1: DSI Protocol Engine clock divider is enabled.	RW	0x0
17	DSS_CLOCK_PWDN	Power down for DSS clock source 0x0: DSS clock divider is active. 0x1: DSS clock divider is powered down.	RW	0x0
16	DSS_CLOCK_EN	Enable for DSS clock source 0x0: DSS clock divider is disabled. 0x1: DSS clock divider is enabled.	RW	0x0
15	DSI_BYPASSEN	Selects DSS functional clock as CLKIN4DDR source 0x0: PLL controls CLKIN4DDR source: PLL DCO if PLL is locked DSS functional clock if not locked. 0x1: Force DSS functional clock to be used as CLKIN4DDR source	RW	0x0
14	DSI_PHY_CLKINEN	CLKIN4DDR control 0x0: CLKIN4DDR is disabled. 0x1:CLKIN4DDR is enabled.	RW	0x0
13	DSI_PLL_REFEN	PLL reference clock control 0x0: PLL reference clock disabled. 0x1: PLL reference clock enabled.	RW	0x0
12	DSI_PLL_HIGHFREQ	Enables a division of pixel clock by 2 before input to the PLL Required for pixel clock frequencies above 32 MHz (21 MHZ if N = 0) 0x0: Pixel clock is not divided. 0x1: Pixel clock is divided by 2.	RW	0x0

Bits	Field Name	Description	Type	Reset
11	DSI_PLL_CLKSEL	Reference clock selection 0x0: Selects DSS2_ALWON_FCLK as PLL reference clock 0x1: Selects Pixel Clock (PCLKFREE) as PLL reference clock	RW	0x0
10:9	DSI_PLL_LOCKSEL	Selects the lock criteria for the PLL 0x0: Phase Lock criteria depends on setting of DSI_PLL_TIGHTPHASELOCK bit 0x1: Frequency lock 0x2: Spare	RW	0x0
8	DSI_PLL_ DRIFTGUARDEN	DSI PLL DRIFTGUARDEN 0x0: Only RECAL flag is asserted in case of temperature drift. The programmer should take appropriate action. 0x1: Temperature drift will initiate automatic recalibration. RECAL flag will be asserted while this is taking place.	RW	0x0
7	DSI_PLL_ TIGHTPHASELOCK	DSI PLL Phase Lock criteria If this bit is set, the phase lock tolerance is reduced 0x0: Normal phase lock criteria Phase error lower than 6.4 % 0x1: Tightened phase lock criteria Phase error lower than 3.2 %	RW	0x0
6	DSI_PLL_ LOWCURRSTBY	PLL LOW CURRENT STANDBY 0x0: LOWCURRSTBY is not selected. 0x1: LOWCURRSTBY is selected.	RW	0x0
5	DSI_PLL_ PLLLPMODE	Select the power/performance of the PLL 0x0: Full performance, minimized jitter 0x1: Reduced power, increased jitter	RW	0x0
4:1	DSI_PLL_ FREQSEL	PLL internal reference frequency range selection 0x3: 0.75MHz to 1.0MHz 0x4: 1.0MHz to 1.25MHz 0x5: 1.25MHz to 1.5MHz 0x6: 1.5MHz to 1.75MHz 0x7: 1.75MHz to 2.1MHz 0xB: 7.5MHz to 10MHz 0xC: 10MHz to 12.5MHz 0xD: 12.5MHz to 15MHz 0xE: 15MHz to 17.5MHz 0xF: 17.5MHz to 21MHz	RW	0x0
0	DSI_PLL_IDLE	DSI PLL IDLE: 0x0: IDLE is not selected. 0x1: IDLE is selected.	RW	0x0

Table 12-470. Register Call Summary for Register DSI_PLL_CONFIGURATION2

Display Subsystem Integration
- Clocks: [0] [1]

Display Subsystem Functional Description
- DSI PLL Controller Architecture: [2] [3]
- DSI PLL Operations: [4] [5] [6]

Display Subsystem Basic Programming Model
- DSI PLL Go Sequence: [7] [8] [9]
- DSI PLL Lock Sequence: [10] [11] [12] [13] [14] [15]

Display Subsystem Use Cases and Tips
- Set Up DSI DPLL: [16]
- Configure DSI PLL: [17] [18] [19] [20] [21]

Table 12-470. Register Call Summary for Register DSI_PLL_CONFIGURATION2 (continued)

Display Subsystem Register Manual • Display Subsystem Register Mapping Summary: [22]

Timers

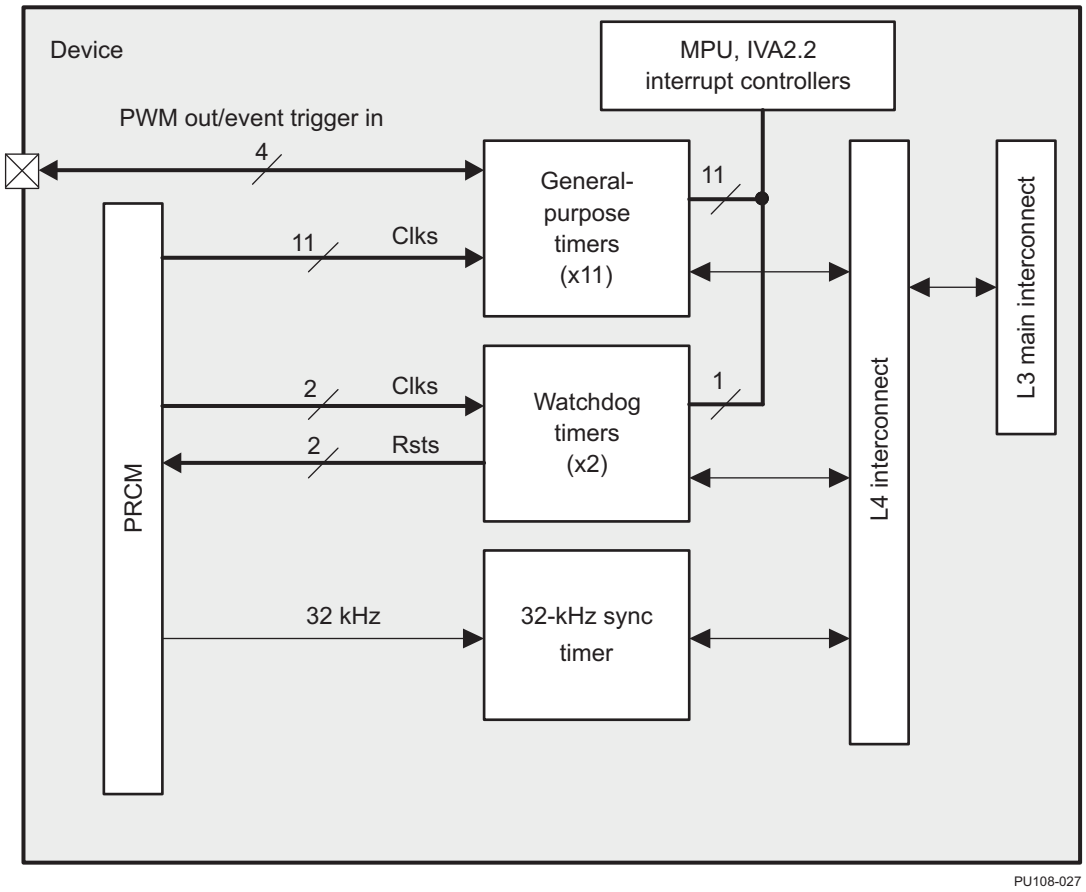
This chapter describes the timer modules.

Topic **Page**

13.1 Timers Overview

The device includes several types of timers used by the system software, including 11 general-purpose timers (GP timers), 2 watchdog timers (WDTs), a 32-kHz synchronized timer. Figure 13-1 shows the counters in the device in a high-level block diagram.

Figure 13-1. Timers



The 2 WDTs are clocked with 32-kHz clocks. The 32-kHz synchronized timer, which is reset only at power up, provides the operating system with a stable timing source that stores the relative time since the last power cycle of the product. Finally, 11 GP timers, which are useful simply as basic timers, are included to generate time-stamp-based interrupts to the system software or to use as a source of pulse-width modulation (PWM) signals.

13.2 General-Purpose Timers

13.2.1 GP Timers Overview

The device has 11 GP timers: GPTIMER1 through GPTIMER11.

Each timer can be clocked from either the system clock (12, 13, 16.8, 19.2, 26, or 38.4 MHz) or the 32-kHz clock. The selection of the clock source is made at the power, reset, clock management (PRCM) module level. For more information, see Section 13.2.3.1, *Clocking, Reset, and Power-Management Scheme*.

GPTIMER1 has its GPT1_EVENT_CAPTURE pin tied to the 32-kHz clock and can be used to gauge the system clock input; it detects its frequency among 12, 13, 16.8, 19.2, 26, or 38.4 MHz.

Each timer can provide an interrupt to the microprocessor unit (MPU) subsystem. In addition, GPTIMER5 through GPTIMER8 also have interrupts connected to the IVA2.2 subsystem.

GPTIMER1, GPTIMER2, and GPTIMER10 include specific functions to generate accurate tick interrupts to the operating system. GPTIMER8 through GPTIMER11 are connected to external pins by their PWM output or their event capture input pin (for external timer triggering). Figure 13-2 shows an overview of the GP timers.

Figure 13-2. GP Timers Overview

13.2.1.1 GP Timers Features

The following are the main features of the GP timers controllers:

- L4 slave interface support:
  - 32-bit data bus width
  - 32-/16-bit access supported
  - 8-bit access not supported
  - 10-bit address bus width
  - Burst mode not supported
  - Software-selectable nonposted or posted write/read internal resynchronization modes
- Interrupts generated on overflow, compare, and capture
- Free-running 32-bit upward counter
- Compare and capture modes
- Autoreload mode
- Start/stop mode
- Programmable divider clock source (2^n with n = [0:8])
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- Dedicated output signal for general-purpose using GPTi_GPOCFG signal
- On-the-fly read/write register (while counting)
- 1-ms tick with 32,768 Hz functional clock generated (only GPTIMER1, GPTIMER2, and GPTIMER10)

13.2.2 GP Timers Environment

13.2.2.1 GP Timers External System Interface

Four of the 11 GP timers can send or receive stimulus to/from the external (off-chip) system. In the device, however, only GPTIMER8 through GPTIMER11 are configured to output a PWM pulse or receive an external event signal used as a trigger to capture the current timer count. GPTIMER1 is also configured to receive an event trigger input (GPT1_EVENT_CAPTURE) tied to the internal 32-kHz clock. This event signal gauges the system clock input; it detects its frequency among 12, 13, 16.8, 19.2, 26, or 38.4 MHz.

Figure 13-3 shows the external system interface for the GP timers, and Table 13-1 describes the GP timer inputs and outputs.

NOTE: Software must ensure that MUX mode is configured to select the gpt_x_pwm_evt (where x = 8 to 11) signal on only one pad. Other pad(s) on which the same signal is multiplexed must be configured in safe mode or non-gptimer mode to avoid two different pads driving the same signal

For more information about the gpt_8_pwm_evt through gpt11_pwm_evt I/O pad configuration, see Chapter 6, *System Control Module*.

Figure 13-3. GP Timers External System Interface

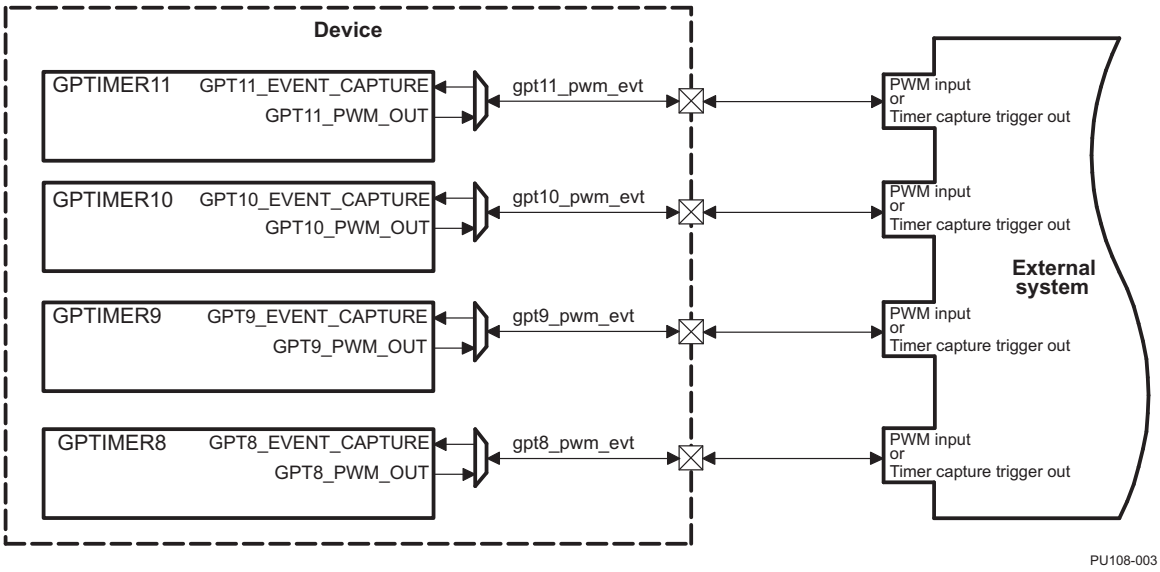


Table 13-1. Input/Output Description

Pin Name	Type [1]	Reset Value	Signal Name	Description
gpt8_pwm_evt	I/O	0	GPT8_EVENT_CAPTURE GPT8_PWM_OUT	GPTIMER8 trigger input/ PWM output
gpt9_pwm_evt	I/O	0	GPT9_EVENT_CAPTURE GPT9_PWM_OUT	GPTIMER9 trigger input/ PWM output
gpt10_pwm_evt	I/O	0	GPT10_EVENT_CAPTURE GPT10_PWM_OUT	GPTIMER10 trigger input/ PWM output
gpt11_pwm_evt	I/O	0	GPT11_EVENT_CAPTURE GPT11_PWM_OUT	GPTIMER11 trigger input/ PWM output

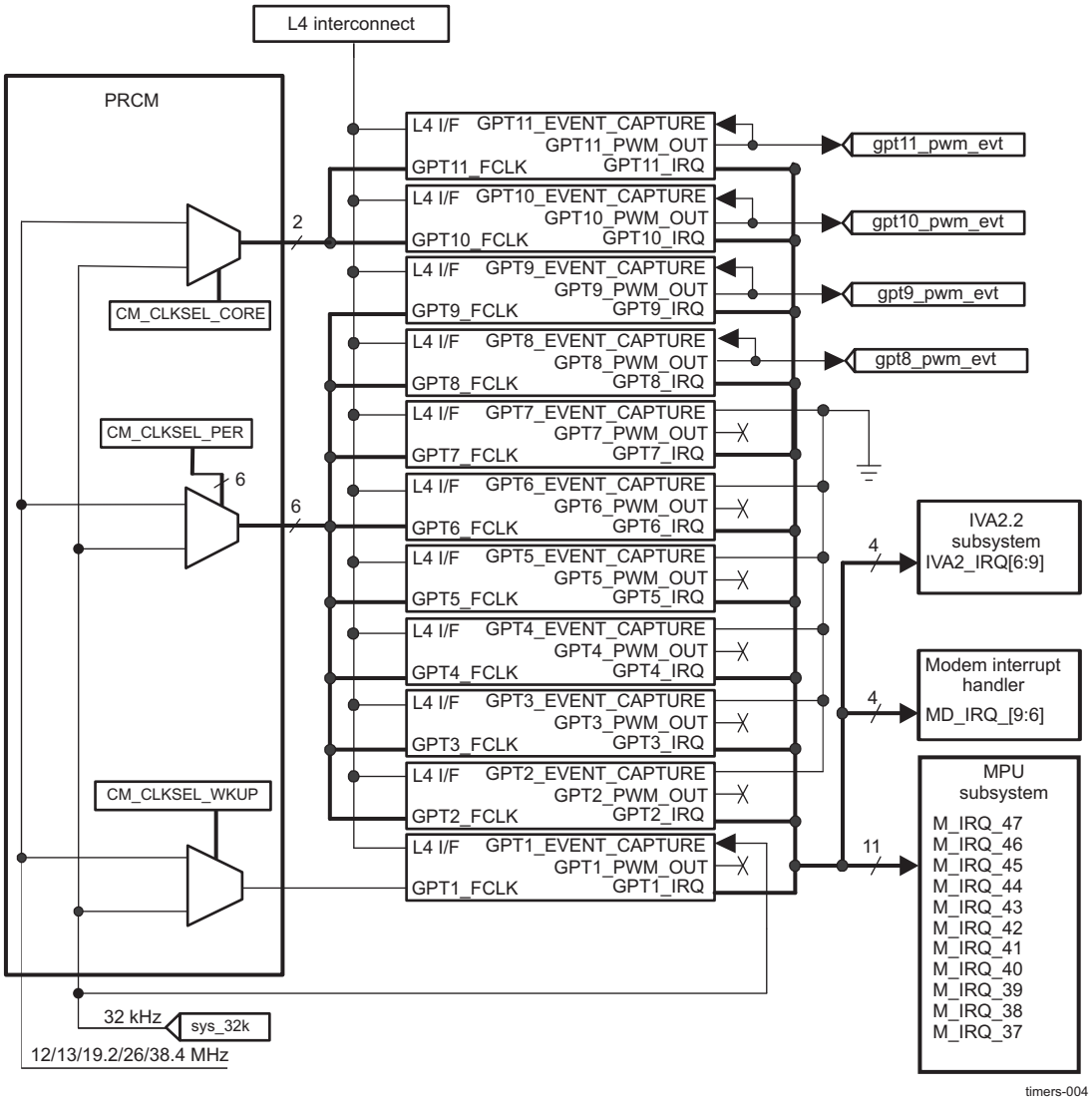
(1) When configured for that function; I = input, O = output

NOTE: The event trigger input (GPTi_EVENT_CAPTURE) for GPTIMER2 through GPTIMER7 is internally tied low, and the PWM output (GPTi_PWM_OUT) is not connected.

13.2.3 GP Timers Integration

Figure 13-4 shows the GP timer integration in the device.

Figure 13-4. GP Timer Integration



13.2.3.1 Clocking, Reset, and Power-Management Scheme

13.2.3.1.1 Clock Management

There are two clock domains in the GP timers:

- Functional clock domain: GPTi_FCLK is the GP timer functional clock. It is used to clock the GP timer internal logic.
- Interface clock domain: GPTi_ICLK is the GP timer interface clock. It is used to synchronize the GP timer L4 port to the L4 interconnect. All accesses from the interconnect are synchronous to GPTi_ICLK.

Table 13-2 lists the source clocks for each GP timer in the device. For more information on clock control and domains, see , *Power, Reset, and Clock Management*.

Table 13-2. Clock, Power, and Reset Domains for GP Timers

Timer	Interface Clock	Functional Clock	Power Domain
GPTIMER1	WKUP_L4_ICLK	GPT1_FCLK	WKUP
GPTIMER2 through GPTIMER9	PER_L4_ICLK	GPTx_ALWON_FCLK (x = 2 through 9)	PER
GPTIMER10 and 11	CORE_L4_ICLK	GPTx_FCLK (x = 10 or 11)	CORE

GP timers can be selected as the source of GPTi_FCLK (32-kHz clock or 12, 13, 16.8, 19.2, 26, or 38.4 MHz) at the PRCM level in the corresponding registers. For more information, see , *Power, Reset, and Clock Management*. Table 13-3 lists the GP timer PRCM clock selection bits.

Table 13-3. GP Timer PRCM Clock Selection Bits

Name	Associated PRCM Clock Output	Selection Bit
GPT1_FCLK	GPT1_FCLK	PRCM.CM_CLKSEL_WKUP[0] CLKSEL_GPT1
GPT[2:9]_FCLK	GPT[2:9]_ALWON_FCLK	PRCM.CM_CLKSEL_PER [0:7] CLKSEL_GPT[2:9]
GPT10_FCLK	GPT10_FCLK	PRCM.CM_CLKSEL_CORE [6] CLKSEL_GPT10
GPT11_FCLK	GPT11_FCLK	PRCM.CM_CLKSEL_CORE [7] CLKSEL_GPT11

From a global system power-management perspective, when one or both of the GP timer clocks is no longer required, the GP timers can be deactivated at the PRCM level in the corresponding registers. Table 13-4 lists the GP timer PRCM clock control bits.

Table 13-4. GP Timer PRCM Clock Control Bits

Name	Associated PRCM Clock Output	Enable Bit	Autoidle Bit
GPT1_FCLK	GPT1_FCLK	PRCM.CM_FCLKEN_WKUP[0] EN_GPT1 bit	N/A
GPT[2:9]_FCLK	GPT[2:9]_ALWON_FCLK	PRCM.CM_FCLKEN_PER [3:10] EN_GPT[2:9] bit	N/A
GPT10_FCLK	GPT10_FCLK	PRCM.CM_FCLKEN1_CORE [11] EN_GPT10 bit	N/A
GPT11_FCLK	GPT11_FCLK	PRCM.CM_FCLKEN1_CORE [12] EN_GPT11 bit	N/A
GPT1_ICLK	WKUP_L4_ICLK	PRCM.CM_ICLKEN_WKUP[0] EN_GPT1 bit	PRCM.CM_AUTOIDLE_WKUP[0] AUTO_GPT1 bit
GPT[2:9]_ICLK	PER_L4_ICLK	PRCM.CM_ICLKEN_PER[3:10] EN_GPT[2:9] bit	PRCM.CM_AUTOIDLE_PER[3:10] AUTO_GPT[2:9] bit
GPT10_ICLK	CORE_L4_ICLK	PRCM.CM_ICLKEN1_CORE[11] EN_GPT10 bit	PRCM.CM_AUTOIDLE1_CORE[11] AUTO_GPT10 bit
GPT11_ICLK		PRCM.CM_ICLKEN1_CORE[12] EN_GPT11 bit	PRCM.CM_AUTOIDLE1_CORE[12] AUTO_GPT11 bit

NOTE:

- The PRCM function clock outputs are gated at the PRCM level, assuming all the modules that share it are disabled in the corresponding register. Disabling GP timers is a necessary but not sufficient action. The clock is effectively out when all PRCM conditions are fulfilled. For the other actions to be taken, see , *Power, Reset, and Clock Management*.
- The PRCM interface clock outputs are gated at the PRCM level, assuming all the modules that share it are disabled in the corresponding register. Disabling GP timers is a necessary but not sufficient action. The clock is effectively out when all PRCM conditions are fulfilled. For the other actions to be taken, see , *Power, Reset, and Clock Management*.
- The PRCM AUTOIDLE bits are used to link/unlink the GP timers from the clock domain transitions related to the GPTi_ICLK clocks.
- For further details about source clock gating and domain transitions, see , *Power, Reset, and Clock Management.*

GP timer modules also have an internal bit, GPTi.TIOCP_CFG[0] AUTOIDLE. The GP timer AUTOIDLE bit is used to apply an internal interface clock gating strategy.

At the PRCM level, when all conditions to shut off the PRCM functional or interface output clocks are met (see , *Power, Reset, and Clock Management*, for details), the PRCM automatically launches a hardware handshake protocol to ensure the GP timer is ready to have its clocks switched off. Namely, the PRCM asserts an IDLE request to the GP timer.

Although this handshake is a hardware function and is out of software control, the way the GP timer acknowledges the PRCM IDLE request is configurable through the GPTi.TIOCP_CFG[4:3] IDLEMODE bit. Table 13-5 lists the IDLEMODE settings and the related acknowledgement modes.

Table 13-5. IDLEMODE Settings

IDLEMODE Value	Selected Mode	Description
00	Force-idle	The GP timer acknowledges unconditionally the IDLE request from the PRCM module, regardless of its internal operations. This mode must be used carefully, because it does not prevent the loss of data when the clock is switched off.
01	No-idle	The GP timer never acknowledges an IDLE request from the PRCM module. This mode is safe from a module point of view, because it ensures that the clocks remain active. It is not efficient from a power-saving perspective, however, because it does not allow the PRCM output clock to be shut off and thus the power domain to be set to a lower power state.
10	Smart-idle	The GP timer acknowledges the IDLE request, basing its decision on its internal activity. The acknowledge signal is asserted only when all pending transactions and IRQ requests are treated. This is the best approach for efficient system power management.
11	Reserved	

When configured in smart-idle mode, the GP timer also offers an additional granularity on GPTi_FCLK and GPTi_ICLK gating. The GPTi.TIOCP_CFG[9:8] CLOCKACTIVITY bit field is used to determine which clock will be shut down (GPTi_FCLK, GPTi_ICLK, neither of them, or both of them).

The CLOCKACTIVITY setting is used internally to the GP timer to determine the part of the module on which the conditions to acknowledge the PRCM IDLE request will be tested. For example, if GPTi_FCLK is not to be shut down on a PRCM IDLE request, the GP timer considers only GPTi_ICLK and the associated pending activities before acknowledging the request.

Some GP timer features are associated with GPTi_ICLK and others are associated with GPTi_FCLK. Using the CLOCKACTIVITY setting along with the smart-idle mode ensures that the features associated with the clock that will remain active are always enabled, even if the GP timer acknowledges an IDLE request.

Table 13-6 lists the CLOCKACTIVITY settings and the associated features.

Table 13-6. CLOCKACTIVITY Settings

CLOCKACTIVITY Value	GPTi_ICLK Effects	GPTi_FCLK Effects	Description	Associated Features
00	OFF	OFF	Both GPTi_ICLK and GPTi_FCLK are considered for generating the acknowledge. This setting also means both GPTi_FCLK and GPTi_ICLK are likely to be shut down on PRCM IDLE request.	The idle acknowledge signal is asserted when there are no pending activities on the functional clock domain (improved latency in assertion of idle acknowledge). The wake-up capability of the GP timer is disabled.
01	ON	OFF	GPTi_ICLK is not shut down on PRCM IDLE request; only GPTi_FCLK is affected.	
10	OFF	ON	GPTi_FCLK is not shut down on PRCM IDLE request; only GPTi_ICLK is affected.	The idle acknowledge signal is asserted when there are no pending activities on the interface clock domain, without evaluating the pending activities on the functional clock domain (the GP timer enters into sleep mode, and if a pending interrupt event is finished during idle mode, the wake-up signal is asserted). The wake-up signal is enabled.
11	ON	ON	None of the clocks are shut down. Therefore, the GP timer can potentially acknowledge the IDLE request without checking the internal functionalities linked to its clocks.	

CAUTION

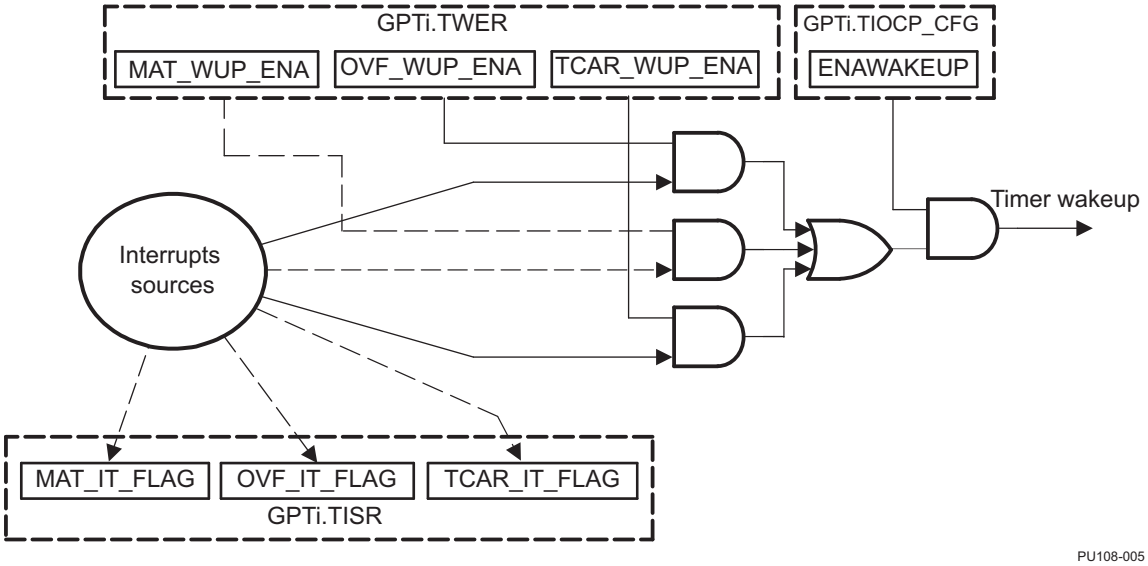
The PRCM module does not have any hardware means to read the CLOCKACTIVITY settings. The software must ensure consistent programming between the GP timer CLOCKACTIVITY and the PRCM functional clock and interface clock control bits. If the GP timer is disabled in both CM_FCLKEN and CM_ICLKEN PRCM registers while CLOCKACTIVITY is set to 11, nothing prevents the PRCM module from asserting its IDLE request, which is acknowledged regardless of the features associated with the GP timer clocks. This can lead to unpredictable behavior.

13.2.3.1.2 Wake-Up Capability

If the GPTi.TIOCP_CFG[4:3] IDLEMODE bit field sets the smart-idle mode, the timer module evaluates its internal capability to have the interface clock switched off. When there is no more internal activity (no pending interrupt sources: match, overflow, or timer capture events), the idle acknowledge signal is asserted and the timer enters into sleep mode, ready to issue a wake-up request. This wake-up request is sent only if the GPTi.TIOCP_CFG[2] ENAWAKEUP bit enables the timer wake-up capability.

Figure 13-5 shows the wake-up request generation. For more information on the GP timer clock control, see Section 13.2.3.1.1, *Clock Management*.

Figure 13-5. Wake-Up Request Generation



The timer wake-up enable register (GPTi.TWER) allows masking the expected source of the wake-up event that generates a wake-up request. The GPTi.TWER register is programmed synchronously with the interface clock before the PRCM module sends an idle mode request. The expected source of the wake-up event is an overflow (GPTi.TCRR), a timer match (the compare result of GPTi.TCRR and GPTi.TMAR matches the counter value), and a timer capture (an external pulse transition of the correct polarity is detected on the GPTi_EVENT_CAPTURE).

When the wake-up event is issued, the associated interrupt status bit is set in the timer status register (GPTi.TISR). The pending wake-up event is reset when the set status bit is overwritten by 1.

NOTE: The status bit must be reset to re-enter idle mode.

13.2.3.1.3 Reset and Power Management

GPTIMER1 belongs to the WKUP power domain. As part of that domain, this GP timer is sensitive to a WKUP_RST signal issued by the PRCM module. For further details about the WKUP power domain implementation and the WKUP_RST signal, see , *Power, Reset, and Clock Management*.

GPTIMER2 through GPTIMER9 belong to the PER power domain. As part of that domain, these GP timers are sensitive to a PER_RST signal issued by the PRCM module. For further details about the PER power domain implementation and the PER_RST signal, see , *Power, Reset, Clock Management*.

GPTIMER10 and GPTIMER11 belong to the CORE power domain. As part of that domain, these GP timers are sensitive to a CORE_RST signal issued by the PRCM module. For further details about the CORE power domain implementation and the CORE_RST signal, see , *Power, Reset, and Clock Management*.

13.2.3.2 Software Reset

Two bit fields (GPTi.TIOCP_CFG[1] SOFTRESET and GPTi.TSICR[1] SFT) can generate a software reset of the GP timer. For both of these bits, all read accesses return 0.

The GPTi.TIOCP_CFG[1] SOFTRESET bit allows resetting the functional and interface domains. The GPTi.TSICR[1] SFT bit allows resetting the functional part of the GP timer.

Before accessing or using the GP timer, the local host must ensure that both internal resets are released by reading the GPTi.TISTAT[0] RESETDONE bit. This bit field monitors the internal reset status.

13.2.3.3 GP Timer Interrupts

Table 13-7 lists the interrupt mapping from the 12 GP timers to the three internal processors.

Table 13-7. Timer Interrupt Names and Processor IRQ Mapping

Timer	Interrupt Name	Mapping	Comments
GPTIMER 1	GPT1_IRQ	M_IRQ_37	GPTIMER1 interrupt to MPU
GPTIMER2	GPT2_IRQ	M_IRQ_38	GPTIMER2 interrupt to MPU
GPTIMER3	GPT3_IRQ	M_IRQ_39	GPTIMER3 interrupt to MPU
GPTIMER4	GPT4_IRQ	M_IRQ_40	GPTIMER4 interrupt to MPU
GPTIMER5	GPT5_IRQ	M_IRQ_41	GPTIMER5 interrupt to MPU
		IVA2_IRQ[6]	GPTIMER5 interrupt to IVA2.2
GPTIMER6	GPT6_IRQ	M_IRQ_42	GPTIMER6 interrupt to MPU
		IVA2_IRQ[7]	GPTIMER6 interrupt to IVA2.2
		MD_IRQ_6	GPTIMER6 interrupt to modem subsystem (D2D)
GPTIMER7	GPT7_IRQ	M_IRQ_43	GPTIMER7 interrupt to MPU
		IVA2_IRQ[8]	GPTIMER7 interrupt to IVA2.2
		MD_IRQ_7	GPTIMER7 interrupt to modem subsystem (D2D)
GPTIMER8	GPT8_IRQ	M_IRQ_44	GPTIMER8 interrupt to MPU
		IVA2_IRQ[9]	GPTIMER8 interrupt to IVA2.2
		MD_IRQ_8	GPTIMER8 interrupt to modem subsystem (D2D)
GPTIMER9	GPT9_IRQ	M_IRQ_45	GPTIMER9 interrupt to MPU
		MD_IRQ_9	GPTIMER9 interrupt to modem subsystem (D2D)
GPTIMER10	GPT10_IRQ	M_IRQ_46	GPTIMER10 interrupt to MPU
GPTIMER11	GPT11_IRQ	M_IRQ_47	GPTIMER11 interrupt to MPU

The timer can issue an overflow interrupt, a timer match interrupt, and a timer capture interrupt. Each internal interrupt source can be independently enabled/disabled in the interrupt enable register (GPTi.TIER). When the interrupt event is issued, the associated interrupt status bit is set in the timer status register (GPTi.TISR). The pending interrupt event is reset when the set status bit is overwritten by a1.

13.2.4 GP Timers Functional Description

Each GP timer contains a free-running upward counter with autoreload capability on overflow. The timer counter can be read and written on-the-fly (while counting). Each GP timer includes compare logic to allow an interrupt event on a programmable counter matching value. A dedicated output signal can be pulsed or toggled on either an overflow or a match event. This offers time-stamp trigger signaling or PWM signal sources. A dedicated input signal can be used to trigger an automatic timer counter capture or an interrupt event on a programmable input signal transition type. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged into one module interrupt line and one wake-up line. Each internal interrupt source can be independently enabled/disabled with a dedicated bit of the GPTi.TIER register for the interrupt features and a dedicated bit of the GPTi.TWER register for the wakeup. In addition, GPTIMER1, GPTIMER2, and GPTIMER10 have implemented a mechanism to generate an accurate tick interrupt.

For each GP timer implemented in the device, there are two possible clock sources:

- 32-kHz clock
- System clock

Selection of the input clock source is done in the registers in the PRCM configuration (see Section 13.2.1, *GP Timer Overview*).

Each GP timer supports three functional modes:

- Timer mode
- Capture mode
- Compare mode

By default, after core reset, the capture and compare modes are disabled.

13.2.4.1 GP Timers Block Diagram

Figure 13-6 shows the block diagram of common GP timers, and Figure 13-7 shows the block diagram of GP timers with 1-ms tick generation module.

Figure 13-6. Block Diagram of GPTIMER3 through GPTIMER9 and GPTIMER11

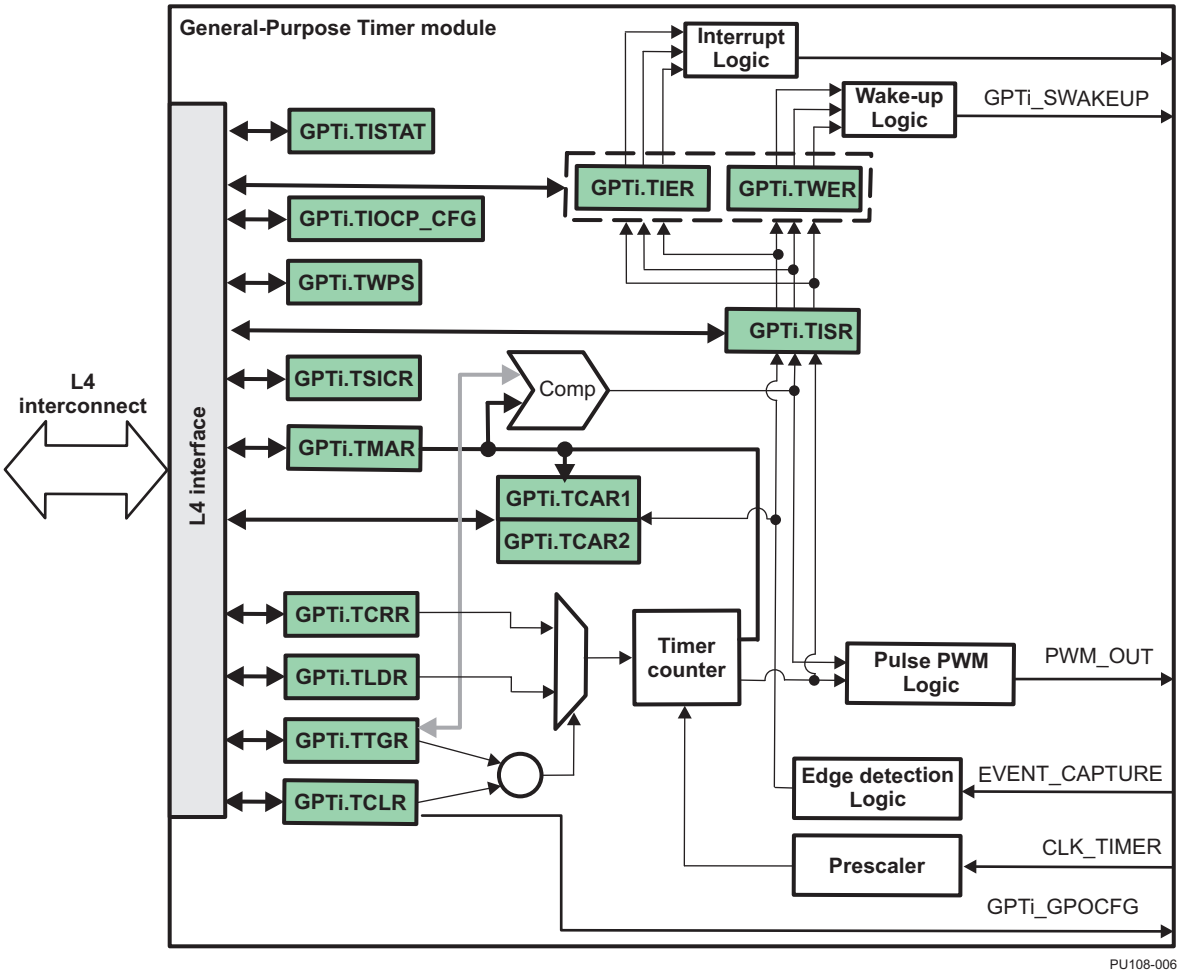
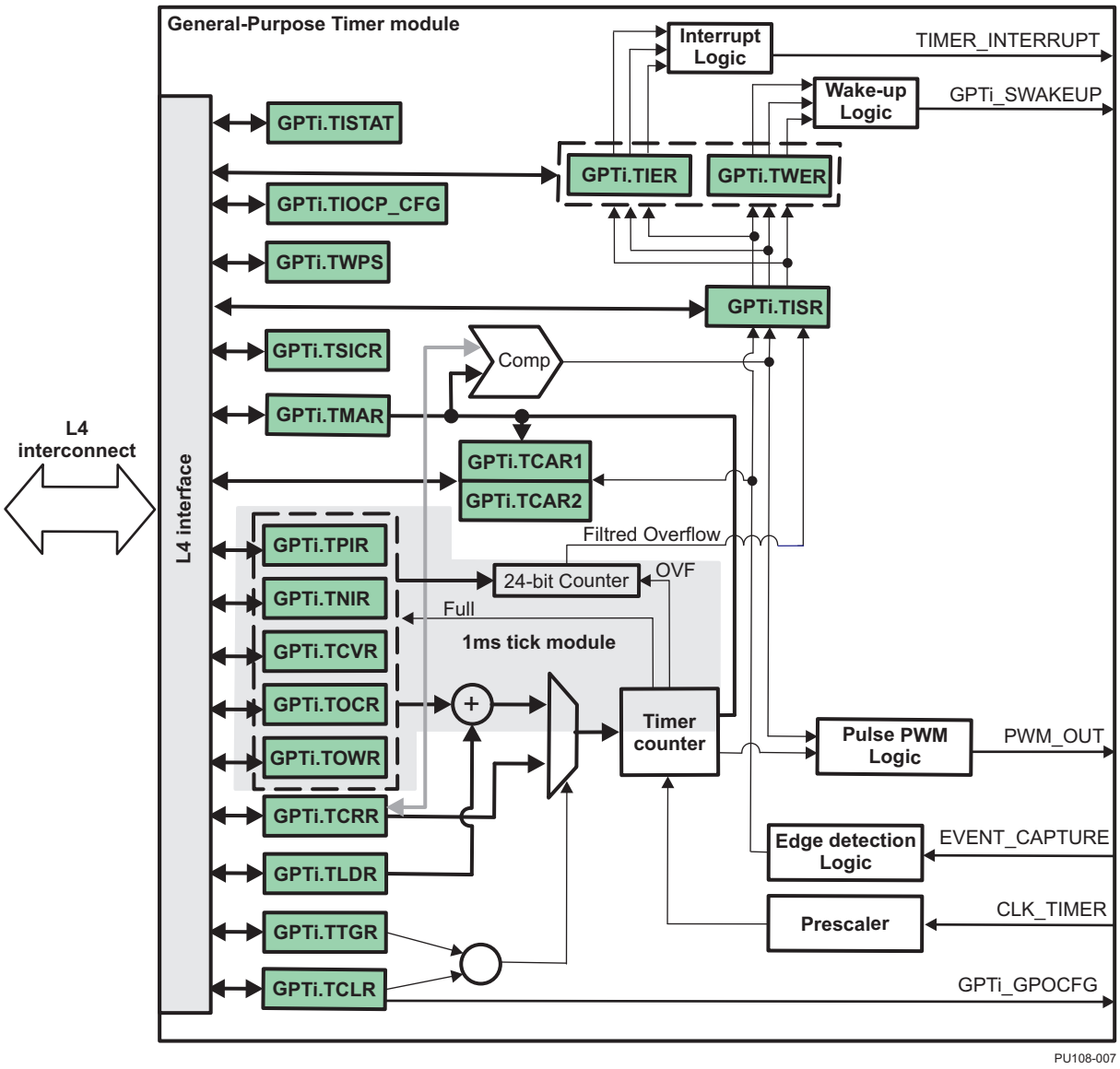


Figure 13-7. Block Diagram of GPTIMER1, GPTIMER2, and GPTIMER10



13.2.4.2 Timer Mode Functionality

The timer is an upward counter that can be started and stopped at any time through the timer control register (GPTi.TCLR[0] ST bit). The timer counter register (GPTi.TCRR) can be loaded when stopped or on-the-fly (while counting). GPTn.TCRR can be loaded directly by a GPTi.TCRR write access with a new timer value. The GPTi.TCRR register can also be loaded with the value held in the timer load register GPTi.TLDR by a trigger register (GPTi.TTGR) write access. The GPTi.TCRR loading is done regardless of the GPTi.TTGR written value. The timer counter register GPTi.TCRR value can be read when stopped or captured on-the-fly by a GPTi.TCRR read access. The timer is stopped and the counter value is set to 0 when the module reset is asserted. The timer is maintained at stop after the reset is released.

In one-shot mode (the GPTi.TCLR[1] AR bit set to 0), the counter is stopped after counting overflow occurs (the counter value remains at 0).

When the autoreload mode is enabled (the GPTi.TCLR[1] AR bit set to 1), the GPTi.TCRR register is reloaded with the timer load register (GPTi.TLDR) value after a counting overflow occurs.

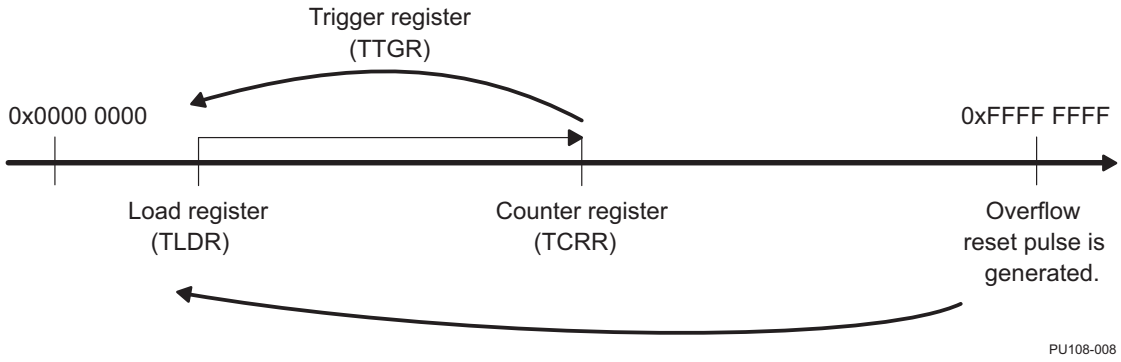
CAUTION

Do not put the overflow value (0xFFFFFFFF) in the GPTi.TLDR register because it can lead to undesired results.

An interrupt can be issued on overflow if the overflow interrupt enable bit is set in the timer interrupt enable register (GPTi.TIER[1] OVF_IT_ENA bit set to 1), the interrupt is enabled after 10 * GPTi.ICLK clock cycles. A dedicated output pin (timer PWM) can be programmed in GPTi.TCLR[12] through GPTi.TCLR[11:10] (PT and TRG bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs. The GPTi.TCLR[12] PT bit selects pulse/toggle modulation (GPTi.TCLR[11:10] TRG bit field select trigger mode).

Figure 13-8 shows the GPTi.TCRR timing value.

Figure 13-8. GPTi.TCRR Timing Value



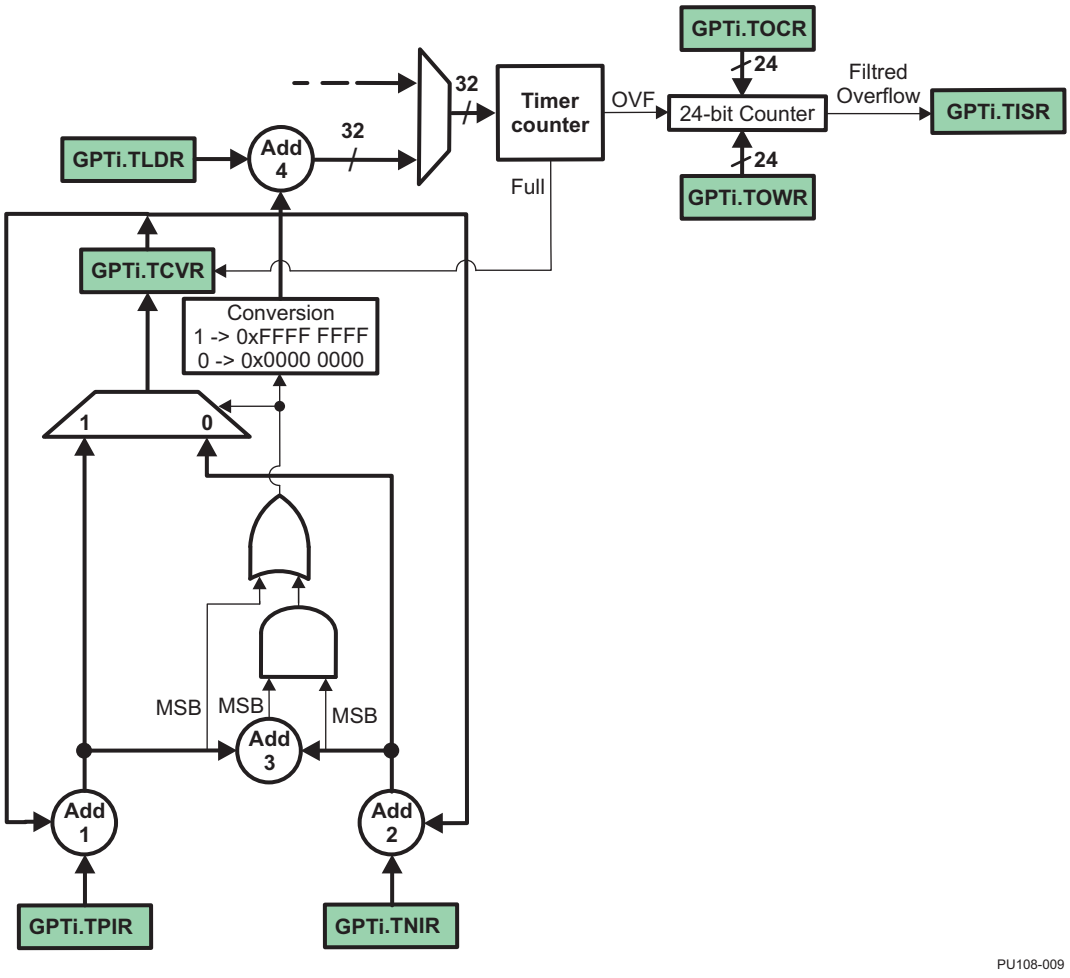
13.2.4.2.1 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10)

Because the timer input clock is 32,768 Hz, the interrupt period is not exactly 1 ms. If the clock counts up to 32, it obtains a 0.977-ms period; if it counts up to 33, it obtains a 1.007-ms period. For large granularity, the error is cumulative and can generate important deviations to the standard value.

To minimize the error between a true 1-ms tick and the tick generated by the 32,768 Hz timer, the sequencing of periods less than 1 ms and periods greater than 1 ms must be shuffled. An additional 1-ms block is used to correct this error. Refer to Figure 13-9.

In this implementation, the increment sequencing is automatically managed by the timer to minimize the error. The user must define only the value of the timer positive increment register (GPTi.TPIR[31:0] POSITIVE_INC_VALUE bit field) and the timer negative increment register (GPTi.TNIR[31:0] NEGATIVE_INC_VALUE bit field). An automatic adaptation mechanism is used to simplify the programming model.

Figure 13-9. Block Diagram of the 1-ms Tick Module



The GPTi.TPIR, GPTi.TNIR, and GPTi.TCVR registers and adders Add1, Add2, and Add3 are used to define whether the next value loaded in the timer counter register (GPTi.TCRR[31:0] TIMER_COUNTER bit field) is the value of the GPTi.TLDR[31:0] LOAD_VALUE bit field (period less than 1 ms) or the value of GPTi.TLDR[31:0] LOAD_VALUE - 1 (period greater than 1 ms).

Table 13-8 lists the value loaded in the GPTi.TCRR register according to the sign of the result of Add1, Add2, and Add3.

MSB = 0: Positive value, MSB = 1: Negative value

Table 13-8. Value Loaded in GPTi.TCRR to Generate 1-ms Tick

Add1 MSB	Add2 MSB	Add3 MSB	Value of GPTi.TCRR Register
0	0	0	GPTi.TLDR[31:0] LOAD_VALUE bit field
0	0	1	GPTi.TLDR[31:0] LOAD_VALUE bit field
0	1	0	GPTi.TLDR[31:0] LOAD_VALUE bit field
0	1	1	GPTi.TLDR[31:0] LOAD_VALUE - 1
1	0	0	N/A
1	0	1	N/A
1	1	0	GPTi.TLDR[31:0] LOAD_VALUE - 1
1	1	1	GPTi.TLDR[31:0] LOAD_VALUE - 1

The values of the GPTi.TPIR and GPTi.TNIR registers are calculated using the following formula:

- Positive increment value = ((INTEGER[Fclk * Ttick] + 1) * 1e6) - (Fclk * Ttick * 1e6)
- Negative increment value = (INTEGER[Fclk * Ttick] * 1e6) - (Fclk * Ttick * 1e6)

NOTE: Fclk clock frequency (kHz)

Ttick tick period (ms)

The timer overflow counter register (GPTi.TOCR) and the timer overflow wrapping register (GPTi.TOWR) are used to filter interrupts. When the timer overflows, it increments the 24-bit TOCR register. When the 24-bit TOCR register values match the value in the 24-bit TOWR register and the timer overflow is asserted, the TOCR register is reset and an interrupt is generated to the TISR register.

With the conversion block in reset state (the positive increment register, negative increment register, and counter value register are zeroed), the programming model and the behavior of GPTIMER1, GPTIMER2, and GPTIMER10 remain unchanged.

For 1-ms tick with a 32,768-Hz clock:

- GPTi.TPIR[31:0] POSITIVE_INC_VALUE = 232000
- GPTi.TNIR[31:0] NEGATIVE_INC_VALUE = -768000
- GPTi.TLDR[31:0] LOAD_VALUE = 0xFFFFFFE0

NOTE: Any value of the tick period can be generated with the appropriate value of the GPTi.TPIR, GPTi.TNIR, and GPTi.TLDR registers.

By default, the GPTi.TPIR, GPTi.TNIR, GPTi.TCVR, GPTi.TOCR, and GPTi.TOWR registers and the associated logic are in reset mode (all 0s) and have no action on the programming model.

13.2.4.3 Capture Mode Functionality

When a transition is detected on the module input pin (EVENT_CAPTURE), the timer value in the GPTi.TCRR register can be captured and saved in the GPTi.TCAR1 or GPTi.TCAR2 register function of the mode selected in the GPTi.TCLR[13] CAPT_MODE bit. The edge detection circuitry monitors transitions on the input pin (EVENT_CAPTURE).

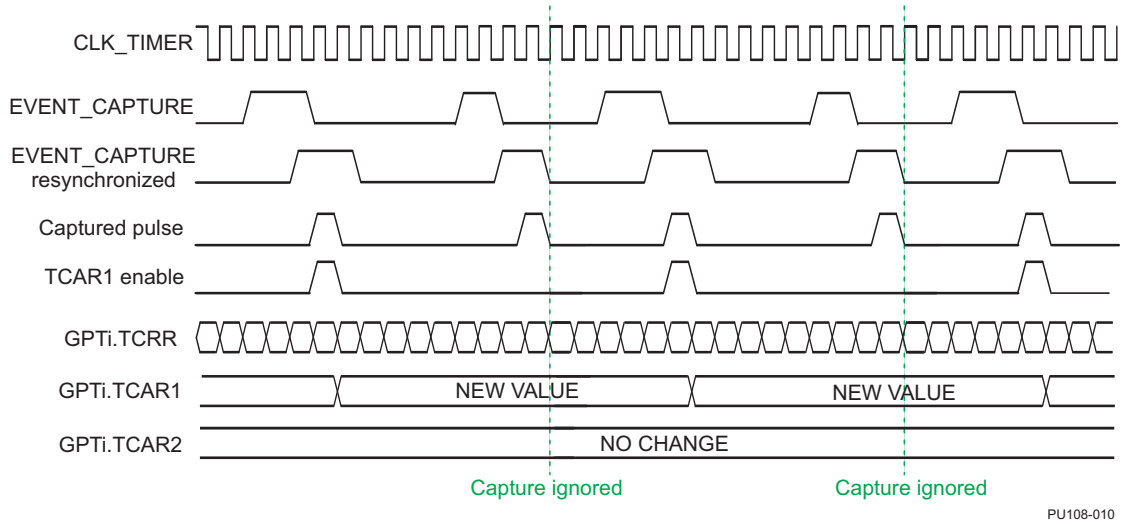
The rising edge, falling edge, or both, can be selected in the GPTi.TCLR[9:8] TCM field to trigger the timer counter capture. The module sets the GPTi.TISR[2] TCAR_IT_FLAG bit when an active edge is detected, and at the same time, the counter value GPTi.TCRR is stored in timer capture register GPTi.TCAR1 or GPTi.TCAR2, as follows:

- If the GPTi.TCLR[13] CAPT_MODE bit is 0, then on the first enabled capture event the value of the counter register is saved in the GPTi.TCAR1 register, and all the next events are ignored (no update on the GPTi.TCAR1 register and no interrupt triggering) until the detection logic is reset or the GPTi.TISR[2] TCAR_IT_FLAG bit is cleared by writing 1 in it.
- If the GPTi.TCLR[13] CAPT_MODE bit is 1, then on the first enabled capture event the value of the counter register is saved in the GPTi.TCAR1 register, and on the second enabled capture event, the value of the counter register is saved in the GPTi.TCAR2 register. If a capture interrupt is enabled, the interrupt triggers on the second event capture. All other events are ignored (no update on GPTi.TCAR1/GPTi.TCAR2 and no interrupt triggering) until the detection logic is reset or GPTi.TISR[2] TCAR_IT_FLAG bit is cleared by writing 1 in it. This mechanism is useful for period calculation of a clock, if that clock is connected to the EVENT_CAPTURE input pin.

The edge detection logic is reset (a new capture is enabled) when the active capture interrupt is servedthe GPTi.TISR[2] TCAR_IT_FLAG bit (previously 1) is cleared by writing 1 to it or when the edge detection mode bits (the GPTi.TCLR[9:8] TCM field) are changed from no-capture mode detection to any other mode. The timer functional clock (input to prescaler) is used to sample the input pin (EVENT_CAPTURE). An input negative or positive pulse can be detected when the pulse time is greater than the functional clock period. An interrupt is issued on edge detection if the capture interrupt enable bit is set in the GPTi.TIER[2] TCAR_IT_ENA bit. See the examples in Figure 13-10 and Figure 13-11.

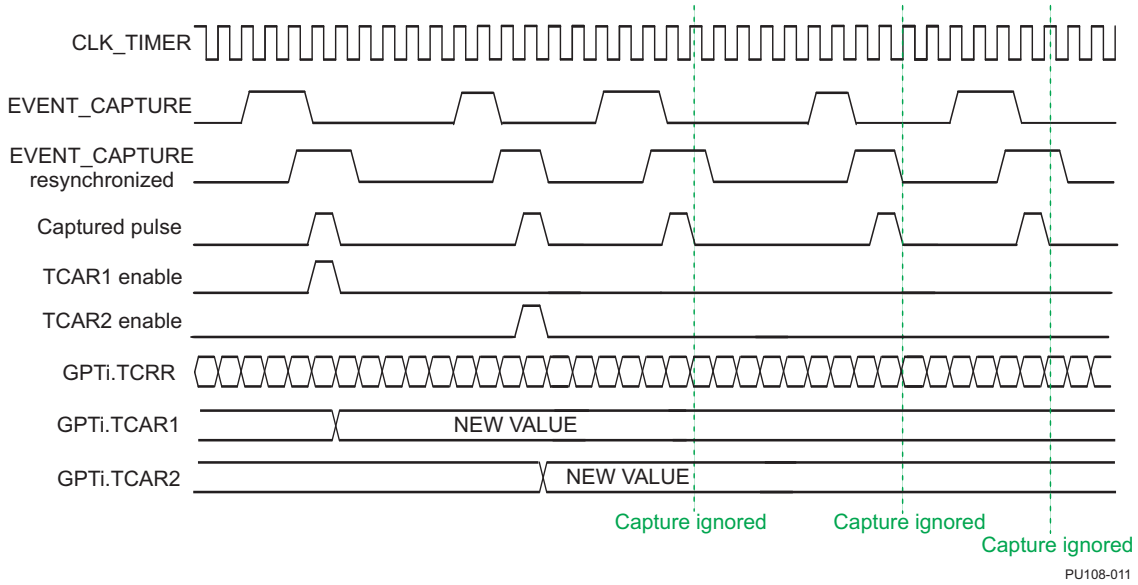
In Figure 13-10, the GPTi.TCLR[9:8] TCM value is 0b01, and GPTi.TCLR[13] CAPT_MODE is 0. Only the rising edge of EVENT_CAPTURE triggers a capture in the GPTi.TCAR1 and GPTi.TCAR2 registers, and only the GPTi.TCAR1 register updates.

Figure 13-10. Capture Wave Example for GPTi.TCLR[13] CAPT_MODE = 0



In Figure 13-11, the GPTi.TCLR[9:8] TCM value is 0b01, and GPTi.TCLR[13] CAPT_MODE is 1. Only the rising edge of EVENT_CAPTURE triggers a capture in the GPTi.TCAR1 register on the first enabled event, and the GPTi.TCAR2 register updates on the second enabled event.

Figure 13-11. Capture Wave Example for GPTi.TCLR[13] CAPT_MODE = 1



13.2.4.4 Compare Mode Functionality

When the compare enable register GPTi.TCLR[6] CE bit is set to 1, the timer value (GPTi.TCRR[31:0] TIMER_COUNTER field) is continuously compared to the value held in the timer match register (GPTi.TMAR). The GPTi.TMAR[31:0] COMPARE_VALUE value can be loaded at any time (timer counting or stopped). When the GPTi.TCRR and the GPTi.TMAR values match, an interrupt is issued, if the GPTi.TIER[0] MAT_IT_ENA bit is set.

The dedicated output pin (timer PWM) can be programmed in the GPTi.TCLR[12] PT bit through the GPTi.TCLR[11:10] TRG field to generate one positive pulse (timer clock duration) or to invert the current value (toggle mode) when an overflow or a match occurs.

13.2.4.5 Prescaler Functionality

A prescaler can be used to divide the timer counter input clock frequency. The prescaler is enabled when the GPTi.TCLR[5] PRE bit is set. The GPTi.TCLR[4:2] PTV field sets the second prescaler ratio. The prescaler counter is reset when the timer counter is stopped or reloaded on-the-fly.

Table 13-9 lists the prescaler/timer reload values versus contexts.

Table 13-9. Prescaler/Timer Reload Values Versus Contexts

Context	Prescaler	Timer Counter
Overflow (when autoreload is on)	Reset	GPTi.TLDR[31:0]
TCRR write	Reset	GPTi.TCRR[31:0]
TTGR write	Reset	GPTi.TLDR[31:0]
Stop	Reset	Frozen

13.2.4.6 Pulse-Width Modulation

The timer can be configured to provide a programmable PWM output. The timer PWM output pin can be configured to toggle on an event. The GPTi.TCLR[11:10] TRG field determines on which register value the PWM pin toggles. Either overflow alone or both overflow and match can be selected to toggle the timer PWM pin when a compare condition occurs.

CAUTION

In toggle mode when GPTi.TCLR[11:10] TRG = 0x2 (overflow and match), the first event that will toggle the PWM line is an overflow event. If a match event occurs first, it will not toggle the PWM line. Figure 13-13 illustrates those.

The GPTi.TCLR[7] SCPWM bit can be programmed to set or clear the timer PWM output signal only while the counter is stopped or the trigger is off. This allows setting the output pin to a known state before modulation starts. Modulation synchronously stops when the GPTi.TCLR[11:10] TRG field is cleared and overflow occurs. This allows fixing a deterministic state of the output pin when modulation stops.

In Figure 13-12, the internal overflow pulse is set each time (0xFFFF FFFF - GPTi.TLDR[31:0] LOAD_VALUE + 1) the value is reached, and the internal match pulse is set when the counter reaches the GPTi.TMAR register value. According to the value of the GPTi.TCLR[12] PT and GPTi.TCLR[11:10] TRG bits, the timer provides pulse or PWM event on the output pin (timer PWM).

The GPTi.TLDR and GPTi.TMAR registers must keep values smaller than the overflow value (0xFFFF FFFF) by at least two units. In case the PWM trigger events are both overflow and match, the difference between the values kept in the GPTi.TMAR register and the value in the GPTi.TLDR register must be at least two units. When match event is used, the compare mode GPTi.TCLR[6] CE bit must be set.

In Figure 13-12, the GPTi.TCLR[7] SCPWM bit is set to 0. In Figure 13-13, the GPTi.TCLR[7] SCPWM bit is set to 1. To obtain the desired wave form, start the counter at 0xFFFF FFFE value (to ensure an overflow first) or adjust the line polarity (GPTi.TCLR[7] SCPWM bit).

Figure 13-12. Timing Diagram of PWM With GPTi.TCLR[7] SCPWM Bit = 0

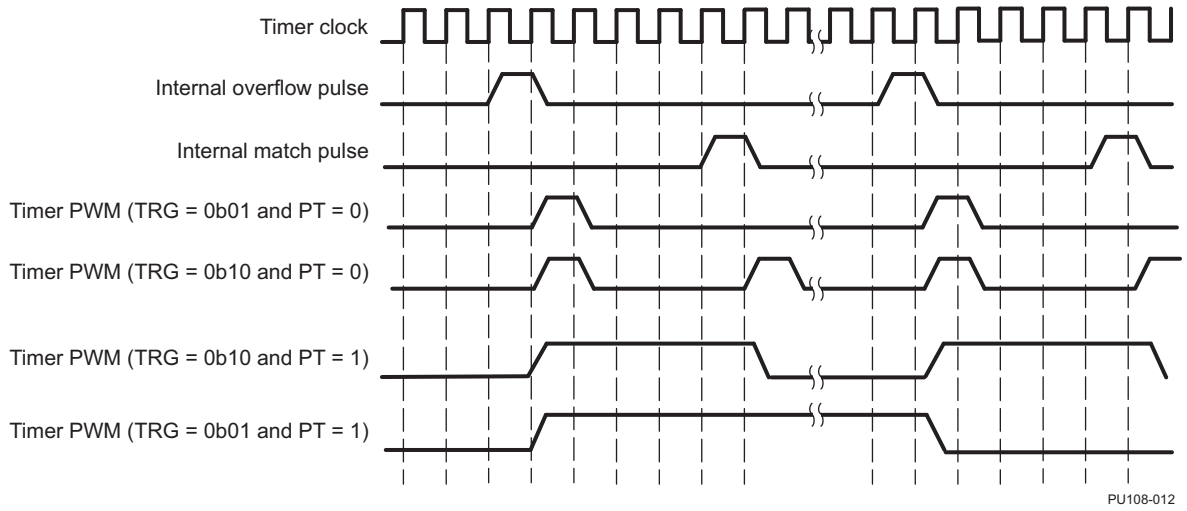
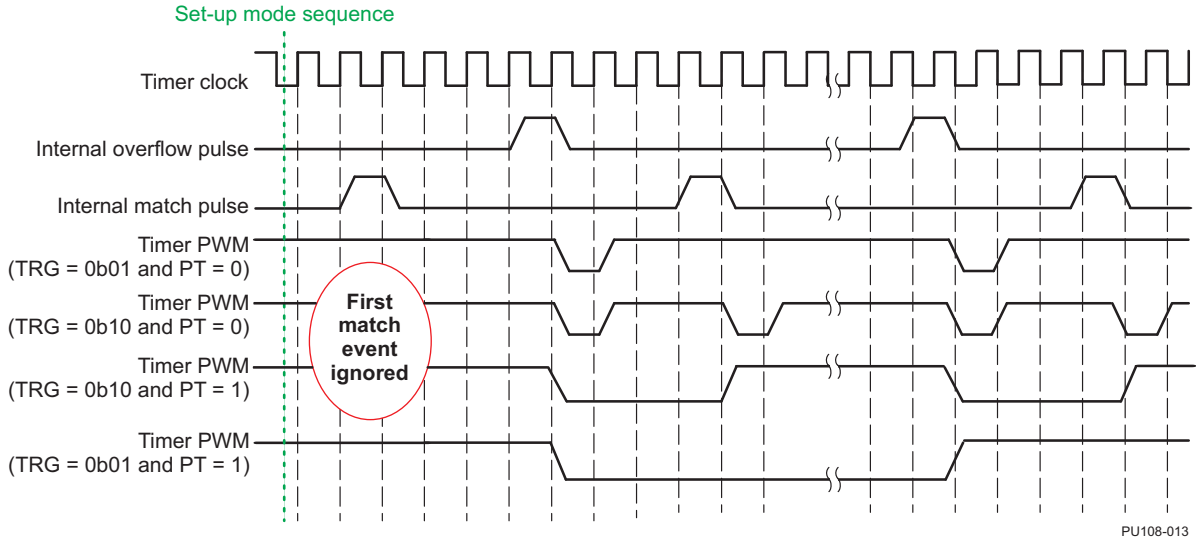


Figure 13-13. Timing Diagram of PWM With GPTi.TCLR[7] SCPWM Bit = 1



13.2.4.7 Timer Counting Rate

The timer rate is defined by the following values:

- Value of the prescaler fields (GPTi.TCLR[5] PRE bit and GPTi.TCLR[4:2] PTV field)
- Value loaded into the timer load register (GPTi.TLDR)

Table 13-10 lists prescaler clock ratio values.

Table 13-10. Prescaler Clock Ratio Values

GPTi.TCLR[5] PRE	GPTi.TCLR[4:2] PTV	Divisor (PS)
0	X	1
1	0	2
1	1	4
1	2	8
1	3	16
1	4	32

Table 13-10. Prescaler Clock Ratio Values (continued)

GPTi.TCLR[5] PRE	GPTi.TCLR[4:2] PTV	Divisor (PS)
1	5	64
1	6	128
1	7	256

Thus, the timer overflow-rate is expressed as:

OVF_Rate = (0xFFFF FFFF - GPTn.TLDR + 1) * (timer-functional clock period) * PS

With (timer-functional clock period) = 1 / (timer-functional clock frequency) and PS = $2^{(PTV + 1)}$ if prescaler is enabled, or PS = 1 if prescaler is disabled.

CAUTION

Internal resynchronization causes any write to the GPTn.TCLR[1] ST bit to have some latency before the register is updated:

2.5 * functional clock cycles write_GPTn.TCLR_latency 3.5 * functional clock cycles

Remember to take this latency into account whenever the timer must be started or stopped by a software change to the GPTn.TCLR[1] ST bit.

CAUTION

- In the non-PWM mode, GTPi.TLDR must be maintained at less than or equal to 0xFFFF FFFE.
- In the PWM mode, GTPi.TLDR must be maintained at less than or equal to 0xFFFF FFFD.

For example, with a timer clock input of 32 kHz and a GPTn.TCLR[5] PRE field equal to 0, the timer output period is as listed in Table 13-11.

Table 13-11. Value and Corresponding Interrupt Period

GPTi.TLDR[31:0] LOAD_VALUE	Interrupt Period
0x0000 0000	39 h
0xFFFF 0000	2.1 s
0xFFFF FFF0	524 µs
0xFFFF FFFE	65.5 µs

13.2.5 Timer Under Emulation

During emulation mode, the timer continues to run according to the value of the GPTi.TIOCP_CFG[5] EMUFREE bit.

If the GPTi.TIOCP_CFG[5] EMUFREE bit is set to 1, timer execution is not stopped in emulation mode and the interrupt is still generated when overflow or match is reached.

If the GPTi.TIOCP_CFG[5] EMUFREE bit is set to 0, the prescaler and timer are frozen and both resume on exit from emulation mode. The asynchronous external input pin (gpti_pwm_evt, with i=[9:12]) is internally synchronized on two timer-clock rising edges.

13.2.6 Accessing GP Timer Registers

All accesses are nonposted until software reconfiguration.

All registers are 32 bits wide, accessible through the L4 interface with 16-bit or 32-bit access (read/write).

Any 16-bit write access must be least-significant bit (LSB) first, and the second write access must be most-significant bit (MSB). Write operations to the GP timer registers (GPTi.TIDR, GPTi.TIOCP_CFG, GPTi.TISR, GPTi.TIER, GPTi.TWER, and GPTi.TSICR) can skip the MSB access if it is not necessary to update the 16 MSBs of the register.

Write operations to any functional register (GPTi.TCLR, GPTi.TCRR, GPTi.TLDR, GPTi.TTGR, and GPTi.TMAR, and GPTi.TPIR, GPTi.TNIR, GPTi.TCVR , GPTi.TOCR , and GPTi.TOWR for GPTIMER1, GPTIMER2, and GPTIMER10) must be complete (the MSB must be written even if the MSB data is not used).

13.2.6.1 Writing to Timer Registers

The host uses the L4 interface to write the following registers synchronously with the timer interface clock:

- GPTi.TLDR
- GPTi.TCRR
- GPTi.TIER
- GPTi.TISR
- GPTi.TCLR
- GPTi.TIOCP_CFG
- GPTi.TWER
- GPTi.TTGR
- GPTi.TSICR
- GPTi.TMAR

GPTIMER1, GPTIMER2, and GPTIMER10 also have the following registers:

- GPTi.TPIR
- GPTi.TNIR
- GPTi.TCVR
- GPTi.TOCR
- GPTi.TOWR

In 16-bit access mode, the 16 LSBs must be written before writing to the 16 MSBs.

13.2.6.1.1 Write Posting Synchronization Mode

This mode is used if the GPTi.TSICR[2] POSTED bit is set to 1.

This mode uses a posted write scheme to update any internal register (GPTi.TCLR, GPTi.TCRR, GPTi.TLDR, GPTi.TTGR, GPTi.TMAR, and GPTi.TPIR, GPTi.TNIR, GPTi.TCVR, GPTi.TOCR, and GPTi.TOWR for GPTIMER1, GPTIMER2, and GPTIMER10). Therefore, the write transaction is immediately acknowledged on the L4 interface, although the effective write operation occurs later, because of a resynchronization in the timer clock domain. The advantage is that neither the interconnect nor the device that requested the write transaction is stalled.

For each register, a status bit is provided in the timer write-posted status register GPTi.TWPS. In this mode, it is mandatory that the software checks this status bit prior to any write access. In case a write is attempted to a register with a previous access pending, the previous access is discarded without notice.

The timer module updates the timer counter register value synchronously with the L4 clock. Consequently, any read access to the timer counter register GPTi.TCRR does not add any resynchronization latency; the current value is always available.

If a write access is pending for a register, reading from this register does not yield a correct result. Software synchronization must be used to avoid incorrect results.

The drawback of this automatic update mechanism is that it assumes a given relationship between the timer interface frequency and the timer clock frequency.

Functional frequency range: freq(timer clock) freq(L4 interface clock)/4

13.2.6.1.2 Write Nonposting Synchronization Mode

This mode is used if the GPTi.TSICR[2] POSTED bit is set to 0.

This mode uses a nonposted write scheme to update any internal register. Therefore, the write transaction is not acknowledged on the L4 interface until the effective write operation occurs after the resynchronization in the timer functional clock domain. The drawback is that both the interconnect and the device that requested the write transaction are stalled during this period.

The same full resynchronization scheme is used for a read transaction, and the same stall period applies. A register read following a write to the same register is always coherent.

This mode is functional regardless of the ratio between the L4 interface frequency and the timer clock frequency.

13.2.6.2 Reading From Timer Counter Registers

In 16-bit access mode, reading the 16 LSBs from the timer counter registers (GPTi.TCRR, GPTi.TCAR1, and GPTi.TCAR2) captures the current timer counter value. This must be followed by reading the 16MSBs.

IVA2.2 subsystem 16-bit accesses can be interleaved with MPU subsystem 32-bit accesses.

NOTE: LSB/MSB accesses cannot be interleaved (that is, the sequence LSB register 1, LSB register 2, MSB register 1, MSB register 2 is not supported).

13.3 General-Purpose Timers Register Manual

13.3.1 GP Timer Register Map

13.3.1.1 Instance Summary

Table 13-12 lists the base address and block size for the GP timer module instances. All timers are memory mapped to the L4 peripheral bus memory space.

Table 13-12. GP Timer Instance Summary

Module Name	Base Address	Size
GPTIMER1	0x4831 8000	4K bytes
GPTIMER2	0x4903 2000	4K bytes
GPTIMER3	0x4903 4000	4K bytes
GPTIMER4	0x4903 6000	4K bytes
GPTIMER5	0x4903 8000	4K bytes
GPTIMER6	0x4903 A000	4K bytes
GPTIMER7	0x4903 C000	4K bytes
GPTIMER8	0x4903 E000	4K bytes
GPTIMER9	0x4904 0000	4K bytes
GPTIMER10	0x4808 6000	4K bytes
GPTIMER11	0x4808 8000	4K bytes

13.3.2 GP Timer Register Mapping Summary

CAUTION

The GP timer registers are limited to 32-bit and 16-bit data accesses; 8-bit access is not allowed and can corrupt the register content.

Table 13-13 through Table 13-15 provide the register summary and associated offset addresses for the 11 GP timer internal registers. (Example: The physical address for the TCLR register of GPTIMER8 is 0x4903 E024.)

Table 13-13. GPTIMER1 to GPTIMER4 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address (GPTIMER1)	Physical Address (GPTIMER2)	Physical Address (GPTIMER3)	Physical Address (GPTIMER4)
TIDR	R	32	0x000	0x4831 8000	0x4903 2000	0x4903 4000	0x4903 6000
TIOCP_CFG	RW	32	0x010	0x4831 8010	0x4903 2010	0x4903 4010	0x4903 6010
TISTAT	R	32	0x014	0x4831 8014	0x4903 2014	0x4903 4014	0x4903 6014
TISR	RW	32	0x018	0x4831 8018	0x4903 2018	0x4903 4018	0x4903 6018
TIER	RW	32	0x01C	0x4831 801C	0x4903 201C	0x4903 401C	0x4903 601C
TWER	RW	32	0x020	0x4831 8020	0x4903 2020	0x4903 4020	0x4903 6020
TCLR	RW	32	0x024	0x4831 8024	0x4903 2024	0x4903 4024	0x4903 6024
TCRR	RW	32	0x028	0x4831 8028	0x4903 2028	0x4903 4028	0x4903 6028
TLDR	RW	32	0x02C	0x4831 802C	0x4903 202C	0x4903 402C	0x4903 602C
TTGR	RW	32	0x030	0x4831 8030	0x4903 2030	0x4903 4030	0x4903 6030
TWPS	R	32	0x034	0x4831 8034	0x4903 2034	0x4903 4034	0x4903 6034
TMAR	RW	32	0x038	0x4831 8038	0x4903 2038	0x4903 4038	0x4903 6038
TCAR1	R	32	0x03C	0x4831 803C	0x4903 203C	0x4903 403C	0x4903 603C
TSICR	RW	32	0x040	0x4831 8040	0x4903 2040	0x4903 4040	0x4903 6040
TCAR2	R	32	0x044	0x4831 8044	0x4903 2044	0x4903 4044	0x4903 6044
TPIR	RW	32	0x048	0x4831 8048	0x4903 2048	-	-
TNIR	RW	32	0x04C	0x4831 804C	0x4903 204C	-	-
TCVR	RW	32	0x050	0x4831 8050	0x4903 2050	-	-
TOCR	RW	32	0x054	0x4831 8054	0x4903 2054	-	-
TOWR	RW	32	0x058	0x4831 8058	0x4903 2058	-	-

Table 13-14. GPTIMER5 to GPTIMER8 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address (GPTIMER5)	Physical Address (GPTIMER6)	Physical Address (GPTIMER7)	Physical Address (GPTIMER8)
TIDR	R	32	0x000	0x4903 8000	0x4903 A000	0x4903 C000	0x4903 E000
TIOCP_CFG	RW	32	0x010	0x4903 8010	0x4903 A010	0x4903 C010	0x4903 E010
TISTAT	R	32	0x014	0x4903 8014	0x4903 A014	0x4903 C014	0x4903 E014
TISR	RW	32	0x018	0x4903 8018	0x4903 A018	0x4903 C018	0x4903 E018
TIER	RW	32	0x01C	0x4903 801C	0x4903 A01C	0x4903 C01C	0x4903 E01C
TWER	RW	32	0x020	0x4903 8020	0x4903 A020	0x4903 C020	0x4903 E020
TCLR	RW	32	0x024	0x4903 8024	0x4903 A024	0x4903 C024	0x4903 E024
TCRR	RW	32	0x028	0x4903 8028	0x4903 A028	0x4903 C028	0x4903 E028
TLDR	RW	32	0x02C	0x4903 802C	0x4903 A02C	0x4903 C02C	0x4903 E02C
TTGR	RW	32	0x030	0x4903 8030	0x4903 A030	0x4903 C030	0x4903 E030
TWPS	R	32	0x034	0x4903 8034	0x4903 A034	0x4903 C034	0x4903 E034
TMAR	RW	32	0x038	0x4903 8038	0x4903 A038	0x4903 C038	0x4903 E038
TCAR1	R	32	0x03C	0x4903 803C	0x4903 A03C	0x4903 C03C	0x4903 E03C
TSICR	RW	32	0x040	0x4903 8040	0x4903 A040	0x4903 C040	0x4903 E040
TCAR2	R	32	0x044	0x4903 8044	0x4903 A044	0x4903 C044	0x4903 E044

Table 13-15. GPTIMER9 to GPTIMER11 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address (GPTIMER9)	Physical Address (GPTIMER10)	Physical Address (GPTIMER11)
TIDR	R	32	0x000	0x4904 0000	0x4808 6000	0x4808 8000
TIOCP_CFG	RW	32	0x010	0x4904 0010	0x4808 6010	0x4808 8010
TISTAT	R	32	0x014	0x4904 0014	0x4808 6014	0x4808 8014
TISR	RW	32	0x018	0x4904 0018	0x4808 6018	0x4808 8018
TIER	RW	32	0x01C	0x4904 001C	0x4808 601C	0x4808 801C
TWER	RW	32	0x020	0x4904 0020	0x4808 6020	0x4808 8020
TCLR	RW	32	0x024	0x4904 0024	0x4808 6024	0x4808 8024
TCRR	RW	32	0x028	0x4904 0028	0x4808 6028	0x4808 8028
TLDR	RW	32	0x02C	0x4904 002C	0x4808 602C	0x4808 802C
TTGR	RW	32	0x030	0x4904 0030	0x4808 6030	0x4808 8030
TWPS	R	32	0x034	0x4904 0034	0x4808 6034	0x4808 8034
TMAR	RW	32	0x038	0x4904 0038	0x4808 6038	0x4808 8038
TCAR1	R	32	0x03C	0x4904 003C	0x4808 603C	0x4808 803C
TSICR	RW	32	0x040	0x4904 0040	0x4808 6040	0x4808 8040
TCAR2	R	32	0x044	0x4904 0044	0x4808 6044	0x4808 8044
TPIR	RW	32	0x048	-	0x4808 6048	-
TNIR	RW	32	0x04C	-	0x4808 604C	-
TCVR	RW	32	0x050	-	0x4808 6050	-
TOCR	RW	32	0x054	-	0x4808 6054	-
TOWR	RW	32	0x058	-	0x4808 6058	-

13.3.3 GP Timer Register Descriptions

Table 13-16 through Table 13-54 describe the GP timer register bits.

Table 13-16. TIDR

Address Offset	0x000		
Physical Address	0x4831 8000	**Instance**	GPTIMER1
	0x4903 2000		GPTIMER1
	0x4903 4000		GPTIMER1
	0x4903 6000		GPTIMER1
	0x4903 8000		GPTIMER5
	0x4903 A000		GPTIMER5
	0x4903 C000		GPTIMER5
	0x4903 E000		GPTIMER5
	0x4904 0000		GPTIMER9
	0x4808 6000		GPTIMER9
	0x4808 8000		GPTIMER9
	0x4903 2000		GPT2
	0x4903 4000		GPT3
	0x4903 6000		GPT4
	0x4903 8000		GPT5
	0x4903 A000		GPT6
	0x4903 C000		GPT7
	0x4903 E000		GPT8
	0x4904 0000		GPT9
	0x4808 6000		GPT10
	0x4808 8000		GPT11
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TID_REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0.	R	0x000000
7:0	TID_REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 13-17. Register Call Summary for Register TIDR

General-Purpose Timers • Accessing GP Timer Registers: [0]
General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [1] [2] [3]

Table 13-18. TIOCP_CFG

Address Offset	0x010		
Physical Address	0x4831 8010	**Instance**	GPTIMER1
	0x4903 2010		GPTIMER1
	0x4903 4010		GPTIMER1
	0x4903 6010		GPTIMER1
	0x4903 8010		GPTIMER5
	0x4903 A010		GPTIMER5
	0x4903 C010		GPTIMER5
	0x4903 E010		GPTIMER5
	0x4904 0010		GPTIMER9
	0x4808 6010		GPTIMER9
	0x4808 8010		GPTIMER9
	0x4903 2010		GPT2
	0x4903 4010		GPT3
	0x4903 6010		GPT4
	0x4903 8010		GPT5
	0x4903 A010		GPT6
	0x4903 C010		GPT7
	0x4903 E010		GPT8
	0x4904 0010		GPT9
	0x4808 6010		GPT10
	0x4808 8010		GPT11
Description	This register controls the various parameters of the GP timer L4 interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CLOCKACTIVITY		Reserved		EMUFREE		IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	DESCRIPTION	Type	Reset
31:10	Reserved	Write 0s for future compatibility. Reads return 0.	R	0x0000000
9:8	CLOCKACTIVITY	Clock activity during wakeup mode period: 0x0: L4 interface and Functional clocks can be switched off. 0x1: L4 interface clock is maintained during wake-up period; Functional clock can be switched off. 0x2: L4 interface clock can be switched off; Functional clock is maintained during wake-up period. 0x3: L4 interface and Functional clocks are maintained during wake-up period.	RW	0x0
7:6	Reserved	Write 0s for future compatibility. Reads return 0.	R	0x0
5	EMUFREE	Emulation mode 0x0: Timer counter frozen in emulation 0x1: Timer counter free-running in emulation	RW	0
4:3	IDLEMODE	Power management, req/ack control 0x0: Force-idle. An idle request is acknowledged unconditionally. 0x1: No-idle. An idle request is never acknowledged. 0x2: Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module. 0x3: Reserved. Do not use.	RW	0x0
2	ENAWAKEUP	Wake-up feature global control 0x0: No wake-up line assertion in idle mode 0x1: Wake-up line assertion enabled in smart-idle mode	RW	0
1	SOFTRESET	Software reset. This bit is automatically reset by the hardware. During reads, it always returns 0. 0x0: Normal mode 0x1: The module is reset.	RW	0
0	AUTOIDLE	Internal L4 interface clock gating strategy 0x0: L4 interface clock is free-running. 0x1: Automatic L4 interface clock gating strategy is applied, based on the L4 interface activity.		0

Table 13-19. Register Call Summary for Register TIOCP_CFG

General-Purpose Timers

- Clock Management: [0] [1] [2]
- Wake-Up Capability: [3] [4]
- Software Reset: [5] [6]
- Timer Under Emulation: [7] [8] [9]
- Accessing GP Timer Registers: [10]
- Writing to Timer Registers: [11]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [12] [13] [14]

Table 13-20. TISTAT

Address Offset	0x014		
Physical Address	0x4831 8014	**Instance**	GPTIMER1
	0x4903 2014		GPTIMER1
	0x4903 4014		GPTIMER1
	0x4903 6014		GPTIMER1
	0x4903 8014		GPTIMER5
	0x4903 A014		GPTIMER5
	0x4903 C014		GPTIMER5
	0x4903 E014		GPTIMER5
	0x4904 0014		GPTIMER9
	0x4808 6014		GPTIMER9
	0x4808 8014		GPTIMER9
	0x4903 2014		GPT2
	0x4903 4014		GPT3
	0x4903 6014		GPT4
	0x4903 8014		GPT5
	0x4903 A014		GPT6
	0x4903 C014		GPT7
	0x4903 E014		GPT8
	0x4904 0014		GPT9
	0x4808 6014		GPT10
	0x4808 8014		GPT11
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								Reserved							RESETDONE

Bits	Field Name	DESCRIPTION	Type	Reset
31:8	Reserved	Reads return 0.	R	0x000000
7:1	Reserved	Reads return 0.	R	0x00
0	RESETDONE	Internal reset monitoring	R	0
		0x0: Internal module reset is ongoing.		
		0x1: Reset completed		

Table 13-21. Register Call Summary for Register TISTAT

General-Purpose Timers
- Software Reset: [0]

General-Purpose Timers Register Manual
- GP Timer Register Mapping Summary: [1] [2] [3]

Table 13-22. TISR

Address Offset	0x018		
Physical Address	0x4831 8018	**Instance**	GPTIMER1
	0x4903 2018		GPTIMER1
	0x4903 4018		GPTIMER1
	0x4903 6018		GPTIMER1
	0x4903 8018		GPTIMER5
	0x4903 A018		GPTIMER5
	0x4903 C018		GPTIMER5
	0x4903 E018		GPTIMER5
	0x4904 0018		GPTIMER9
	0x4808 6018		GPTIMER9
	0x4808 8018		GPTIMER9
	0x4903 2018		GPT2
	0x4903 4018		GPT3
	0x4903 6018		GPT4
	0x4903 8018		GPT5
	0x4903 A018		GPT6
	0x4903 C018		GPT7
	0x4903 E018		GPT8
	0x4904 0018		GPT9
	0x4808 6018		GPT10
	0x4808 8018		GPT11
Description	This register shows which interrupt events are pending inside the module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													TCAR_IT_FLAG	OVF_IT_FLAG	MAT_IT_FLAG

Bits	Field Name	Description		Type	Reset
31:3	Reserved	Reads return 0.		R	0x00000000
2	TCAR_IT_FLAG	Pending capture interrupt status		RW	0
		Read 0x0:	No capture interrupt event pending		
		Write 0x0:	Status unchanged		
		Read 0x1:	Capture interrupt event pending		
		Write 0x1:	Status bit cleared		
1	OVF_IT_FLAG	Pending overflow interrupt status		RW	0
		Read 0x0:	No overflow interrupt pending		
		Write 0x0:	Status unchanged		
		Read 0x1:	Overflow interrupt pending		
		Write 0x1:	Status bit cleared		
0	MAT_IT_FLAG	Pending match interrupt status		RW	0
		Read 0x0:	No match interrupt pending		
		Write 0x0:	Status unchanged		
		Read 0x1:	Match interrupt pending		
		Write 0x1:	Status bit cleared		

Table 13-23. Register Call Summary for Register TISR

General-Purpose Timers

- Wake-Up Capability: [0]
- GP Timer Interrupts: [1]
- 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [2]
- Capture Mode Functionality: [3] [4] [5] [6]
- Accessing GP Timer Registers: [7]
- Writing to Timer Registers: [8]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [9] [10] [11]

Table 13-24. TIER

Address Offset	0x01C		
Physical Address	0x4831 801C	**Instance**	GPTIMER1
	0x4903 201C		GPTIMER1
	0x4903 401C		GPTIMER1
	0x4903 601C		GPTIMER1
	0x4903 801C		GPTIMER5
	0x4903 A01C		GPTIMER5
	0x4903 C01C		GPTIMER5
	0x4903 E01C		GPTIMER5
	0x4904 001C		GPTIMER9
	0x4808 601C		GPTIMER9
	0x4808 801C		GPTIMER9
	0x4903 201C		GPT2
	0x4903 401C		GPT3
	0x4903 601C		GPT4
	0x4903 801C		GPT5
	0x4903 A01C		GPT6
	0x4903 C01C		GPT7
	0x4903 E01C		GPT8
	0x4904 001C		GPT9
	0x4808 601C		GPT10
	0x4808 801C		GPT11
Description	This register controls (enable/disable) the interrupt events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													TCAR_IT_ENA	OVF_IT_ENA	MAT_IT_ENA

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reads return 0.	R	0x00000000
2	TCAR_IT_ENA	Enable capture interrupt	RW	0
		0x0: Disable capture interrupt.		
		0x1: Enable capture interrupt.		
1	OVF_IT_ENA	Enable overflow interrupt	RW	0
		0x0: Disable overflow interrupt.		
		0x1: Enable overflow interrupt.		
0	MAT_IT_ENA	Enable match interrupt	RW	0
		0x0: Disable match interrupt.		

Bits	Field Name	Description	Type	Reset
		0x1: Enable match interrupt.		

Table 13-25. Register Call Summary for Register TIER

General-Purpose Timers
- GP Timer Interrupts: [0]
- GP Timers Functional Description: [1]
- Timer Mode Functionality: [2]
- Capture Mode Functionality: [3]
- Compare Mode Functionality: [4]
- Accessing GP Timer Registers: [5]
- Writing to Timer Registers: [6]

General-Purpose Timers Register Manual
- GP Timer Register Mapping Summary: [7] [8] [9]

Table 13-26. TWER

Address Offset	0x020		
Physical Address	0x4831 8020	**Instance**	GPTIMER1
	0x4903 2020		GPTIMER1
	0x4903 4020		GPTIMER1
	0x4903 6020		GPTIMER1
	0x4903 8020		GPTIMER5
	0x4903 A020		GPTIMER5
	0x4903 C020		GPTIMER5
	0x4903 E020		GPTIMER5
	0x4904 0020		GPTIMER9
	0x4808 6020		GPTIMER9
	0x4808 8020		GPTIMER9
	0x4903 2020		GPT2
	0x4903 4020		GPT3
	0x4903 6020		GPT4
	0x4903 8020		GPT5
	0x4903 A020		GPT6
	0x4903 C020		GPT7
	0x4903 E020		GPT8
	0x4904 0020		GPT9
	0x4808 6020		GPT10
	0x4808 8020		GPT11
Description	This register controls (enable/disable) the wake-up feature on specific interrupt events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													TCAR_WUP_ENA	OVF_WUP_ENA	MAT_WUP_ENA

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reads return 0	R	0x00000000
2	TCAR_WUP_ENA	Enable capture wake-up	RW	0
		0x0: Disable capture wake-up.		

Bits	Field Name	Description	Type	Reset
		0x1: Enable capture wake-up.		
1	OVF_WUP_ENA	Enable overflow wake-up	RW	0
		0x0: Disable overflow wake-up.		
		0x1: Enable overflow wake-up.		
0	MAT_WUP_ENA	Enable match wake-up	RW	0
		0x0: Disable match wake-up.		
		0x1: Enable match wake-up.		

Table 13-27. Register Call Summary for Register TWER

General-Purpose Timers

- Wake-Up Capability: [0] [1]
- GP Timers Functional Description: [2]
- Accessing GP Timer Registers: [3]
- Writing to Timer Registers: [4]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [5] [6] [7]

Table 13-28. TCLR

Address Offset	0x024		
Physical Address	0x4831 8024	**Instance**	GPTIMER1
	0x4903 2024		GPTIMER1
	0x4903 4024		GPTIMER1
	0x4903 6024		GPTIMER1
	0x4903 8024		GPTIMER5
	0x4903 A024		GPTIMER5
	0x4903 C024		GPTIMER5
	0x4903 E024		GPTIMER5
	0x4904 0024		GPTIMER9
	0x4808 6024		GPTIMER9
	0x4808 8024		GPTIMER9
	0x4903 2024		GPT2
	0x4903 4024		GPT3
	0x4903 6024		GPT4
	0x4903 8024		GPT5
	0x4903 A024		GPT6
	0x4903 C024		GPT7
	0x4903 E024		GPT8
	0x4904 0024		GPT9
	0x4808 6024		GPT10
	0x4808 8024		GPT11
Description	This register controls optional features specific to the timer functionality.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																GPO_CFG	CAPT_MODE	PT	TRG		TCM		SCPWM	CE	PRE	PTV				AR	ST

Bits	Field Name	Description	Type	Reset
31:15	Reserved	Reads return 0.	R	0x00000
14	GPO_CFG	PWM output/event detection input pin direction control: 0x0: Configures the pin as an output (needed when PWM mode is required) 0x1: Configures the pin as an input (needed when capture mode is required)	RW	0
13	CAPT_MODE	Capture mode select bit (first/second) 0x0: Capture the first enabled capture event in TCAR1. 0x1: Capture the second enabled capture event in TCAR2.	RW	0
12	PT	Pulse or toggle select bit 0x0: Pulse modulation 0x1: Toggle modulation	RW	0
11:10	TRG	Trigger output mode 0x0: No trigger 0x1: Overflow trigger 0x2: Overflow and match trigger 0x3: Reserved	RW	0x0
9:8	TCM	Transition capture mode 0x0: No capture 0x1: Capture on rising edges of EVENT_CAPTURE pin. 0x2: Capture on falling edges of EVENT_CAPTURE pin. 0x3: Capture on both edges of EVENT_CAPTURE pin.	RW	0x0
7	SCPWM	Pulse-width-modulation output pin default setting when counter is stopped or trigger output mode is set to no trigger. 0x0: Default value of PWM_out output: 0 0x1: Default value of PWM_out output: 1	RW	0
6	CE	Compare enable 0x0: Compare disabled 0x1: Compare enabled	RW	0
5	PRE	Prescaler enable 0x0: Prescaler disabled 0x1: Prescaler enabled	RW	0
4:2	PTV	Trigger output mode 0x0: The timer counter is prescaled with the value: $2^{(PTV+1)}$. Example: PTV = 3, counter increases value (if started) after 16 functional clock periods.	RW	0x0
1	AR	Autoreload mode 0x0: One-shot mode overflow 0x1: Autoreload mode overflow	RW	0
0	ST	Start/stop timer control 0x0: Stop the timer 0x1: Start the timer	RW	0

Table 13-29. Register Call Summary for Register TCLR

General-Purpose Timers

- Timer Mode Functionality: [0] [1] [2] [3] [4] [5] [6]
- Capture Mode Functionality: [7] [8] [9] [10] [11] [12] [13] [14] [15]
- Compare Mode Functionality: [16] [17] [18]
- Prescaler Functionality: [19] [20]
- Pulse-Width Modulation: [21] [22] [23] [24] [25] [26] [27] [28] [29] [30]
- Timer Counting Rate: [31] [32] [33] [34] [35] [36] [37]
- Accessing GP Timer Registers: [38]
- Writing to Timer Registers: [39]
- Write Posting Synchronization Mode: [40]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [41] [42] [43] [44]

Table 13-30. TCRR

Address Offset	0x028		
Physical Address	0x4831 8028	**Instance**	GPTIMER1
	0x4903 2028		GPTIMER1
	0x4903 4028		GPTIMER1
	0x4903 6028		GPTIMER1
	0x4903 8028		GPTIMER5
	0x4903 A028		GPTIMER5
	0x4903 C028		GPTIMER5
	0x4903 E028		GPTIMER5
	0x4904 0028		GPTIMER9
	0x4808 6028		GPTIMER9
	0x4808 8028		GPTIMER9
	0x4903 2028		GPT2
	0x4903 4028		GPT3
	0x4903 6028		GPT4
	0x4903 8028		GPT5
	0x4903 A028		GPT6
	0x4903 C028		GPT7
	0x4903 E028		GPT8
	0x4904 0028		GPT9
	0x4808 6028		GPT10
	0x4808 8028		GPT11
Description	This register holds the value of the internal counter.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER_COUNTER																															

Bits	Field Name	Description	Type	Reset
31:0	TIMER_COUNTER	The value of the timer counter register	RW	0x00000000

Table 13-31. Register Call Summary for Register TCRR

General-Purpose Timers

- Wake-Up Capability: [0] [1]
- Timer Mode Functionality: [2] [3] [4] [5] [6] [7] [8] [9] [10]
- 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [11] [12] [13]
- Capture Mode Functionality: [14] [15]
- Compare Mode Functionality: [16] [17]
- Prescaler Functionality: [18] [19]
- Accessing GP Timer Registers: [20]
- Writing to Timer Registers: [21]
- Write Posting Synchronization Mode: [22] [23]
- Reading From Timer Counter Registers: [24]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [25] [26] [27]
- GP Timer Register Descriptions: [28] [29] [30]

Table 13-32. TLDR

Address Offset	0x02C		
Physical Address	0x4831 802C	**Instance**	GPTIMER1
	0x4903 202C		GPTIMER1
	0x4903 402C		GPTIMER1
	0x4903 602C		GPTIMER1
	0x4903 802C		GPTIMER5
	0x4903 A02C		GPTIMER5
	0x4903 C02C		GPTIMER5
	0x4903 E02C		GPTIMER5
	0x4904 002C		GPTIMER9
	0x4808 602C		GPTIMER9
	0x4808 802C		GPTIMER9
	0x4903 202C		GPT2
	0x4903 402C		GPT3
	0x4903 602C		GPT4
	0x4903 802C		GPT5
	0x4903 A02C		GPT6
	0x4903 C02C		GPT7
	0x4903 E02C		GPT8
	0x4904 002C		GPT9
	0x4808 602C		GPT10
	0x4808 802C		GPT11
Description	This register holds the timer load values.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	LOAD_VALUE	The value of the timer load register	RW	0x00000000

Table 13-33. Register Call Summary for Register TLDR

General-Purpose Timers • Timer Mode Functionality: [0] [1] [2] • 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] • Prescaler Functionality: [13] [14] • Pulse-Width Modulation: [15] [16] [17] • Timer Counting Rate: [18] [19] [20] [21] [22] • Accessing GP Timer Registers: [23] • Writing to Timer Registers: [24] • Write Posting Synchronization Mode: [25]
General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [26] [27] [28]

Table 13-34. TTGR

Address Offset	0x030		
Physical Address	0x4831 8030	**Instance**	GPTIMER1
	0x4903 2030		GPTIMER1
	0x4903 4030		GPTIMER1
	0x4903 6030		GPTIMER1
	0x4903 8030		GPTIMER5
	0x4903 A030		GPTIMER5
	0x4903 C030		GPTIMER5
	0x4903 E030		GPTIMER5
	0x4904 0030		GPTIMER9
	0x4808 6030		GPTIMER9
	0x4808 8030		GPTIMER9
	0x4903 2030		GPT2
	0x4903 4030		GPT3
	0x4903 6030		GPT4
	0x4903 8030		GPT5
	0x4903 A030		GPT6
	0x4903 C030		GPT7
	0x4903 E030		GPT8
	0x4904 0030		GPT9
	0x4808 6030		GPT10
	0x4808 8030		GPT11
Description	This register triggers a counter reload of timer by writing any value in it.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTGR_ VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	TTGR_ VALUE	The value of the trigger register. During reads, it always returns 0xFFFFFFFF.	RW	0xFFFFFFFF

Table 13-35. Register Call Summary for Register TTGR

General-Purpose Timers • Timer Mode Functionality: [0] [1] • Prescaler Functionality: [2] • Accessing GP Timer Registers: [3] • Writing to Timer Registers: [4] • Write Posting Synchronization Mode: [5]

Table 13-35. Register Call Summary for Register TTGR (continued)

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [6] [7] [8]

Table 13-36. TWPS

Address Offset	0x034		
Physical Address	0x4831 8034	**Instance**	GPTIMER1
	0x4903 2034		GPTIMER1
	0x4903 4034		GPTIMER1
	0x4903 6034		GPTIMER1
	0x4903 8034		GPTIMER5
	0x4903 A034		GPTIMER5
	0x4903 C034		GPTIMER5
	0x4903 E034		GPTIMER5
	0x4904 0034		GPTIMER9
	0x4808 6034		GPTIMER9
	0x4808 8034		GPTIMER9
	0x4903 2034		GPT2
	0x4903 4034		GPT3
	0x4903 6034		GPT4
	0x4903 8034		GPT5
	0x4903 A034		GPT6
	0x4903 C034		GPT7
	0x4903 E034		GPT8
	0x4904 0034		GPT9
	0x4808 6034		GPT10
	0x4808 8034		GPT11
Description	This register indicates if a Write-Posted is pending.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						W_PEND_TOWR	W_PEND_TOCR	W_PEND_TCVR	W_PEND_TNIR	W_PEND_TPIR	W_PEND_TMAR	W_PEND_TTGR	W_PEND_TLDR	W_PEND_TCRR	W_PEND_TCLR

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Reads return 0.	R	0x0000000
9	W_PEND_TOWR	Write pending for register GPT_TOWR	R	0
		0x0: Overflow wrapping register write not pending		
		0x1: Overflow wrapping register write pending		
	Reserved for instances 3, 4, 5, 6, 7, 8, 9, 11, 12	Read returns reset value.	R	0
8	W_PEND_TOCR	Write pending for register GPT_TOCR	R	0
		0x0: Overflow counter register write not pending		
		0x1: Overflow counter register write pending		
	Reserved for instances 3, 4, 5, 6, 7, 8, 9, 11, 12	Read returns reset value.	R	0
7	W_PEND_TCVR	Write pending for register GPT_TCVR	R	0
		0x0: Counter value register write not pending		
		0x1: Counter value register write pending		

Bits	Field Name	Description	Type	Reset
	Reserved for instances 3, 4, 5, 6, 7, 8, 9, 11, 12	Read returns reset value.	R	0
6	W_PEND_TNIR	Write pending for register GPT_TNIR 0x0: Negative increment register write not pending 0x1: Negative increment register write pending	R	0
	Reserved for instances 3, 4, 5, 6, 7, 8, 9, 11, 12	Read returns reset value.	R	0
5	W_PEND_TPIR	Write pending for register GPT_TPIR 0x0: Positive increment register write not pending 0x1: Positive increment register write pending	R	0
	Reserved for instances 3, 4, 5, 6, 7, 8, 9, 11, 12	Read returns reset value.	R	0
4	W_PEND_TMAR	Write pending for register GPT_TMAR 0x0: Match register write not pending 0x1: Match register write pending	R	0
3	W_PEND_TTGR	Write pending for register GPT_TTGR 0x0: Trigger register write not pending 0x1: Trigger register write pending	R	0
2	W_PEND_TLDR	Write pending for register GPT_TLDR 0x0: Load register write not pending 0x1: Load register write pending	R	0
1	W_PEND_TCRR	Write pending for register GPT_TCRR 0x0: Counter register write not pending 0x1: Counter register write pending	R	0
0	W_PEND_TCLR	Write pending for register GPT_TCLR 0x0: Control register write not pending 0x1: Control register write pending	R	0

Table 13-37. Register Call Summary for Register TWPS

General-Purpose Timers • Write Posting Synchronization Mode: [0]
General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [1] [2] [3]

Table 13-38. TMAR

Address Offset	0x038		
Physical Address	0x4831 8038	**Instance**	GPTIMER1
	0x4903 2038		GPTIMER1
	0x4903 4038		GPTIMER1
	0x4903 6038		GPTIMER1
	0x4903 8038		GPTIMER5
	0x4903 A038		GPTIMER5
	0x4903 C038		GPTIMER5
	0x4903 E038		GPTIMER5
	0x4904 0038		GPTIMER9
	0x4808 6038		GPTIMER9
	0x4808 8038		GPTIMER9
	0x4903 2038		GPT2
	0x4903 4038		GPT3
	0x4903 6038		GPT4
	0x4903 8038		GPT5
	0x4903 A038		GPT6
	0x4903 C038		GPT7
	0x4903 E038		GPT8
	0x4904 0038		GPT9
	0x4808 6038		GPT10
	0x4808 8038		GPT11
Description	This register holds the value to be compared with the counter value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPARE_ VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	COMPARE_ VALUE	The value of the match register	RW	0x00000000

Table 13-39. Register Call Summary for Register TMAR

General-Purpose Timers

- Wake-Up Capability: [0]
- Compare Mode Functionality: [1] [2] [3]
- Pulse-Width Modulation: [4] [5] [6]
- Accessing GP Timer Registers: [7]
- Writing to Timer Registers: [8]
- Write Posting Synchronization Mode: [9]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [10] [11] [12]

Table 13-40. TCAR1

Address Offset	0x03C		
Physical Address	0x4831 803C	**Instance**	GPTIMER1
	0x4903 203C		GPTIMER1
	0x4903 403C		GPTIMER1
	0x4903 603C		GPTIMER1
	0x4903 803C		GPTIMER5
	0x4903 A03C		GPTIMER5
	0x4903 C03C		GPTIMER5
	0x4903 E03C		GPTIMER5
	0x4904 003C		GPTIMER9
	0x4808 603C		GPTIMER9
	0x4808 803C		GPTIMER9
	0x4903 203C		GPT2
	0x4903 403C		GPT3
	0x4903 603C		GPT4
	0x4903 803C		GPT5
	0x4903 A03C		GPT6
	0x4903 C03C		GPT7
	0x4903 E03C		GPT8
	0x4904 003C		GPT9
	0x4808 603C		GPT10
	0x4808 803C		GPT11
Description	This register holds the first captured value of the counter register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAPTURE_ VALUE1																															

Bits	Field Name	Description	Type	Reset
31:0	CAPTURE_ VALUE1	The value of first captured counter register	R	0x00000000

Table 13-41. Register Call Summary for Register TCAR1

General-Purpose Timers

- Capture Mode Functionality: [0] [1] [2] [3] [4] [5] [6] [7] [8]
- Reading From Timer Counter Registers: [9]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [10] [11] [12]
- GP Timer Register Descriptions: [13]

Table 13-42. TSICR

Address Offset	0x040		
Physical Address	0x4831 8040	**Instance**	GPTIMER1
	0x4903 2040		GPTIMER1
	0x4903 4040		GPTIMER1
	0x4903 6040		GPTIMER1
	0x4903 8040		GPTIMER5
	0x4903 A040		GPTIMER5
	0x4903 C040		GPTIMER5
	0x4903 E040		GPTIMER5
	0x4904 0040		GPTIMER9
	0x4808 6040		GPTIMER9
	0x4808 8040		GPTIMER9
	0x4903 2040		GPT2
	0x4903 4040		GPT3
	0x4903 6040		GPT4
	0x4903 8040		GPT5
	0x4903 A040		GPT6
	0x4903 C040		GPT7
	0x4903 E040		GPT8
	0x4904 0040		GPT9
	0x4808 6040		GPT10
	0x4808 8040		GPT11
Description	This register contains the bits that control the interface between the L4 interface and functional clock domains-posted mode and functional SW reset.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													POSTED	SFT	Reserved

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reads return 0.	R	0x00000000
2	POSTED	Posted mode selection	RW	1
		0x0: Non-posted mode selected		
		0x1: Posted mode selected		
1	SFT	Reset software functional registers. This bit is automatically reset by the hardware. During reads, it always returns 0.	RW	0
		0x0: Normal functional mode		
		0x1: The functional registers are reset.		
0	Reserved	Reads return 0.	R	0

Table 13-43. Register Call Summary for Register TSICR

General-Purpose Timers

- Software Reset: [0] [1]
- Accessing GP Timer Registers: [2]
- Writing to Timer Registers: [3]
- Write Posting Synchronization Mode: [4]
- Write Nonposting Synchronization Mode: [5]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [6] [7] [8]

Table 13-44. TCAR2

Address Offset	0x044		
Physical Address	0x4831 8044	**Instance**	GPTIMER1
	0x4903 2044		GPTIMER1
	0x4903 4044		GPTIMER1
	0x4903 6044		GPTIMER1
	0x4903 8044		GPTIMER5
	0x4903 A044		GPTIMER5
	0x4903 C044		GPTIMER5
	0x4903 E044		GPTIMER5
	0x4904 0044		GPTIMER9
	0x4808 6044		GPTIMER9
	0x4808 8044		GPTIMER9
	0x4903 2044		GPT2
	0x4903 4044		GPT3
	0x4903 6044		GPT4
	0x4903 8044		GPT5
	0x4903 A044		GPT6
	0x4903 C044		GPT7
	0x4903 E044		GPT8
	0x4904 0044		GPT9
	0x4808 6044		GPT10
	0x4808 8044		GPT11
Description	This register holds the second captured value of the counter register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAPTURE_ VALUE2																															

Bits	Field Name	Description	Type	Reset
31:0	CAPTURE_ VALUE2	The value of second captured counter register	R	0x00000000

Table 13-45. Register Call Summary for Register TCAR2

General-Purpose Timers

- Capture Mode Functionality: [0] [1] [2] [3] [4] [5]
- Reading From Timer Counter Registers: [6]

General-Purpose Timers Register Manual

- GP Timer Register Mapping Summary: [7] [8] [9]
- GP Timer Register Descriptions: [10]

Table 13-46. TPIR

Address Offset	0x048		
Physical Address	0x4831 8048	**Instance**	GPTIMER1
	0x4903 2048		GPTIMER1
	0x4808 6048		GPTIMER9
	0x4903 2048		GPT2
	0x4808 6048		GPT10
Description	This register is used for 1 ms tick generation. The TPIR register holds the value of the positive increment. The value of this register is added with the value of the TCVR to define whether next value loaded in TCRR will be the sub-period value or the over-period value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POSITIVE_INC_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	POSITIVE_INC_VALUE	The value of positive increment.	RW	0x00000000

Table 13-47. Register Call Summary for Register TPIR

General-Purpose Timers • 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [0] [1] [2] [3] [4] [5] • Accessing GP Timer Registers: [6] • Writing to Timer Registers: [7] • Write Posting Synchronization Mode: [8]
General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [9] [10] • GP Timer Register Descriptions: [11]

Table 13-48. TNIR

Address Offset	0x04C		
Physical Address	0x4831 804C 0x4903 204C 0x4808 604C	**Instance**	GPTIMER1 GPTIMER1 GPTIMER9
	0x4903 204C		GPT2
	0x4808 604C		GPT10
Description	This register is used for 1 ms tick generation. The TNIR register holds the value of the negative increment. The value of this register is added with the value of the TCVR to define whether next value loaded in TCRR will be the sub-period value or the over-period value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NEGATIVE_INC_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	NEGATIVE_INC_VALUE	The value of negative increment.	RW	0x00000000

Table 13-49. Register Call Summary for Register TNIR

General-Purpose Timers • 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [0] [1] [2] [3] [4] [5] • Accessing GP Timer Registers: [6] • Writing to Timer Registers: [7] • Write Posting Synchronization Mode: [8]
General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [9] [10] • GP Timer Register Descriptions: [11]

Table 13-50. TCVR

Address Offset	0x050		
Physical Address	0x4831 8050 0x4903 2050 0x4808 6050	**Instance**	GPTIMER1 GPTIMER1 GPTIMER9
	0x4903 2050		GPT2
	0x4808 6050		GPT10
Description	This register is used for 1 ms tick generation. The TCVR register defines whether next value loaded in TCRR will be the sub-period value or the over-period value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTER_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	COUNTER_VALUE	The value of CVR counter.	RW	0x00000000

Table 13-51. Register Call Summary for Register TCVR

General-Purpose Timers
- 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [0] [1]
- Accessing GP Timer Registers: [2]
- Writing to Timer Registers: [3]
- Write Posting Synchronization Mode: [4]

General-Purpose Timers Register Manual
- GP Timer Register Mapping Summary: [5] [6]
- GP Timer Register Descriptions: [7] [8] [9]

Table 13-52. TOCR

Address Offset	0x054		
Physical Address	0x4831 8054 0x4903 2054 0x4808 6054	**Instance**	GPTIMER1 GPTIMER1 GPTIMER9
	0x4903 2054		GPT2
	0x4808 6054		GPT10
Description	This register is used to mask the tick interrupt for a selected number of ticks.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OVF_COUNTER_VALUE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Reads return 0.	RW	0x00
23:0	OVF_COUNTER_VALUE	The number of overflow events.	RW	0x00000000

Table 13-53. Register Call Summary for Register TOCR

General-Purpose Timers
- 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [0] [1] [2] [3] [4]
- Accessing GP Timer Registers: [5]
- Writing to Timer Registers: [6]
- Write Posting Synchronization Mode: [7]

Table 13-53. Register Call Summary for Register TOCR (continued)

General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [8] [9]

Table 13-54. TOWR

Address Offset	0x058		
Physical Address	0x4831 8058 0x4903 2058 0x4808 6058	**Instance**	GPTIMER1 GPTIMER1 GPTIMER9
	0x4903 2058		GPT2
	0x4808 6058		GPT10
Description	This register holds the number of masked overflow interrupts.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OVF_WRAPPING_VALUE																							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Reads return 0.	RW	0x00
23:0	OVF_WRAPPING_VALUE	The number of masked interrupts.	RW	0x00000000

Table 13-55. Register Call Summary for Register TOWR

General-Purpose Timers • 1-ms Tick Generation (Only GPTIMER1, GPTIMER2, and GPTIMER10): [0] [1] [2] • Accessing GP Timer Registers: [3] • Writing to Timer Registers: [4] • Write Posting Synchronization Mode: [5]
General-Purpose Timers Register Manual • GP Timer Register Mapping Summary: [6] [7]

13.4 Watchdog Timers

13.4.1 WDTs Overview

The device includes two instances of the 32-bit WDT: WDT2 and WDT3. Figure 13-14 shows how each timer is connected in the device.

NOTE: WDTi (where i is the watchdog timer instance: i = 2 or 3) stands for the following:
- WDT2: Watchdog timer 2, also called MPU watchdog timer
- WDT3: Watchdog timer 3, also called IVA2 watchdog timer

Each WDT is an upward counter capable of generating both a pulse on the reset pin and an interrupt to the device system modules following an overflow condition. The MPU WDT serves resets to the PRCM module (its interrupt outputs are unused), and the IVA2 WDT serves watchdog interrupts to the MPU (its reset outputs are unused).

The WDTs can be accessed, loaded, and cleared by registers through the L4 interface. The MPU and IVA2 WDTs have the 32-kHz clock for their timer clock input.

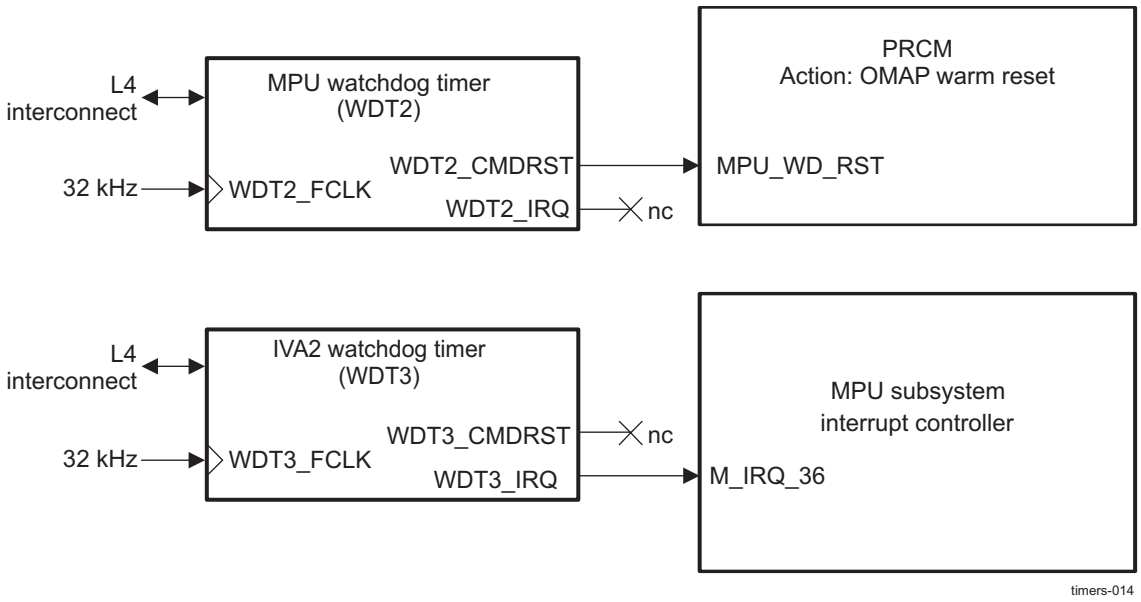
The MPU WDT directly generates a warm reset condition on overflow. The IVA2 WDT generates an MPU interrupt condition on overflow, which can be used by the application software via the PRCM to indirectly trigger a reset condition (that is, to the IVA2 subsystem).

The MPU WDT connects to a single target agent port on the L4 interconnect.

Table 13-56. WD Timers Default State for GP and EMU devices

Timer	Device			
	EMU		GP	
MPU WDTIMER2	Enabled	Running	Enabled	Running
IVA2 WDTIMER3	Enabled	NOT Running	Enabled	NOT Running

Figure 13-14. WDTs Block Diagram



13.4.1.1 WDT Features

The following are the main features of the WDT controllers:

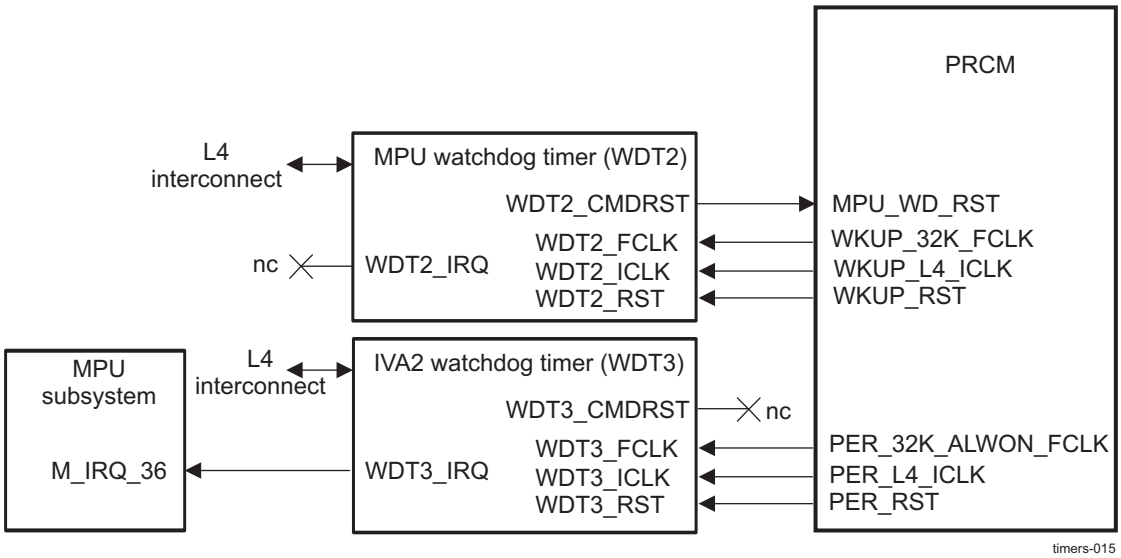
- L4 slave interface support:

  - 32-bit data bus width
  - 32-/16-bit access supported
  - 8-bit access not supported
  - 11-bit address bus width
  - Burst mode not supported
  - Only posted write/read internal resynchronization mode supported
- Free-running 32-bit upward counter
- Programmable divider clock source (2^n with n = [0:7])
- On-the-fly read/write register (while counting)
- Subset programming model of the GP timer
- WDTs are reset either on power-on or after a warm reset.
- Reset or interrupt actions when a timer overflow condition occurs
- WDT generates either a reset or an interrupt in its hardware integration (WDT2, or WDT3).

13.4.2 WDT Integration

Figure 13-15 shows the integration of the WDT in the device.

Figure 13-15. WDT Integration



13.4.2.1 Clocking, Reset, and Power-Management Scheme

13.4.2.1.1 Clock Management

There are two clock domains in the WDTs:

- Functional clock domain: WDTi_FCLK is the WDT functional clock. It is used to clock the WDT internal logic.
- Interface clock domain: WDTi_ICLK is the WDT interface clock. It is used to synchronize the WDT L4 port to the L4 interconnect. All accesses from the interconnect are synchronous to WDTi_ICLK.

Table 13-57 lists the the source clocks for each WDT in the device. For more information on clock control and domains, see , *Power, Reset, and Clock Management*.

Table 13-57. Clock, Power, and Reset Domains for WDTs

Timer	Interface Clock	Functional Clock	Power Domain
MPU WDT	WKUP_L4_ICLK	WKUP_32K_FCLK	WKUP
IVA2 WDT	PER_L4_ICLK	PER_32K_ALWON_FCLK	PER

From a global system power-management perspective, when one or both of the WDT clocks is no longer required, the WDTs can be deactivated at the PRCM level in the corresponding registers. Table 13-58 lists the WDT PRCM clock control bits.

Table 13-58. WDT PRCM Clock Control Bits

Name	Associated PRCM Clock Output	Enable Bit	Autoidle Bit
WDT2_FCLK	WKUP_32K_FCLK	PRCM.CM_FCLKEN_WKUP[5] EN_WDT2 bit	N/A
WDT3_FCLK	PER_32K_ALWON_FCLK	PRCM.CM_FCLKEN_PER[12] EN_WDT3 bit	N/A
WDT2_ICLK	WKUP_L4_ICLK	PRCM.CM_ICLKEN_WKUP[5] EN_WDT2 bit	PRCM.CM_AUTOIDLE_WKUP[5] AUTO_WDT2 bit
WDT3_ICLK	PER_L4_ICLK	PRCM.CM_ICLKEN_PER[12] EN_WDT3 bit	PRCM.CM_AUTOIDLE_PER[12] AUTO_WDT3 bit

NOTE:

- The PRCM function clock outputs are gated at the PRCM level, assuming the modules that share it have been disabled in the corresponding register. Disabling the WDTs is a necessary but not sufficient action.The clock is effectively out when all PRCM conditions are fulfilled. For the other actions to be taken, see , *Power, Reset, and Clock Management*.
- The PRCM interface clock outputs are gated at the PRCM level, assuming the modules that share it have been disabled in the corresponding register. Disabling the WDTs is a necessary but not sufficient condition. The clock is effectively out when all PRCM conditions are fulfilled.For the other actions to be taken, see , *Power, Reset, and Clock Management*.
- The PRCM AUTOIDLE bits are used to link/unlink the WDTs from the clock domain transitions related to the GPTi_ICLK clocks.
- For further details about source clocks gating and domain transitions, see , *Power, Reset, and Clock Management.*

At the PRCM level, when the conditions to shut off the PRCM functional or interface output clocks are met (see , *Power, Reset, and Clock Management,* for details), the PRCM automatically launches a hardware handshake protocol to ensure the WDT is ready to have its clocks switched off. Namely, the PRCM module asserts an IDLE request to the WDT.

Although this handshake is a hardware function and out of software control, the way the WDT acknowledges the PRCM IDLE request is configurable through the WDTi.WD_SYSCONFIG[4:3] IDLEMODE bit. Table 13-59 lists the IDLEMODE settings and the related acknowledgement modes.

Table 13-59. IDLEMODE Settings

IDLEMODE Value	Selected Mode	Description
00	Force-idle	The WDT acknowledges unconditionally the IDLE request from the PRCM module, regardless of its internal operations. This mode must be used carefully, because it does not prevent loss of data when the clock is switched off.
01	No-idle	The WDT never acknowledges an IDLE request from the PRCM module. This mode is safe from a module point of view, because it ensures the clocks remain active; it is not efficient from a power-saving perspective, however, because it does not allow the PRCM output clock to be shut off and thus the power domain to be set to a lower power state.

Table 13-59. IDLEMODE Settings (continued)

IDLEMODE Value	Selected Mode	Description
10	Smart-idle	The WDT acknowledges the IDLE request, basing its decision on its internal activity. The acknowledge signal is asserted only when all pending transactions and IRQs requests are treated. This is the best approach for efficient system power management.
11	Reserved	

When configured in smart-idle mode, the WDT also offers an additional granularity on WDTi_FCLK and WDTi_ICLK gating. The WDTi.WD_SYSCONFIG[9:8] CLOCKACTIVITY bit field is used to determine which clock will be shut down (WDTi_FCLK, WDTi_ICLK, neither of them, or both of them).

The CLOCKACTIVITY setting is used internally to the WDT to determine the part of the module on which the conditions to acknowledge the PRCM IDLE request will be tested. For example, if WDTi_FCLK is not to be shut down on a PRCM IDLE request, the WDT considers only WDTi_ICLK and the associated pending activities before acknowledging the request.

Some WDT features are associated with WDTi_ICLK, and others are associated with WTDi_FCLK. Using the CLOCKACTIVITY setting along with the smart-idle mode ensures that the features associated with the clock that will remain active are always enabled, even if the WDT acknowledges an IDLE request.

Table 13-60 lists the CLOCKACTIVITY settings.

Table 13-60. CLOCKACTIVITY Settings

CLOCKACTIVITY Value	WDTi_ICLK Effects	WDTi_FCLK Effects	Description
00	OFF	OFF	Both WDTi_ICLK and WDTi_FCLK are considered for generating the acknowledge. This setting also means both WDTi_FCLK and WDTi_ICLK are likely to be shut down on a PRCM IDLE request.
01	OFF	ON	WDTi_FCLK is not shut down on a PRCM IDLE request; only WDTi_ICLK is concerned.
10	ON	OFF	WDTi_ICLK is not shut down on a PRCM IDLE request; only WDTi_FCLK is concerned.
11	ON	ON	None of the clocks are shut down. Therefore, the WDT can potentially acknowledge the IDLE request without checking the internal functionalities linked to its clocks.

CAUTION

The PRCM module does *not* have any hardware means to read the CLOCKACTIVITY settings. The software must ensure consistent programming between the WDT CLOCKACTIVITY and the PRCM functional clock and interface clock control bits. If the WDT is disabled in both the CM_FCLKEN and CM_ICLKEN PRCM registers while CLOCKACTIVITY is set to 11, nothing prevents the PRCM module from asserting its IDLE request, which is acknowledged regardless of the features associated with the WDT clocks. This can lead to unpredictable behavior.

13.4.2.1.2 Reset and Power Management

The MPU WDT (WDT2) belong to the WKUP power domain. As part of that domain, this WDT is sensitive to a WKUP_RST signal issued by the PRCM module. For further details about the WKUP power domain implementation and WKUP_RST signal, see , *Power, Reset, and Clock Management.*

The IVA WDT (WDT3) belongs to the PER power domain. As part of that domain, WDT3 is sensitive to the PER_RST signal issued by the PRCM module. For further details about the PER power domain implementation and PER_RST signal, see , *Power, Reset, and Clock Management.*

NOTE: WDT2 is reset on power-on or after a warm reset, and then it starts counting.

WDT3 is reset either on power-on or after a warm reset, and then it doesn't start counting.

13.4.2.2 Interrupts

Table 13-61 shows interrupt mapping from the three WDTs to the MPU.

Table 13-61. WDT Interrupt Names and Processor IRQ Mapping

Timer	Interrupt Name	Mapping	Comments
MPU WDT	WDT2_IRQ	Not connected	
IVA2 WDT	WDT3_IRQ	M_IRQ_36	IVA2 WDT overflow

13.4.3 WDTs Functional Description

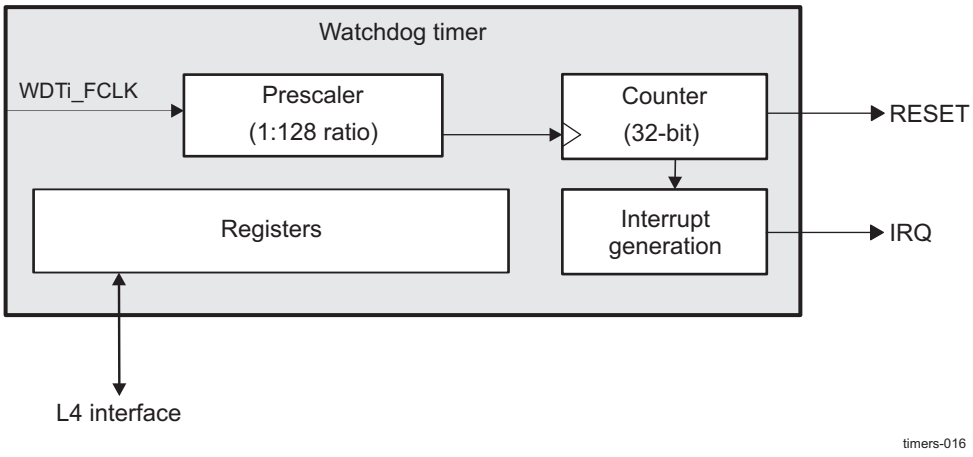
13.4.3.1 General WDT Operation

The WDTs are based on an upward 32-bit counter coupled with a prescaler. The counter overflow is signaled through two independent signals: a simple reset signal and an interrupt signal, both active low. The use of these signals depends on whether they are connected or not. For this information, see Figure 13-14. The interrupt generation mechanism is controlled through the WDTi.WIER and WDTi.WISR registers.

The prescaler ratio can be set between 1 and 128 by accessing the WDTi.WCLR[4:2] PTV and WDTi.WCLR[5] PRE fields of the watchdog control register (WDTi.WCLR).

The current timer value can be accessed on-the-fly by reading the WDT counter register (WDTi.WCRR), modified by accessing the WDT load register (WDTi.WLDR) (no on-the-fly update), or reloaded by following a specific reload sequence on the WDT trigger register (WDTi.WTGR). A start/stop sequence applied to the WDT start/stop register (WDTi.WSPR) can start and stop the WDT.

Figure 13-16. 32-Bit WDT Functional Block Diagram



13.4.3.2 Reset Context

After reset, the WDTs are enabled. Table 13-62 lists the default reset values of the three WDT load registers (WDTi.WLDR) and prescaler ratios (WDTi.WCLR[4:2] PTV field). To get these values, software must read the corresponding WDTi.WCLR[4:2] PTV fields and the 32-bit register to retrieve the static configuration of the module.

Table 13-62. Count and Prescaler Default Reset Values

Timer	WLDR Reset Value	PTV Reset Value
OMAP WDT (WDT2)	0xFFFB 0000	0
IVA2 WDT (WDT3)	0xFFFB 0000	0

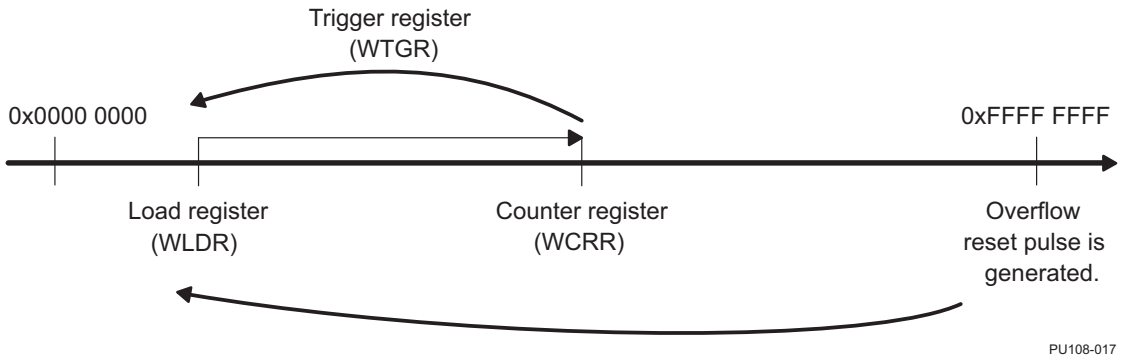
13.4.3.3 Overflow/Reset Generation

When the WDT counter register (WDTi.WCRR) overflows, an active-low reset pulse is generated to the PRCM module. This pulse is one prescaled timer clock cycle wide and occurs at the same time as the timer counter overflow.

After reset generation, the counter is automatically reloaded with the value stored in the watchdog load register (WDTi.WLDR) and the prescaler is reset (the prescaler ratio remains unchanged). Then, after the reset pulse output has been generated, the timer counter begins incrementing again.

Figure 13-17 shows a general functional view of the WDT.

Figure 13-17. WDT General Functional View



13.4.3.4 Prescaler Value/Timer Reset Frequency

Each WDT is composed of a prescaler stage and a timer counter.

The timer rate is defined by the following values:

- Value of the prescaler fields (the WDTi.WCLR[5] PRE bit and the WDTi.WCLR[4:2] PTV field)
- Value loaded into the timer load register (WDTi.WLDR)

The prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage. The ratio is managed by accessing the ratio definition field (the WDTi.WCLR[4:2] PTV field) and is enabled with the WDTi.WCLR[5] PRE bit.

Table 13-63 lists the prescaler clock ratio values.

Table 13-63. Prescaler Clock Ratios

PRE Bit (in WDTi.WCLR Register)	PTV Bits (in WDTi.WCLR Register)	Clock Divider (PS)
0	X	1
1	0	1
1	1	2
1	2	4
1	3	8
1	4	16
1	5	32
1	6	64
1	7	128

The watchdog timer overflow rate is expressed by: OVF_Rate = (0xFFFF FFFF - WDTn.WLDR + 1) * (wd-functional clock period) * PS

With wd-functional clock period = 1 / (wd-functional clock frequency) and PS = $2^{(PTV)}$.

CAUTION

Internal resynchronization causes any software write to GPTn.WSPR to have some latency before WSPR is updated with the programmed value:

1.5 * functional clock cycles ≤ write_WSPR_latency ≤ 2.5 * functional clock cycles

Remember to take this latency into account whenever the watchdog must be started or stopped.

For example, for a timer clock input of 32 kHz with a prescaler ratio value of 0x1 (clock divided by 2) and WDTi.WCLR[5] PRE bit = 1 (clock divider enabled), the reset period is as listed in Table 13-64.

Table 13-64. Reset Period Examples

WDTi.WLDR Value	Reset Period
0x0000 0000	74 h 56 min
0xFFFF 0000	4 s
0xFFFF FFF0	1 ms
0xFFFF FFFF	62.5 us

CAUTION

- Ensure that the reloaded value allows the correct operation of the application. When a WDT is enabled, the software must periodically trigger a reload before the counter overflows. Hence, the WDTi.WLDR[31:0] value must be chosen according to the ongoing activity preceding the watchdog reload.
- Due to design reasons, WDTi.WLDR[31:0] = 0xFFFF FFFF is a special case, although such a WDTi.WLDR value is meaningless. When WDTi.WLDR is programmed with the overflow value, a triggering event generates a reset/interrupt one functional clock cycle later, even if the WDT is stopped.

Table 13-65 lists the default reset periods for the WDTs.

Table 13-65. Default WDT Time Periods

WDTs	Clock Source	Default Reset Period
OMAP/IVA2 WDTs	32 kHz	10 s

13.4.3.5 Triggering a Timer Reload

To reload the timer counter and reset the prescaler before reaching overflow, a reload command is executed by accessing the WDT trigger register (WDTi.WTGR) using a specific reload sequence.

The specific reload sequence is performed whenever the written value on the WDT trigger register (WDTi.WTGR) differs from its previous value. In this case, reload is executed in the same way as an overflow autoreload, but without a reset pulse generation.

The timer counter is loaded with the WDT load register value (the WDTi.WLDR[31:0] TIMER_LOAD field), and the prescaler is reset.

13.4.3.6 Start/Stop Sequence for WDTs (Using WDTi.WSPR Register)

To start/stop a WDT, access must be made through the start/stop register (WDTi.WSPR) using a specific sequence.

To disable the timer, follow this sequence:

1. Write 0xXXXX AAAA in WDTi.WSPR.
2. Write 0xXXXX 5555 in WDTi.WSPR.

To enable the timer, follow this sequence:

1. Write 0xXXXX BBBB in WDTi.WSPR.
2. Write 0xXXXX 4444 in WDTi.WSPR.

All other write sequences on WDTi.WSPR have no effect on the start/stop feature of the module.

13.4.3.7 Modifying Timer Count/Load Values and Prescaler Setting

To modify the timer counter value (WDTi.WCRR), prescaler ratio (the WDTi.WCLR[4:2] PTV field), or load value (the WDTi.WLDR[31:0] TIMER_LOAD field), the WDT must be disabled by using the start/stop sequence (the WDTi.WSPR register).

After a write access, the load register value and prescaler ratio registers are updated immediately, but new values are considered only after the next consecutive counter overflow or after a new trigger command (WDTi.WTGR).

13.4.3.8 Watchdog Counter Register Access Restriction (WDTi.WCRR Register)

Because the WDTi.WCRR register is directly related to the timer counter value and is updated on the timer clock (WDTi_FCLK), a 32-bit shadow register is implemented to read a coherent value of the WDTi.WCRR register. The shadow register is updated by a 16-bit LSB read command.

NOTE: Although the L4 clock (WDTi_ICLK) is completely asynchronous with the timer clock (WDTi_FCLK), some synchronization is done to ensure that the WDTi.WCRR value is not read while it is being incremented.

When 32-bit read access is performed, the shadow register is not updated. Read access is made directly from the accessed register.

To ensure that a coherent value is read inside WDTi.WCRR, the first read access is to the lower 16 bits (offset = 0x08), followed by read access to the upper 16 bits (offset = 0x0A).

13.4.3.9 WDT Interrupt Generation

The WDT issues an overflow interrupt if this interrupt is enabled in the WDT interrupt enable register (WDTi.WIER[0] OVF_IT_ENA = 1). When the overflow occurs, the interrupt status bit (the WDTi.WISR[0] OVF_IT_FLAG bit) is set to 1. The output interrupt line (WDTi_IRQ) is asserted (active low) when both status (OVF_IT_FLAG) and enable (OVF_IT_ENA) flags are set to 1; the order is not relevant. Writing 1 in the enable bit (the status is already set at 1) also triggers the interrupt in the normal order (enable first, status after). The pending interrupt event is cleared when the set status bit is overwritten by a value of 1 by a write command in the WTDi.WISR register. Reading the WTDi.WISR register and writing the value back allows a fast acknowledge interrupt process.

NOTE: Writing 0 in the WDTi.WISR[0] OVF_IT_FLAG bit has no effect on it.

Because the interrupt event is generated on the functional clock domain (WDTi_FCLK), during the interrupt status register (WDTi.WISR) update, the two clock domains are resynchronized.

13.4.3.10 WDT Under Emulation

During emulation mode, the WDT can/cannot continue running, according on the value of the WDTi.WD_SYSCONFIG[5] EMUFREE bit of the system configuration register (WDTi.WD_SYSCONFIG).

- When EMUFREE is 1, WDT execution is not stopped and a reset pulse is still generated when overflow is reached.
- When EMUFREE is 0, the counters (prescaler/timer) are frozen and incrementation restarts after exiting from emulation mode.

13.4.3.11 Accessing Watchdog Timer Registers

Posted mode applies only to functional registers that require synchronization on/from the timer functional clock domain (WDTi_FCLK). For a write/read operation, the following registers are affected:

- WDT_WCLR
- WDT_WCRR
- WDT_WLDR
- WDT_WTGR
- WDT_WDLY
- WDT_WSPR

NOTE: To ensure completion of successive writes in a WDT2/WDT3 functional clock domain register, user software must poll the appropriate WDT_WWPS status bit.

NOTE: To ensure coherent reading when a read transaction immediately follows a write access to a WDT2/WDT3 functional clock domain register, user software must check the appropriate WDT_WWPS status bit.

The synchronous registers in the timer interface clock domain are not affected by the posted mode mechanism; the write/read operation is effective and acknowledged (command accepted) after one WDT_ICLK cycle from the command assertion. The synchronous registers in the timer interface clock domain are:

- WDT_WIDR
- WDT_WDSC
- WDT_WDST
- WDT_WIRQSTATRAW
- WDT_WIRQSTAT
- WDT_WIRQENSET
- WDT_WIRQENCLR
- WDT_WIRQWAKEEN
- WDT_WWPS

NOTE: Accesses to the synchronous registers in the WDT2 and WDT3 functional clock domains are posted.

13.5 Watchdog Timer Register Manual

All registers are 32 bits wide and are accessible through the L4 interface with 16-bit or 32-bit access (read/write).

13.5.1 Instance Summary

Table 13-66 lists the base address and address space for the WDT module instances. All timers are memory mapped to the L4 peripheral bus memory space.

Table 13-66. WDT Instance Summary

Module Name	Base Address	Size
WDTIMER2	0x4831 4000	4K bytes
WDTIMER3	0x4903 0000	4K bytes

13.5.2 WDT Register Mapping Summary

CAUTION

The WDT registers are limited to 32-bit and 16-bit data accesses; 8-bit access is not allowed and can corrupt register content.

Table 13-67 lists the WDT2 registers and Table 13-68 lists the WDT3 registers.

Table 13-67. WDTIMER2 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	WDTIMER2 Physical Address
WIDR	R	32	0x000	0x4831 4000
WD_SYSCONFIG	RW	32	0x010	0x4831 4010
WD_SYSSTATUS	R	32	0x014	0x4831 4014
WISR	RW	32	0x018	0x4831 4018
WIER	RW	32	0x01C	0x4831 401C
WCLR	RW	32	0x024	0x4831 4024
WCRR	RW	32	0x028	0x4831 4028
WLDR	RW	32	0x02C	0x4831 402C
WTGR	RW	32	0x030	0x4831 4030
WWPS	R	32	0x034	0x4831 4034
WSPR	RW	32	0x048	0x4831 4048

Table 13-68. WDTIMER3 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	WDTIMER3 Physical Address
WIDR	R	32	0x000	0x4903 0000
WD_SYSCONFIG	RW	32	0x010	0x4903 0010
WD_SYSSTATUS	R	32	0x014	0x4903 0014
WISR	RW	32	0x018	0x4903 0018
WIER	RW	32	0x01C	0x4903 001C
WCLR	RW	32	0x024	0x4903 0024
WCRR	RW	32	0x028	0x4903 0028
WLDR	RW	32	0x02C	0x4903 002C
WTGR	RW	32	0x030	0x4903 0030

Table 13-68. WDTIMER3 Register Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	WDTIMER3 Physical Address
WWPS	R	32	0x034	0x4903 0034
WSPR	RW	32	0x048	0x4903 0048

13.5.3 WDT Register Descriptions

Table 13-69 through Table 13-89 describe the WDT register bits.

Table 13-69. WIDR

Address Offset	0x000		
Physical Address	0x4831 4000 0x4903 0000	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0000		WDTIMER3
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								WD_REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0.	R	0x000000
7:0	WD_REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 13-70. Register Call Summary for Register WIDR

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [0] [1]

Table 13-71. WD_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4831 4010 0x4903 0010	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0010		WDTIMER3
Description	This register controls the various parameters of the L4 interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CLOCKACTIVITY		Reserved		EMUFREE		IDLEMODE		ENAWAKEUP		SOFTRESET		AUTOIDLE	

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Write 0s for future compatibility. Reads return 0.	R	0x0000000
9:8	CLOCKACTIVITY	Clock Activity selection bits. 0x0: Both clocks can be cut off 0x1: Only L4 clock must be kept active; timer clock can be cut off 0x2: Only timer clock must be kept active; L4 clock can be cut off 0x3: both clocks must be kept active	RW	0x0
7:6	Reserved	Write 0s for future compatibility. Reads return 0.	R	0x0000000
5	EMUFREE	Emulation mode 0x0: Timer counter frozen in emulation 0x1: Timer counter free-running in emulation	RW	0
4:3	IDLEMODE	Idle mode selection bits 0x0: Force Idle mode 0x1: No Idle mode 0x2: Smart Idle mode 0x3: Reserved	RW	0x0
2	ENAWAKEUP	Enable wakeup control bit 0x0: Wakeup mechanism is disabled 0x1: Wakeup mechanism is enabled	RW	0
1	SOFTRESET	Software reset. This bit is automatically reset by the hardware. During reads, it always return 0. 0x0: Normal mode 0x1: The module is reset.	RW	0
0	AUTOIDLE	L4 interconnect clock gating strategy 0x0: L4 interface clock is free-running. 0x1: Automatic L4 interface clock gating strategy is applied, based on the L4 interface activity.	RW	0

Table 13-72. Register Call Summary for Register WD_SYSCONFIG

Watchdog Timers
- Clock Management: [0] [1]
- WDT Under Emulation: [2] [3]

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [4] [5]

Table 13-73. WD_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4831 4014 0x4903 0014	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0014		WDTIMER3
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								Reserved							RESETDONE

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0.	R	0x0000000
7:1	Reserved	Reads return 0.	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset in ongoing. 0x1: Reset completed.	R	-

Table 13-74. Register Call Summary for Register WD_SYSSTATUS

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [0] [1]

Table 13-75. WISR

Address Offset	0x018		
Physical Address	0x4831 4018 0x4903 0018	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0014		WDTIMER3
Description	This register shows which interrupt events are pending inside the module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															OVF_IT_FLAG

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reads return 0.	R	0x00000000
0	OVF_IT_FLAG	Pending overflow interrupt status Read 0x0: No overflow interrupt pending Write 0x0: Status unchanged Read 0x1: Overflow interrupt pending Write 0x1: Status bit cleared	RW	0

Table 13-76. Register Call Summary for Register WISR

Watchdog Timers
- General WDT Operation: [0]
- WDT Interrupt Generation: [1] [2] [3] [4] [5]

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [6] [7]

Table 13-77. WIER

Address Offset	0x01C		
Physical Address	0x4831 401C 0x4903 001C	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0014		WDTIMER3
Description	This register shows controls (enable/disable) the interrupt events.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															OVF_IT_ENA

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reads return 0.	R	0x00000000
0	OVF_IT_ENA	Enable overflow interrupt 0x0: Disable overflow interrupt. 0x1: Enable overflow interrupt.	RW	0

Table 13-78. Register Call Summary for Register WIER

Watchdog Timers
- General WDT Operation: [0]
- WDT Interrupt Generation: [1]

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [2] [3]

Table 13-79. WCLR

Address Offset	0x024		
Physical Address	0x4831 4024 0x4903 0024	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0024		WDTIMER3
Description	This register controls the prescaler stage of the counter.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										PRE	PTV			Reserved	

Bits	Field Name	Description	Type	Reset
31:6	Reserved	Reads return 0.	R	0x0000000
5	PRE	Prescaler enable 0x0: Prescaler disabled 0x1: Prescaler enabled	RW	1
4:2	PTV	Prescaler value: The timer counter is prescaled with the value: pow(2,PTV) Example: PTV = 2: counter increases value is started after 4 functional clock periods.	RW	0x0
1:0	Reserved	Reads return 0.	R	0x0

Table 13-80. Register Call Summary for Register WCLR

Watchdog Timers
- General WDT Operation: [0] [1] [2]
- Reset Context: [3] [4]
- Prescaler Value/Timer Reset Frequency: [5] [6] [7] [8] [9] [10] [11]
- Modifying Timer Count/Load Values and Prescaler Setting: [12]

Table 13-80. Register Call Summary for Register WCLR (continued)

Watchdog Timer Register Manual • WDT Register Mapping Summary: [13] [14] • WDT Register Descriptions: [15]

Table 13-81. WCRR

Address Offset	0x028		
Physical Address	0x4831 4028 0x4903 0028	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0028		WDTIMER3
Description	This register holds the value of the internal counter.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER_COUNTER																															

Bits	Field Name	Description	Type	Reset
31:0	TIMER_COUNTER	The value of the timer counter register	RW	0x00000000

Table 13-82. Register Call Summary for Register WCRR

Watchdog Timers • General WDT Operation: [0] • Overflow/Reset Generation: [1] • Modifying Timer Count/Load Values and Prescaler Setting: [2] • Watchdog Counter Register Access Restriction (WDTi.WCRR Register): [3] [4] [5] [6]
Watchdog Timer Register Manual • WDT Register Mapping Summary: [7] [8] • WDT Register Descriptions: [9]

Table 13-83. WLDR

Address Offset	0x02C		
Physical Address	0x4831 402C 0x4903 002C	**Instance**	WDTIMER2 WDTIMER3
	0x4903 002C		WDTIMER3
Description	This register holds the timer load value.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMER_LOAD																															

Bits	Field Name	Description	Type	Reset
31:0	TIMER_LOAD	The value of the timer load register	RW	0xFFA6 0000 (WDT1) 0xFFFB 000 (WDT2 and 3)

Table 13-84. Register Call Summary for Register WLDR

Watchdog Timers • General WDT Operation: [0] • Reset Context: [1] [2] • Overflow/Reset Generation: [3] • Prescaler Value/Timer Reset Frequency: [4] [5] [6] [7] [8] [9] [10] • Triggering a Timer Reload: [11] • Modifying Timer Count/Load Values and Prescaler Setting: [12]
Watchdog Timer Register Manual • WDT Register Mapping Summary: [13] [14] • WDT Register Descriptions: [15]

Table 13-85. WTGR

Address Offset	0x030		
Physical Address	0x4831 4030 0x4903 0030	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0030		WDTIMER3
Description	Writing a different value than the one already written in this register causes a watchdog counter reload.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTGR_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	TTGR_VALUE	The value of the trigger register	RW	0x00000000

Table 13-86. Register Call Summary for Register WTGR

Watchdog Timers • General WDT Operation: [0] • Triggering a Timer Reload: [1] [2] • Modifying Timer Count/Load Values and Prescaler Setting: [3]
Watchdog Timer Register Manual • WDT Register Mapping Summary: [4] [5] • WDT Register Descriptions: [6]

Table 13-87. WWPS

Address Offset	0x034		
Physical Address	0x4831 4034 0x4903 0034	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0034		WDTIMER3
Description	This register contains the write posting bits for all write-able functional registers.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											W_PEND_WSPR	W_PEND_WTGR	W_PEND_WLDR	W_PEND_WCRR	W_PEND_WCLR

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reads return 0	R	0x0
4	W_PEND_WSPR	Write pending for register WSPR 0x0: No Start-Stop Register write pending 0x1: Start-Stop Register write pending	R	0
3	W_PEND_WTGR	Write pending for register WTGR 0x0: No Trigger Register write pending 0x1: Trigger Register write pending	R	0
2	W_PEND_WLDR	Write pending for register WLDR 0x0: No Load Register write pending 0x1: Load Register write pending	R	0
1	W_PEND_WCRR	Write pending for register WCRR 0x0: No Counter Register write pending 0x1: Counter Register write pending	R	0
0	W_PEND_WCLR	Write pending for register WCLR 0x0: No Control Register write pending 0x1: Control Register write pending	R	0

Table 13-88. Register Call Summary for Register WWPS

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [0] [1]

Table 13-89. WSPR

Address Offset	0x048		
Physical Address	0x4831 4048 0x4903 0048	**Instance**	WDTIMER2 WDTIMER3
	0x4903 0048		WDTIMER3
Description	This register holds the start-stop value that controls the internal start-stop FSM.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WSPR_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	WSPR_VALUE	The value of the start/stop register	RW	0x00000000

Table 13-90. Register Call Summary for Register WSPR

Watchdog Timers
- General WDT Operation: [0]
- Prescaler Value/Timer Reset Frequency: [1] [2]
- Start/Stop Sequence for WDTs (Using WDTi.WSPR Register): [3] [4] [5] [6] [7] [8]
- Modifying Timer Count/Load Values and Prescaler Setting: [9]

Watchdog Timer Register Manual
- WDT Register Mapping Summary: [10] [11]
- WDT Register Descriptions: [12]

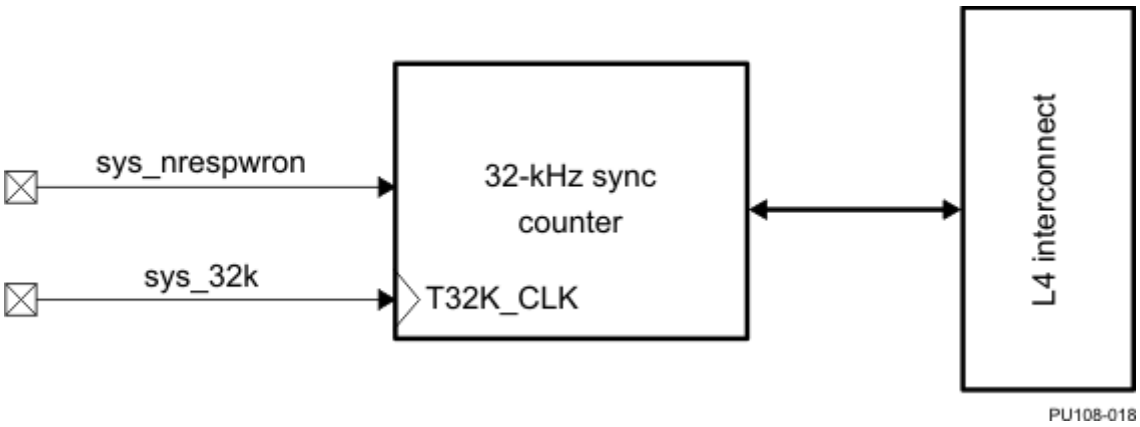
13.6 32-kHz Synchronized Timer

13.6.1 32-kHz Sync Timer Functional Description

The 32-kHz sync timer is a 32-bit counter, clocked by the falling edge of the 32-kHz system clock. It is reset while the external asynchronous power-up reset (sys_nrespwron) primary I/O is active (main device reset). When sys_nrespwron is released (on the rising edge of sys_nrespwron), after three 32-kHz clock periods, the counter starts counting up from the reset value of the counter register on the falling edge of the 32-kHz system clock. After reaching its highest value, the counter wraps back to 0 and starts counting again. Figure 13-18 shows the block diagram of the 32-kHz sync timer.

NOTE: sys_nrespwron is an active low I/O.

Figure 13-18. 32-kHz Sync Timer Block Diagram



13.6.1.1 Reading the 32-kHz Sync Timer

The sync counter register is 32 bits wide and can be read using 32-bit and 16-bit access. For correct count capture using 16-bit access, the counter register must be accessed as 16-bit LSB access first and 16-bit MSB access next. The internal logic ensures the integrity and correctness of the read data from counter register when it is split into 16-bit data. Therefore, the read data corresponds to the data when the read command is issued. Internal synchronization logic allows reading of the counter value with 32KSYNC_ICLK while the counter is running. There is a read latency of one 32KSYNC_ICLK clock cycle, which is a synchronization delay. The time latency to read the counter is a minimum of one read cycle for 32-bit access, and two read cycles for 16-bit access, where each read cycle equals to three 32KSYNC_ICLK clock cycles. .

13.6.1.2 32-kHz Sync Timer Features

The following are the main features of the 32-kHz sync timer controller:

- L4 slave interface support:
  - 32-bit data bus width
  - 32-/16-bit access supported
  - 8-bit access not supported
  - 16-bit address bus width
  - Burst mode not supported
- Only read operations are supported on the module registers; no write operation is supported (no error/no action on write).
- Free-running 32-bit upward counter
- Start and keep counting after power-on reset
- Automatic roll over to 0, highest value reached (0xFFFF FFFF)

- On-the-fly read (while counting)

13.6.2 32-kHz Sync Timer Environment

The sync timer is accessible only through the L4 interface.

13.6.3 32-kHz Sync Timer Integration

13.6.3.1 Clocking, Reset, and Power-Management Scheme

Table 13-91 lists the power and reset domain and the source clock for the 32-kHz sync timer in the device. For more information on power, reset, and clock control and domains, see , *Power, Reset, and Clock Management*.

Table 13-91. Clock, Power, and Reset Domains for 32-kHz Sync Timer

Timer	Source Clock	Intertface Clock	Clock and Power Domain	Reset Domain
32-kHz sync timer	sys_32k	32KSYNC_ICLK	WKUP	SYNCT_RST(1)

13.6.3.2 Interrupts

The 32-kHz sync timer has no interrupt outputs.

13.6.3.3 Sync Timer 32k and MSuspend Signal

By default, the Sync Timer 32k is not stopped when the MPU or DSP is in *halt*. To activate its sensitivity to the MSuspend signal, the CONTROL.CONTROL_MSUSPEND_2[29:27] SyncTMMsctrl bit has been added to the system control module.

13.7 32-kHz Sync Timer Register Manual

13.7.1 32-kHz Sync Timer Instance Summary

Table 13-92 lists the base address and block size for the 32-kHz sync timer. It is memory mapped to the L4 peripheral bus memory space.

Table 13-92. 32-kHz Sync Timer Instance Summary

Module Name	Base Address	Size
32-kHz Sync Timer	0x4832 0000	4K bytes

13.7.2 32-kHz Sync Timer Register Mapping Summary

CAUTION

The 32-kHz sync timer registers are limited to 32-bit and 16-bit data accesses; 8-bit access is not allowed and can corrupt the register content.

Table 13-93 lists the 32-kHz sync timer registers. Table 13-94 through Table 13-98 describe the register bits.

Table 13-93. 32-kHz Sync Timer Register Summary

Register Name	Type	Register Width (Bits)	Offset Address	32-kHz-Sync Timer Physical Address
REG_32KSYNCNT_REV	R	32	0x0000	0x4832 0000
REG_32KSYNCNT_SYSCONFIG	R/W	32	0x0004	0x4832 0004
REG_32KSYNCNT_CR	R	32	0x0010	0x4832 0010

13.7.3 32-kHz Sync Timer Register Descriptions

Table 13-94. REG_32KSYNCNT_REV

Address Offset	0x0000
Physical Address	0x4832 0000
Description	This register contains the sync counter IP revision code.
Type	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CID_REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0.	R	0x000000
7:0	CID_REV	Counter revision number [7:4] = Major revision [3:0] = Minor revision (Examples: 0x10 for 1.0, 0x21 for 2.1)	R	See (1)

(1) TI internal data

Table 13-95. Register Call Summary for Register REG_32KSYNCNT_REV

32-kHz Sync Timer Register Manual

- 32-kHz Sync Timer Register Mapping Summary: [0]

Table 13-96. REG_32KSYNCNT_SYSCONFIG

Address Offset	0x0004
Physical Address	0x4832 0004
Description	This register is used for IDLE modes only.
Type	RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											IDLEMODE			Reserved		

Bits	Field Name	Description	Type	Reset
31:5	Reserved	Reads return 0.	R	0x0
4:3	IDLEMODE	Power management REQ/ACK control	RW	0x0
		0x0: Force idle. An idle request is acknowledged unconditionally.		
		0x1: No-idle. An idle request is never acknowledged.		
		0x2: Reserved		
		0x3: Reserved		
2:0	Reserved	Reads return 0.	R	0x0

Table 13-97. Register Call Summary for Register REG_32KSYNCNT_SYSCONFIG

32-kHz Sync Timer Register Manual

- 32-kHz Sync Timer Register Mapping Summary: [0]

Table 13-98. REG_32KSYNCNT_CR

Address Offset	0x0010
Physical Address	0x4832 0010
Description	This register contains the 32-kHz sync counter value.
Type	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTER_VALUE																															

Bits	Field Name	Description	Type	Reset
31:0	COUNTER_VALUE	Counter register value	R	0x00000003

Table 13-99. Register Call Summary for Register REG_32KSYNCNT_CR

32-kHz Sync Timer Register Manual

- 32-kHz Sync Timer Register Mapping Summary: [0]

UART/IrDA/CIR Module

This chapter describes the function, operation, and configuration of the universal asynchronous receiver/transmitter (UART)/infrared data association (IrDA)/consumer infrared (CIR) module.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

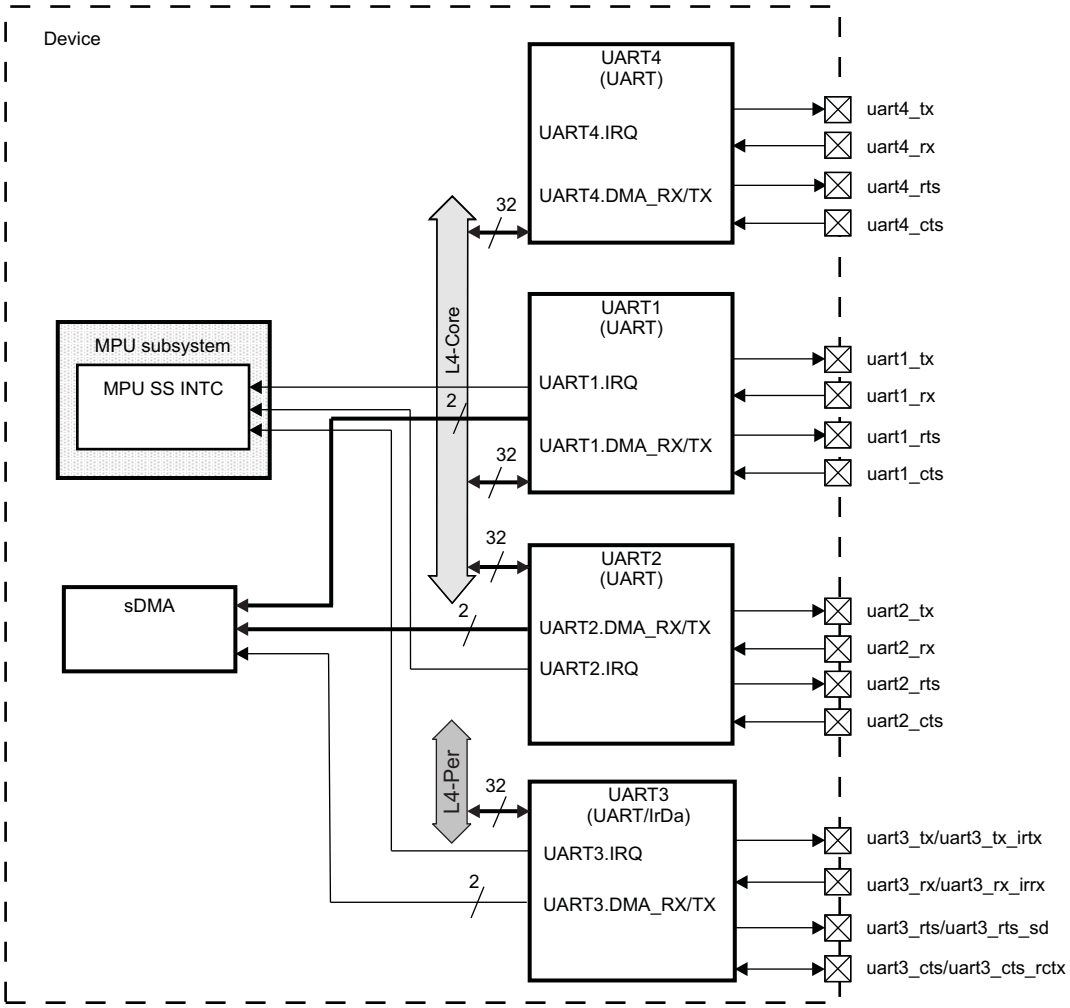
Topic **Page**

14.1 UART/IrDA/CIR Overview

The processor contains three universal asynchronous receiver/transmitter (UART) devices controlled by the microprocessor unit (MPU) (see Figure 14-1):

- Three UART-only modules, UART1, UART2 and UART4 are pinned out for use as UART devices only. UART1, UART2 and UART3 must be programmed by setting the UARTi.MDR1_REG[2:0] MODE_SELECT field to one of the three UART operating modes.
- UART3, which adds infrared communication support, is pinned out for use as a UART, infrared data association (IrDA), or consumer infrared (CIR) device, and can be programmed to any available operating mode.

Figure 14-1. UART Module



14.1.1 UART Features

The UARTs (UART1, UART2, UART3, and UART4 when in UART mode) include the following key features:

- 16C750 compatibility
- 64-byte FIFO for receiver and 64-byte FIFO for transmitter
- Programmable interrupt trigger levels for FIFOs
- Baud generation based on programmable divisors N (N = 1…16,384) operating from a fixed functional clock of 48 MHz

Oversampling is programmed by software as 16 or 13; thus, the baud rate computation is either:

- Baud rate = (functional clock/16)/N
- Baud rate = (functional clock/13)/N

This software programming mode enables higher baud rates with the same error amount without changing the clock source:

- Break character detection and generation
- Configurable data format
  - Data bit: 5, 6, 7, or 8 bits
  - Parity bit: Even, odd, none
  - Stop-bit: 1, 1.5, 2 bit(s)
- Flow control: Hardware (RTS/CTS) or software (XON/XOFF)

The UART clocks are connected to produce a baud rate of up to 3.6M bits/s. Table 14-1 lists the supported baud rates, the requested divisor, and the corresponding error versus the standard baud rate.

Table 14-1. UART Mode Baud Rates, Divisor Values, and Error Rates

Baud Rate	Oversampling	Divisor	Error (%)
300	16	10000	0
600	16	5000	0
1200	16	2500	0
2400	16	1250	0
4800	16	625	0
9600	16	312	0.16
14,400	16	208	0.16
19,200	16	156	0.16
28,800	16	704	0.16
38,400	16	78	0.16
57,600	16	52	0.16
115,200	16	26	0.16
230,400	16	13	0.16
460,800	13	8	0.16
921,600	13	4	0.16
1,843,200	13	2	0.16
3,000,000	16	1	0
3,686,400	13	1	0.16

NOTE: The maximum baud rate required of the module does not depend on the change of OPP.

14.1.2 IrDA Features

The IrDA (UART3 only) includes the following key features:

- Support of IrDA 1.4 slow infrared (SIR), medium infrared (MIR), and fast infrared (FIR) communications
  - Frame formatting: Addition of variable beginning-of-frame (xBOF) characters and end-of-frame (EOF) characters
  - Uplink/downlink cyclic redundancy check (CRC) generation/detection
  - Asynchronous transparency (automatic insertion of break character)
  - Eight-entry status FIFO (with selectable trigger levels) to monitor frame length and frame errors
  - Framing error, CRC error, illegal symbol (FIR), and abort pattern (SIR, MIR) detection

Table 14-2 lists the supported baud rates, the requested divisor, and the corresponding error versus the standard baud rate.

Table 14-2. UART IrDA Mode Baud Rates, Divisor Values, and Error Rates

Baud Rate	IR Mode	Encoding	Divisor	Error (%)
2400	SIR	3/16	1250	0
9600	SIR	3/16	312	0.16
19,200	SIR	3/16	156	0.16
38,400	SIR	3/16	78	0.16
57,600	SIR	3/16	52	0.16
115,200	SIR	3/16	26	0.16
576,000	MIR	1/4	2	0
1,152,000	MIR	1/4	1	0
4,000,000	FIR	4 PPM(1)	1	0

(1) PPM = pulse-position-modulation

14.1.3 CIR Features

The CIR mode uses a variable pulse-width modulation (PWM) technique (based on multiples of a programmable *t* period) to encompass the various formats of infrared encoding for remote-control applications. The CIR logic transmits data packets based on a user-definable frame structure and packet content.

The CIR (UART3 only) includes the following key features to provide CIR support for remote control applications:

- Transmit mode only (receive mode is not supported)
- Free data format (supports any remote-control private standards)
- Selectable bit rate
- Configurable carrier frequency
- 1/2, 5/12, 1/3, or 1/4 carrier duty cycle

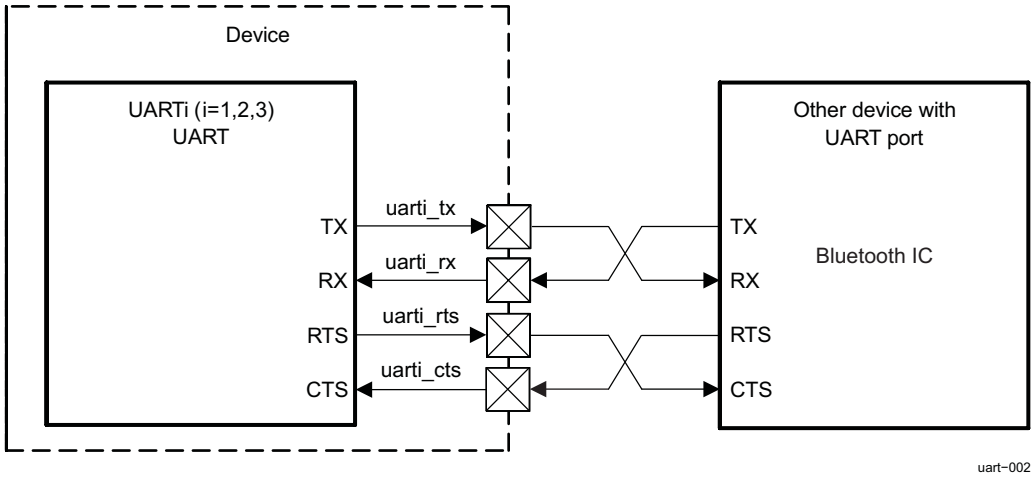
14.2 UART/IrDA/CIR Environment

This section describes the UART/IrDA/CIR connection with an external device.

14.2.1 System Using UART Communication with Hardware Handshake

UART1, UART2, or UART3 can be connected easily to the UART port of an external IC (see Figure 14-2).

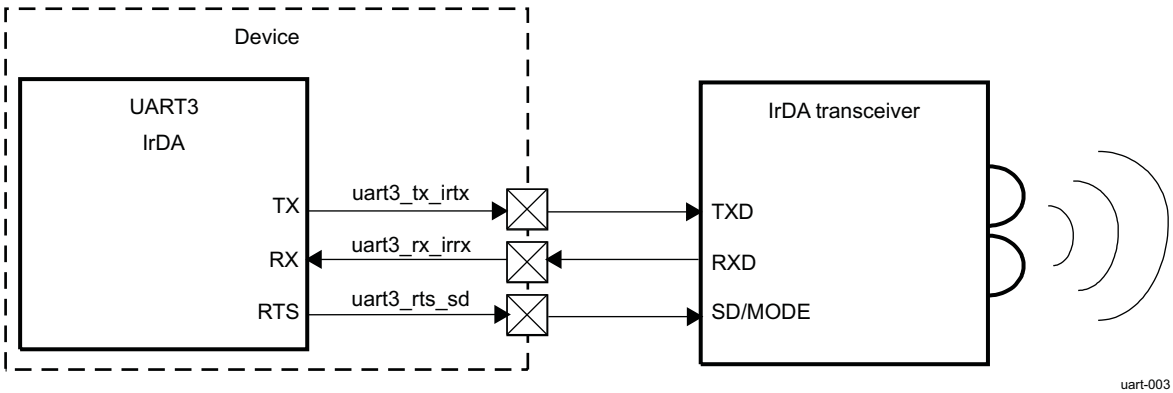
Figure 14-2. UART Mode Bus System Overview



14.2.2 System using IrDA Communication Protocol

As Figure 14-3 shows, UART3 can be connected to an external infrared transceiver in the IrDA modes (FIR, SIR, and MIR).

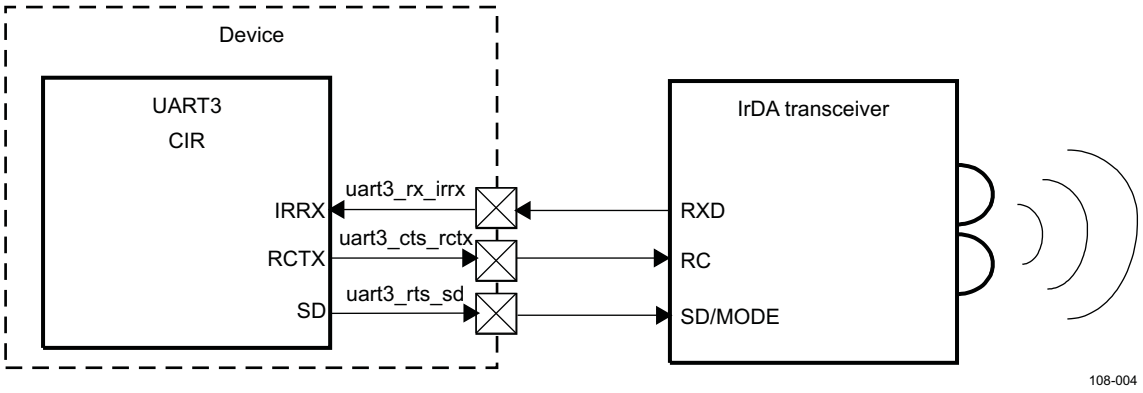
Figure 14-3. IrDA System Overview



14.2.3 System using CIR Communication Protocol with Remote Control

UART3 can be connected to an external Infrared transceiver in CIR mode (see Figure 14-4).

Figure 14-4. CIR System Overview



14.2.4 UART Interface Description

14.2.4.1 UART Interface Description

Table 14-3 describes the UART interface.

Table 14-3. UART I/O Pin Description

Signal	I/O[1]	Description	Reset
UART Modem Signals			
uarti_rx	I	Serial data input	Unknown
uarti_tx	O	Serial data output Because this pin is active high in the IrDA mode and the output is muxed, this pin is set to low on reset (when UARTi.MDR1_REG[2:0] is set to 0x7) and takes the defined inactive level of that signal corresponding to when and how UARTi.MDR1_REG is programmed; that is, the output is 1 (inactive for UART modem modes) and 0 (inactive for IrDA modes).	0
uarti_cts	I	Clear to send Active-low modem status signal. Reading the UARTi.MSR_REG[4] NCTS_STS bit checks the condition of uarti_cts. Reading the UARTi.MSR_REG[0] CTS_STS bit checks a change of state of uarti_cts since the last read of the modem status register. The auto-nCTS mode uses uarti_cts to control the transmitter.	0
uarti_rts	O	Request to send When active (low), the module is ready to receive data. Setting the UARTi.MCR_REG[1] RTS bit activates uarti_rts, which becomes inactive as the result of a module reset, loopback mode, or clearing the UARTi.MCR_REG[1] RTS bit. In auto-RTS mode, uarti_rts becomes inactive as a result of the receiver threshold logic.	0

(1) I = Input, O = Output

14.2.4.2 UART Protocol and Data Format

The UART device operates in three modes:

- UART 16x mode (≤230.4K bits/s)
- UART 16x mode with autobauding (≥1200 bits/s and ≤115.2K bits/s)
- UART 13x mode (≥460.8K bits/s)

CAUTION

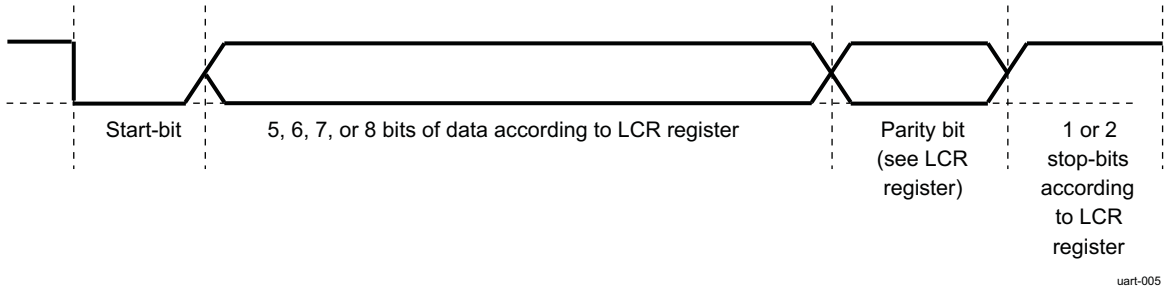
To be used as a UART, the operating mode must be programmed appropriately in the UARTi.MDR1_REG[2:0] MODE_SELECT field to select the UART, IrDA, or CIR mode.

The UART uses a wired interface for serial communication with a remote device.

The UART module is functionally compatible with the TL16C750 UART and earlier designs such as the TL16C550.

Figure 14-5 shows the UART frame data format.

Figure 14-5. UART Frame Data Format



14.2.5 IrDA Functional Interfaces

14.2.5.1 UART3 Interface Description

Table 14-4 describes the UART3 interface.

Table 14-4. UART3 I/O Description

Signal	I/O[1]	Description	Reset
IrDA Signals			
uart3_rx_irrx	I	Serial data input	Unknown
uart3_tx_irtx	O	Serial data output in IrDA modes (SIR, MIR, and FIR). In other modes, this pin is set to the reset value (inactive state).	0
uart3_rts_sd	O	SD mode is used to configure the transceivers. The SD pinout is an inverted value of the UART3.ACREG_REG[6] SD_MOD bit.	1

(1) I = Input, O = Output

14.2.5.2 IrDA Protocol and Data Format

14.2.5.2.1 SIR Mode

In SIR mode, data is transferred between the MPU and peripheral devices at speeds of up to 115,200 baud. A SIR transmit frame begins with start flags (a single 0xC0, a multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), is followed by frame data and a CRC-16, and ends with a stop flag (0xC1).

The bit format for a single word uses 1 start bit, 8 data bits, and 1 stop bit, and is unaffected by the use and settings of the UART3.LCR_REG register.

The UART3.BLR_REG[6] XBOF_TYPE bit selects whether the 0xC0 or 0xFF start patterns are used when multiple start flags are required.

The SIR transmit state-machine attaches start flags, CRC-16, and stop flags, and checks the outgoing data to establish whether data transparency is required.

SIR transparency is carried out if the outgoing data between the start and stop flags contains 0xC0, 0xC1, or 0x7D. If one of these start flags is about to be transmitted, the SIR state-machine first sends an escape character (0x7D), then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 0x7D character.

The SIR receive state-machine recovers the receive clock, removes the start flags and any transparency from the incoming data, and determines the frame boundary with reception of the stop flag. The SIR state-machine also checks for errors, such as a frame abort (0x7D character followed immediately by a 0xC1 stop flag without transparency), a CRC error, and a frame-length error. At the end of a frame reception, the MPU reads the line status register (UART3.LSR_REG) to find possible errors of the received frame.

NOTE: Data can be transferred both ways by the module, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See the UART3.ACREG_REG[5] DIS_IR_RX bit description. This applies to all three modes: SIR, MIR, and FIR.

Infrared output in SIR mode can be either 1.6-us or 3/16th encoding, selected by the UART3.ACREG_REG[7] PULSE_TYPE bit. In 1.6-us encoding, the infrared pulse width is 1.6 µs; and in 3/16th encoding, the infrared pulse width is 3/16th of a bit duration (1/baud rate).

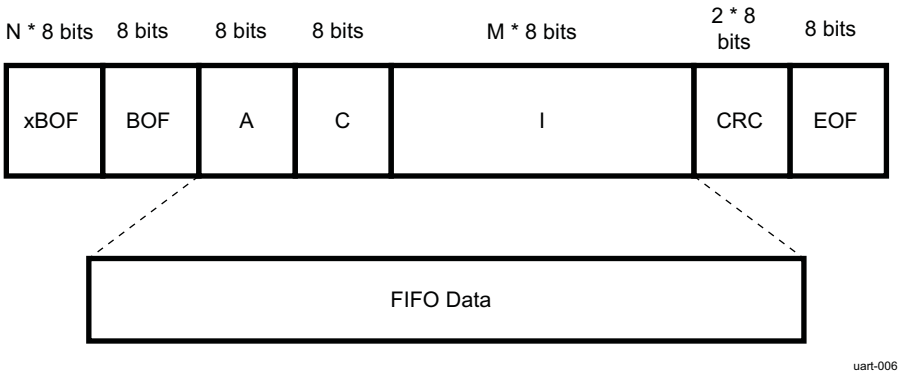
The transmitting device must send at least two start flags at the start of each frame for back-to-back frames.

NOTE: Reception supports variable-length stop-bits.

14.2.5.2.1.1 Frame Format

Figure 14-6 shows the IrDA SIR frame format.

Figure 14-6. IrDA SIR Frame Format



The CRC is applied on the address (A), control ©), and information (I) bytes.

NOTE: The two words of CRC are written to the FIFO in reception.

14.2.5.2.1.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape), the controller performs certain tasks:

- In transmission:
  - Inserts a control escape (CE) byte preceding the byte
  - Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20)

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

- In reception:

  For the A, C, I, CRC field:

  – Compares the byte with the CE byte; if they are not equal, sends the byte to the CRC detector and stores it in the RX FIFO
  – If the byte is equal to the CE byte, discards the CE byte
  – Complements bit 5 of the byte following the CE
  – Sends the complemented byte to the CRC detector and stores it in the RX FIFO

14.2.5.2.1.3 Abort Sequence

The transmitter can decide to prematurely close a frame. The transmitter aborts by sending the following sequence: 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

The receiver treats a frame as an aborted frame when a 0x7D character followed immediately by a 0xC1 character is received without transparency.

14.2.5.2.1.4 Pulse Shaping

The SIR mode supports both the 3/16th and the 1.6-µs pulse duration methods. The UART3.ACREG_REG[7] PULSE_TYPE bit selects the pulse width method in transmit mode.

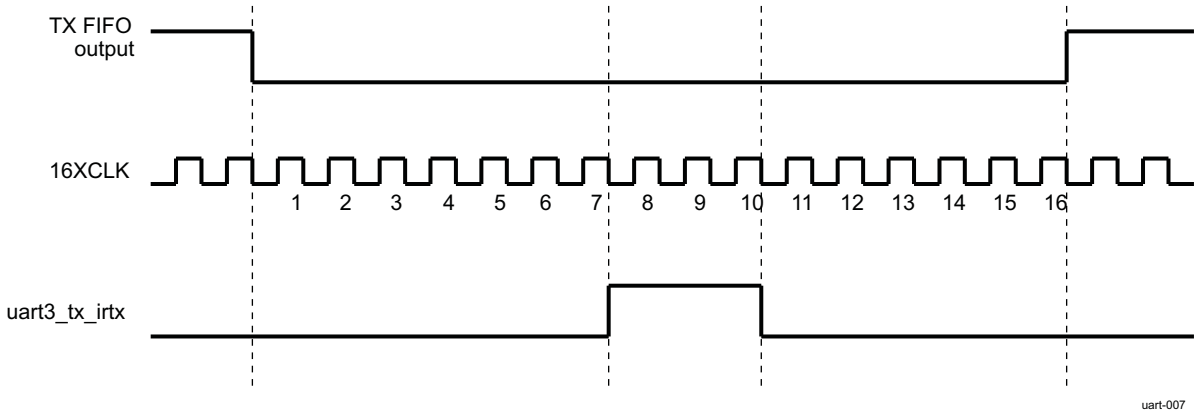
14.2.5.2.1.5 Encoder

Serial data from the transmit state-machine is encoded to transmit data to the optoelectronics. While the TX FIFO output is high, the uart3_tx_irtx line is always low, and the counter used to form a pulse on uart3_tx_irtx is cleared continuously.

After the TX FIFO output resets to 0, uart3_tx_irtx rises on the falling edge of the 7th 16XCLK. On the falling edge of the 10th 16XCLK pulse, uart3_tx_irtx falls, creating a three-clock-wide pulse. While the TX FIFO output stays low, a pulse is transmitted during the 7th to the 10th clock of each 16-clock bit cycle.

Figure 14-7 shows the IrDA SIR encoding mechanism.

Figure 14-7. IrDA SIR Encoding Mechanism

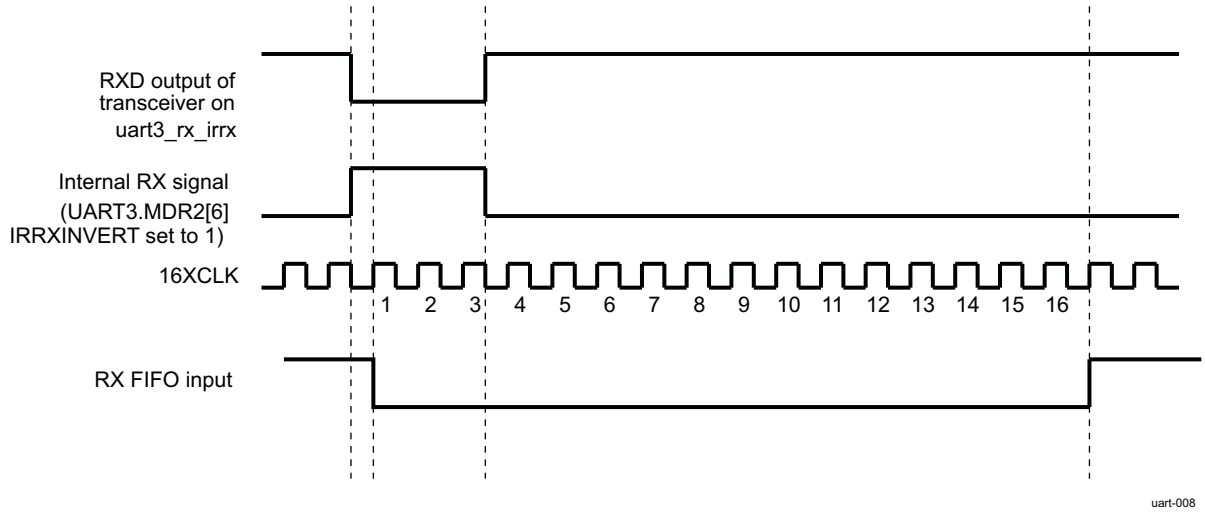


14.2.5.2.1.6 Decoder

After reset, the RX FIFO input is high and the 4-bit counter is cleared. When a rising edge is detected on RX, the RX FIFO input falls on the next rising edge of 16XCLK with sufficient setup time. The RX FIFO input stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RX, the RX FIFO input remains high.

Figure 14-8 shows the IrDA SIR decoding mechanism.

Figure 14-8. IrDA SIR Decoding Mechanism



Data can be transferred both ways by the module, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. The operation of the uart3_rx_irrx input can be disabled using the UART3.ACREG_REG[5] DIS_IR_RX bit. Furthermore, the UART3.MDR2_REG[6] IRRXINVERT bit can invert the signal from the transceiver (RXD) pin to the IRRX logic inside the UART. This inversion is performed by default.

14.2.5.2.1.7 IR Address Checking

In all IR modes, when address checking is enabled by setting EFR_REG[1:0] (see Table 14-5), only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multipoint infrared environment. To program two frame addresses that the UART3 receives in IrDA mode, use the UART3.XON1_ADDR1_REG[7:0] field and the UART3.XON2_ADDR2_REG[7:0] field.

Table 14-5. EFR_REG[0-1] IR Address Checking Options

EFR_REG[1]	EFR_REG[0]	IR Address Checking
0	0	All address-checking operations disabled
0	1	Only address 1 checking enabled
1	0	Only address 2 checking enabled
1	1	All address-checking operations enabled

14.2.5.2.2 SIR Free Format Mode

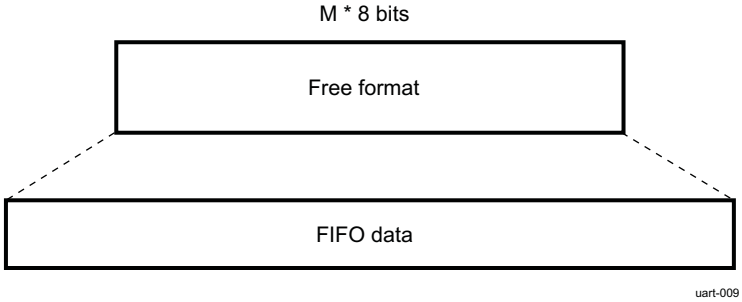
To allow complete software flexibility when transmitting and receiving infrared data packets, the SIR free format (FF) mode is a subfunction of the existing SIR mode; all frames going to and from the FIFO buffers are untouched with respect to appending and removing control characters and CRC values.

This mode corresponds to a UART mode with a pulse modulation of 3/16 of baud rate pulse width.

For example, a normal SIR packet has BOF control and CRC error-checking data appended (transmitting) or removed (receiving) from the data going to and from the FIFOs.

Figure 14-9 shows the SIR free format mode.

Figure 14-9. SIR Free Format Mode

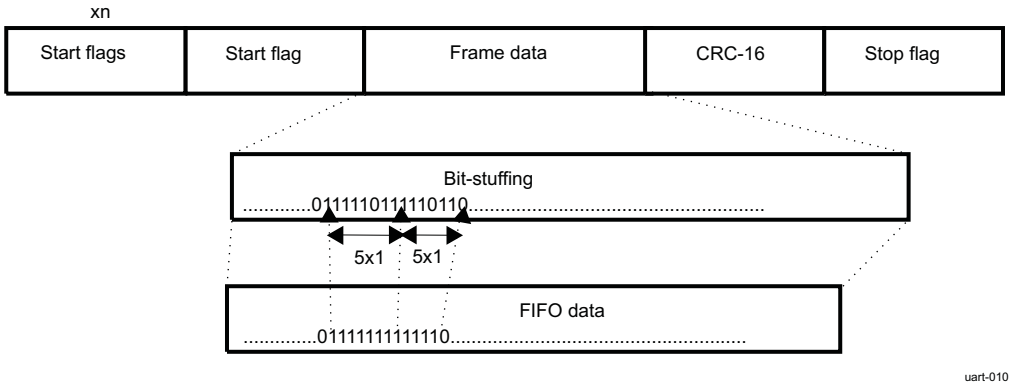


In the SIR free format mode, the MPU software must construct (that is, encode and decode) the entire FIFO data packet.

14.2.5.2.3 MIR Mode

In MIR mode, data is transferred between the MPU and the peripheral devices at 0.576 or 1.152M bits/s speed. A MIR transmit frame starts with start flags (at least two), followed by a frame data, CRC-16, and ends with a stop flag (see Figure 14-10).

Figure 14-10. MIR Transmit Frame Format



On transmit, the MIR state-machine attaches start flags, CRC-16, and stop flags, as in SIR mode. All fields are transmitted least-significant bit (LSB) of each byte first.

Contrary to SIR mode:

- The state-machine looks for consecutive 1s in the frame data and automatically inserts 0 after five consecutive 1s (this is called bit-stuffing).
- 0x7E is used for both start and stop flags (unambiguously, not data, because of bit-stuffing).
- Abort sequence requires a minimum of seven consecutive 1s (unambiguously, not data, because of bit stuffing).
- Back-to-back frames are allowed with three or more stop flags in between. If two consecutive frames are not back to back, the gap between the last stop flag of the first frame and the start flag of the second frame must be separated by at least seven bit durations.

On receive, the MIR receive state-machine recovers the receive clock, removes the start flags, destuffs the incoming data, and determines the frame boundary with reception of the stop flag. The state-machine also checks for errors, such as frame abort, CRC error, and frame-length error. At the end of a frame reception, the MPU reads the line status register (UART3.LSR_REG) to detect possible errors of the received frame.

Data can be transferred both ways by the module, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

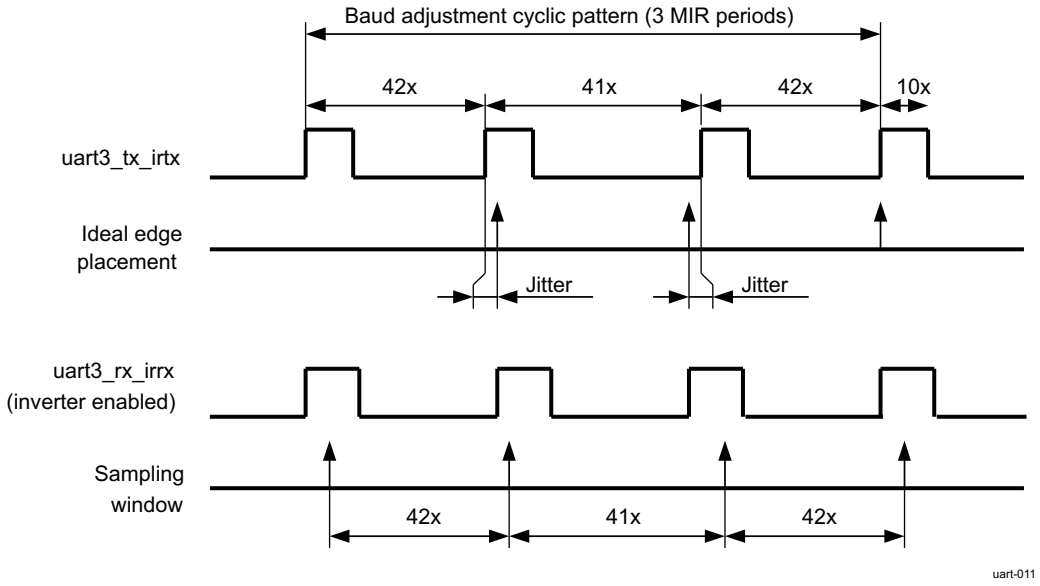
14.2.5.2.3.1 MIR Encoder/Decoder

To meet the MIR baud rate tolerance of +/–0.1 percent with a 48-MHz clock input, a 42-41-42 encoding/decoding adjustment is performed. The reference start point is the first start flag, and the 42-41-42 cyclic pattern is repeated until the stop flag is sent or detected.

The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4, but it is within the tolerances defined by the IrDA specifications.

Figure 14-11 shows the MIR baud rate adjustment mechanism.

Figure 14-11. MIR Baud Rate Adjustment Mechanism

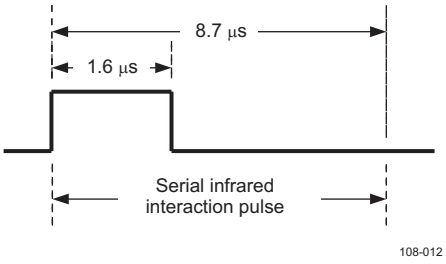


14.2.5.2.3.2 SIP Generation

In the MIR and FIR operation modes, the transmitter must send a serial infrared interaction pulse (SIP) at least once every 500 ms. The SIP informs slow devices (operating in SIR mode) that the medium is currently occupied.

Figure 14-12 shows the SIP.

Figure 14-12. SIP



14.2.5.2.4 FIR Mode

In FIR mode, data is transferred between the MPU and the peripheral devices at 4M bits/s. A FIR transmit frame starts with a preamble that is followed by a start flag, frame data, CRC-32, and ends with a stop flag.

Table 14-6 shows the FIR transmit frame format.

Table 14-6. FIR Transmit Frame Format

Preamble (16x)	Start flag	Frame data	CRC-32	Stop flag

On transmit, the FIR transmit state-machine attaches the preamble, start flag, CRC-32, and stop flag. An abort sequence requires at least two transmissions of 0000. Back-to-back frames are allowed, but each frame must be complete.

The state-machine also encodes the transmit data into 4-PPM format (see Table 14-7) and generates the SIP (see Section 14.2.5.2.3.2, *SIP Generation*).

Table 14-7. 4-PPM Format

Data Bit Pair (Bin)	4-PPM Data Symbol (Bin)
00	1000
01	0100
10	0010
11	0001

The four symbols described in Table 14-7 are the legal encoded data symbols. All other combinations are illegal for encoding data. Some of these illegal symbols are used in the definition of the preamble, start flag, and stop flag because they are unambiguously not data (see Table 14-8).

Table 14-8. FIR Preamble, Start Flag, and Stop Flag

Frame Part	Transmitted Frame (Bin)
Preamble	1000 0000 1010 1000 (16 repeated transmissions)
Start Flag	0000 1100 0000 1100 0110 0000 0110 0000
Stop Flag	0000 1100 0000 1100 0000 0110 0000 0110

All fields are transmitted the LSBs of each byte first (see Table 14-9).

Table 14-9. FIR Data Byte Transmission Order Example

Data Byte (Hex)	Data Byte Pair (Bin)	4-PPM Data Symbol (Bin)	Transmission Order
0x0B	00	1000	4
	00	1000	3
	10	0010	2
	11	0001	1

On receive, the FIR receive state-machine recovers the receive clock, removes the preamble and the start flag, decodes the 4-PPM incoming data, and determines the frame boundary with reception of the stop flag. The state-machine also checks for errors, such as illegal symbol, CRC error, and frame-length error. At the end of a frame reception, the MPU reads the line status register (UART3.LSR_REG) to detect possible errors of the received frame.

Data can be transferred both ways by the module, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

14.2.6 CIR Functional Interfaces

14.2.6.1 CIR Interface Description

Table 14-10 describes the CIR interface.

Table 14-10. CIR I/O Description

Signal	I/O[1]	Description	Reset
CIR Signals			
uart3_rx_irrx	I	Serial data input	Unknown
uart3_cts_rctx	O	Serial data output in CIR mode. In other modes, this pin is set to the reset value (inactive state).	0
uart3_rts_sd	O	SD mode is used to configure the transceivers. The SD pinout is an inverted value of the UART3.ACREG_REG[6] SD_MOD bit.	1

(1) I = Input, O = Output

14.2.6.2 CIR Protocol and Data Format

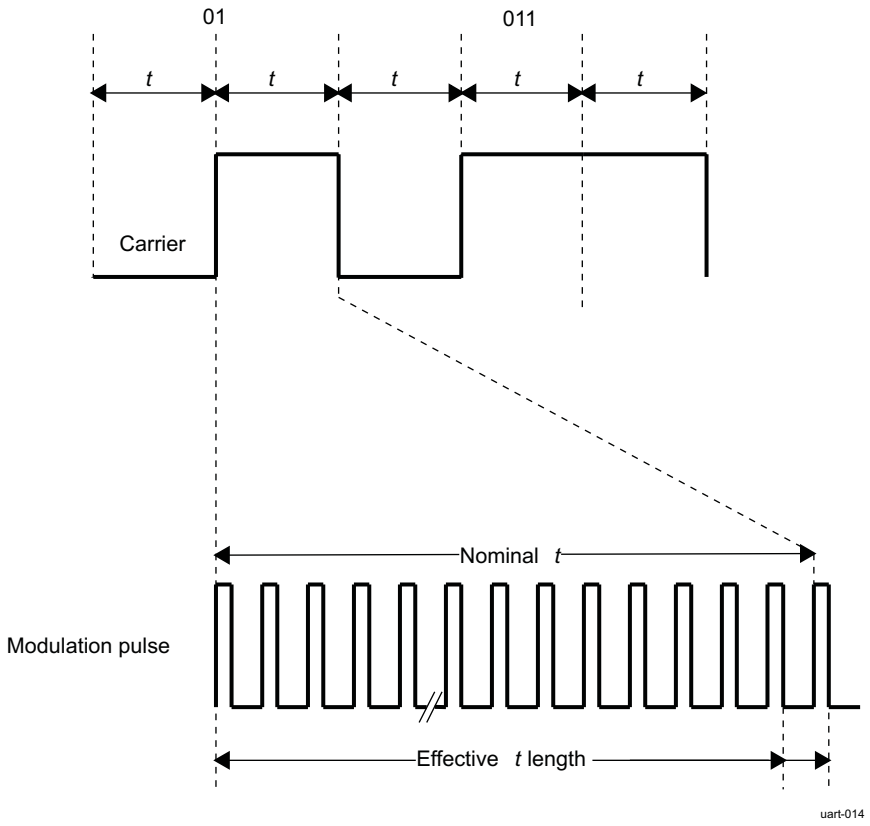
In the CIR mode, the infrared operation functions as a programmable (universal) remote control.

The CIR mode uses a variable PWM technique (based on multiples of a programmable *t* period) to encompass the various formats of infrared encoding for remote control applications. The CIR logic transmits data packets based on the user-defined frame structure and packet content.

14.2.6.2.1 Carrier Modulation

Each modulated pulse that constitutes a digit is a train of on/off pulses (see Figure 14-13).

Figure 14-13. CIR Pulse Modulation

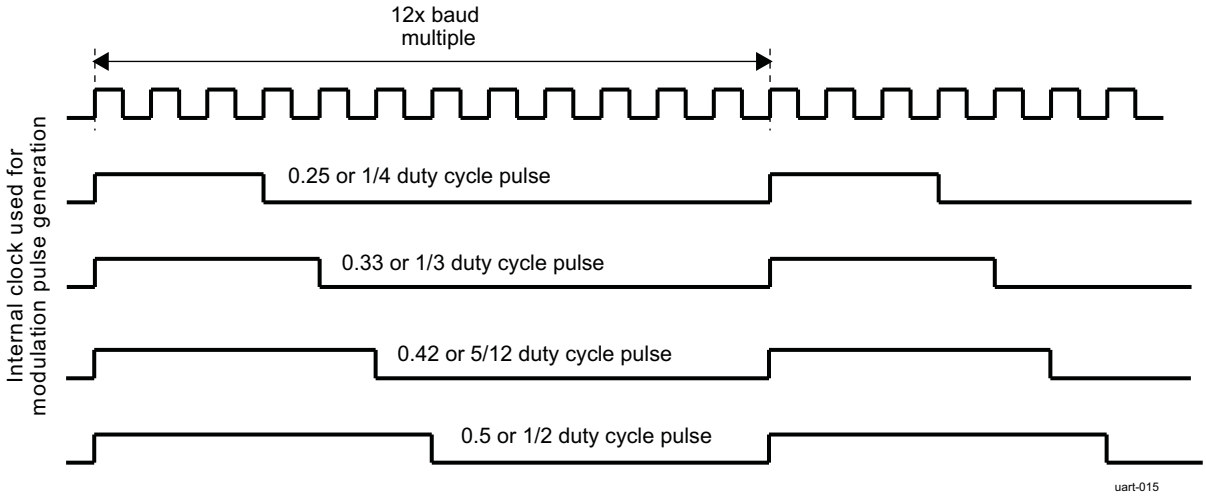


14.2.6.2.2 Pulse Duty Cycle

The programmer can choose between four possible duty cycles for modulation pulses by setting the appropriate value in the UART3.MDR2_REG[5:4] CIR_PULSE_MODE field (1/4, 1/3, 5/12, or ½).

Figure 14-14 shows the CIR modulation duty cycles.

Figure 14-14. CIR Modulation Duty Cycle



The transmission logic ensures that all pulses are transmitted completely (that is, no cutoff during transmission). Furthermore, while transmitting continuous bytes back-to-back, no delay is inserted between 2 transmitted bytes. Thus, software must handle the delay between consecutively transmitted bytes if the receiving end needs it.

14.2.6.2.3 Consumer IR Encoding/Decoding

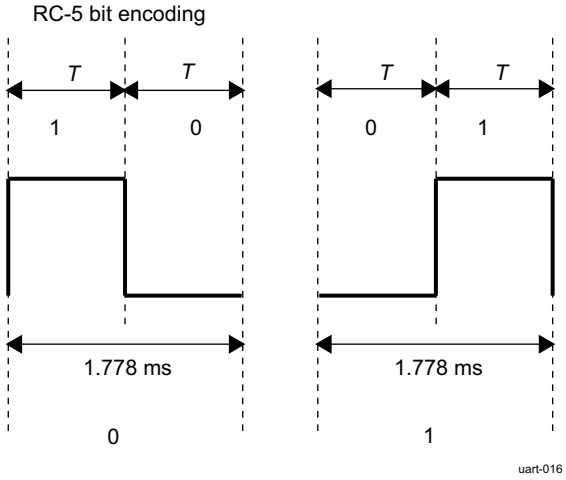
There are two methods of encoding for remote control applications. The first method uses time-extended bit forms (a variable pulse distance, or duration, in which the difference between a logic 1 and logic 0 is the length of the pulse width).

The second encoding method uses a biphase in which the encoding of the logic 0 and logic 1 is in the change of signal level from 1 to 0 or 0 to 1, respectively. Japanese manufacturers favor pulse duration encoding; European manufacturers favor biphase encoding.

The CIR mode uses a completely flexible free-format encoding in which the TX FIFO is transmitted as a modulated pulse with duration T.

Similarly, 0 is transmitted as a blank duration T. The MPU constructs and deciphers the protocol of the data. For example, the RC-5 protocol using Manchester encoding can be emulated as using a 01 pair for 1 and a 10 pair for 0 (see Figure 14-15.).

Figure 14-15. RC-5 Bit Encoding

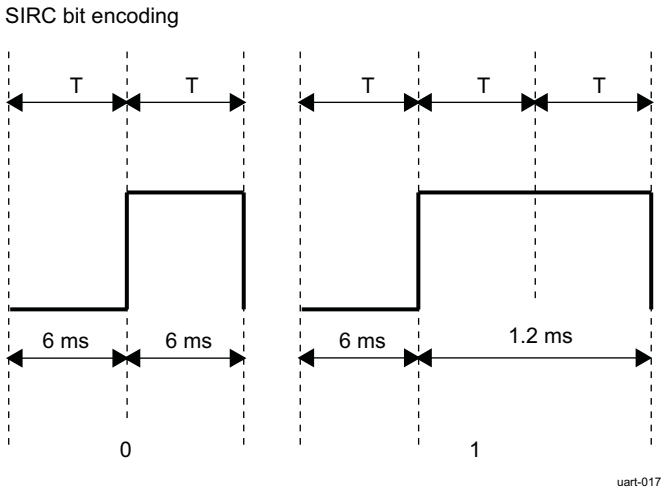


Because the CIR mode logic does not impose a fixed format for infrared packets of data, the MPU software can define the format using simple data structures that are then modulated into an industry standard, such as RC-5 or SIRC. To send a sequence of 0101 in RC-5, the MPU software must write an 8-bit binary character of 10011001 to the data FIFO of the UART.

For SIRC, the modulation length (that is, multiples of T) is the method used to distinguish between 1 and 0. The following SIRC digits show the difference in encoding between this and, for example, RC-5. The pulse width is extended for one digit.

Figure 14-16 shows SIRC bit encoding.

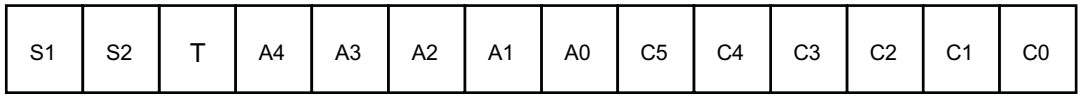
Figure 14-16. SIRC Bit Encoding



To construct comprehensive packets constituting remote-control commands, the MPU software must combine a number of 8-bit data characters in a sequence that follows one of the universally accepted formats.

Figure 14-17 shows a standard RC-5 frame as detected by the UART3 in CIR mode (the SIRC format follows this). Each field in RC-5 can be considered as two T pulses (digital bits) from the TX and RX FIFOs.

Figure 14-17. RC-5 Standard Packet Format



Where:

S1, S2:	Start bits (always 1)
T:	Toggle bit
A4..A0:	Address (or system) bits
C5..C0:	Command bits

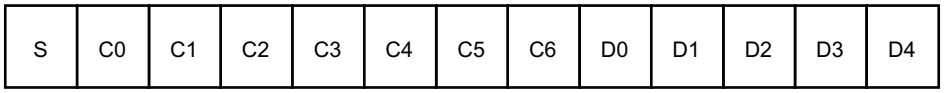
The toggle bit T changes when a new command is transmitted to detect when the same key is pressed twice (effectively receiving the same data from the host consecutively). A brief delay in the transmission of the same command is detected by the use of the toggle bit because a code is sent while the MPU transmits characters to the UART for transmission. The address bits define the machine or device for which the infrared transmission is intended, and the command defines the operation.

To accommodate an extended RC-5 format, the S2 bit is replaced by an additional command bit (C6) that allows the command range to increase to 7 bits. This format is known as the extended RC-5 format.

SIRC encoding uses the duration of modulation for mark and space; therefore, the duration of data bits inside the standard frame length varies.

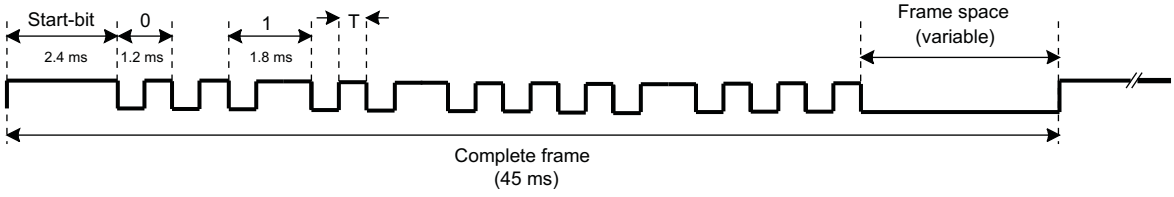
Figure 14-18 shows the packet format and bit encoding. As Figure 14-19 shows, 1 start bit of 2 ms and control codes are followed by data that constitute the entire frame.

Figure 14-18. SIRC Packet Format



NOTE: The encoding must take a standard duration, but the contents of the data can vary. This implies that the control software for emitting and receiving data packets must exercise a scheme of interpacket delay, where the emission of successive packets can be done only after a real-time delay has expired.

Figure 14-19. SIRC Bit Transmission Example

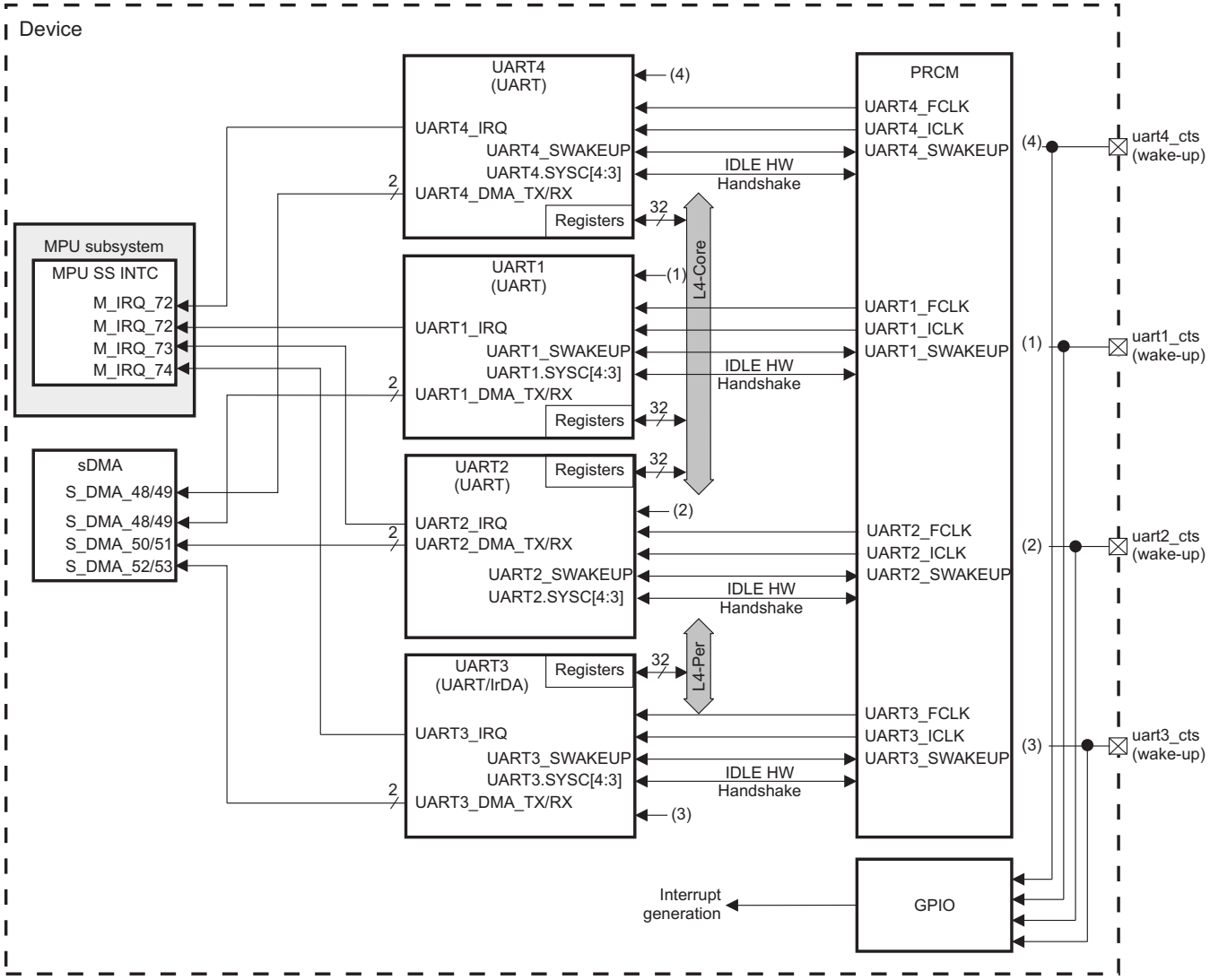


This document does not describe all encoding methods and techniques; the preceding information shows the considerations required to employ different encoding methods for different industry-standard protocols. See industry-standard documentation for specific methods of encoding and protocol usage.

14.3 UART/IrDA/CIR Integration

Figure 14-20 shows the internal connections with related modules for UART functions.

Figure 14-20. UART Functional Integration



14.3.1 Clocking, Reset, and Power-Management Scheme

14.3.1.1 Clocking

Each UART uses a 48-MHz functional clock for its logic and the generation of external interface signals. Each UART uses an interface clock for register accesses. The power, reset, and clock management (PRCM) module generates and controls all these clocks (for more information, see the *Power, Reset, and Clock Management* chapter).

Table 14-11 describes the UART clocks.

Table 14-11. UART Clocks

Clock Type	Frequency	Name	Comments
UART1			
Functional clock	CORE_48M_FCLK	UART1_FCLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM.CM_FCLKEN1_CORE[13] EN_UART1 bit.
Interface clock	CORE_L4_ICLK	UART1_ICLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM.CM_ICLKEN1_CORE[13] EN_UART1 bit. Enable/disable auto-idle for this clock with the PRCM.CM_AUTOIDLE1_CORE[13] AUTO_UART1 bit.
UART2			
Functional clock	CORE_48M_FCLK	UART2_FCLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM.CM_FCLKEN1_CORE[14] EN_UART2 bit.
Interface clock	CORE_L4_ICLK	UART2_ICLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM. CM_ICLKEN1_CORE[14] EN_UART2 bit. Enable/disable auto-idle for this clock with the PRCM.CM_AUTOIDLE1_ CORE[14] AUTO_UART2 bit.
UART3			
Functional clock	PER_48M_FCLK	UART3_FCLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM.CM_FCLKEN_PER[11] EN_UART3 bit.
Interface clock	PER_L4_ICLK	UART3_ICLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM. CM_ICLKEN_PER[11] EN_UART3 bit. Enable/disable auto-idle for this clock with the PRCM.CM_AUTOIDLE_ PER[11] AUTO_UART3 bit.
UART4			
Functional clock	CORE_48M_FCLK	UART4_FCLK	Source and gating is the PRCM module. Enable/disable this clock with the PRMC.CM_FCLKEN1_CORE[23] EN_UART4 bit.
Interface clock	CORE_L4_ICLK	UART4_ICLK	Source and gating is the PRCM module. Enable/disable this clock with the PRCM.CM_ICLKEN1_CORE[23] EN_UART1 bit. Enable/disable auto-idle for this clock with the PRCM.CM_AUTOIDLE1_CORE[23] AUTO_UART1 bit.

The idle and wake-up processes use a handshake protocol between the PRCM module and the UARTs (for a description of the protocol, see the *Power, Reset, and Clock Management* chapter). The UARTi.SYSC_REG[4:3] IDLEMODE field controls the UART idle mode.

14.3.1.2 Hardware Reset

Table 14-12 lists the UART reset domain.

Table 14-12. Reset Domain

Peripherals	Reset Domain
UART1	CORE_RST
UART2	CORE_RST
UART3	PER_RST
UART4	CORE_RST

14.3.1.3 Software Reset

The UARTi.SYSC_REG[1] SOFTRESET bit controls the software reset; writing 1 to this bit triggers a software-reset functionally equivalent to hardware reset.

14.3.2 Hardware Requests

14.3.2.1 Interrupts

Table 14-13 describes the UART interrupt mappings.

Table 14-13. Interrupt Mapping to MPU Subsystem

IRQ	Source	Description
M_IRQ_72	UART1_IRQ	UART module 1 interrupt to MPU
M_IRQ_73	UART2_IRQ	UART module 2 interrupt to MPU
M_IRQ_74	UART3_IRQ	UART module 3 interrupt to MPU
M_IRQ_84	UART4_IRQ	UART module 4 interrupt to MPU

14.3.2.2 DMA Requests

Table 14-14 describes the UART DMA requests.

Table 14-14. UART DMA Requests to System DMA

DMA[1]	Source	Description
S_DMA_48	UART1_DMA_TX	UART module 1 transmit request
S_DMA_49	UART1_DMA_RX	UART module 1 receive request
S_DMA_50	UART2_DMA_TX	UART module 2 transmit request
S_DMA_51	UART2_DMA_RX	UART module 2 receive request
S_DMA_52	UART3_DMA_TX	UART module 3 transmit request
S_DMA_53	UART3_DMA_RX	UART module 3 receive request
S_DMA_54	UART4_DMA_TX	UART module 4 transmit request
S_DMA_55	UART4_DMA_RX	UART module 4 receive request

(1) This table assumes that DMA mode 1 is used.

14.3.2.3 Wake-up Request

The UART can wake up the system on different events (see Section 14.6, *UART/IrDA/CIR Registers*).

One important wake-up generation is event detection on the uarti_cts pin when the system is idle. The uarti_cts pin uses an asynchronous path to transmit the wake-up request to the system (PRCM), which enables wake-up capabilities to be active even if all module clocks are off (see Table 14-15).

Table 14-15. Wake-Up Requests from PRCM

Wake-Up Pin	PRCM Input	Description
uart1_cts	UART1_SWAKEUP	Wake UART1 (system wake-up capabilities)
uart2_cts	UART2_SWAKEUP	Wake UART2 (system wake-up capabilities)
uart3_cts	UART3_SWAKEUP	Wake UART3 (system wake-up capabilities)

CAUTION

UARTs are not in the WAKEUP power domain, which implies limitations on wake-up capability. If the CORE power domain is off, UART1 and UART2 cannot wake up the system, because power is not supplied. If the PER power domain is off, UART3 cannot wake up the system, because power is not supplied. In these cases, a GPIO multiplexed on the uarti_cts pin can be used to capture the event and wake up the system. Software must enable this strategy, if required. See the *General-Purpose Interface* chapter for a full description of this mechanism.

14.4 UART/IrDA/CIR Functional Description

14.4.1 UART/IrDA/CIR Block Description

The UART/IrDA/CIR module can be divided into three main blocks:

- FIFO management
- Mode selection
- Protocol formatting

FIFO management is common to all functions and enables the transmission and reception of data from the host processor point of view.

There are two mode selections:

- Function mode selection: Route the data to the chosen functionality (UART, IrDA, or CIR) and enable the mechanism corresponding to the chosen functionality.
- Register mode selection, which enables conditional access to registers

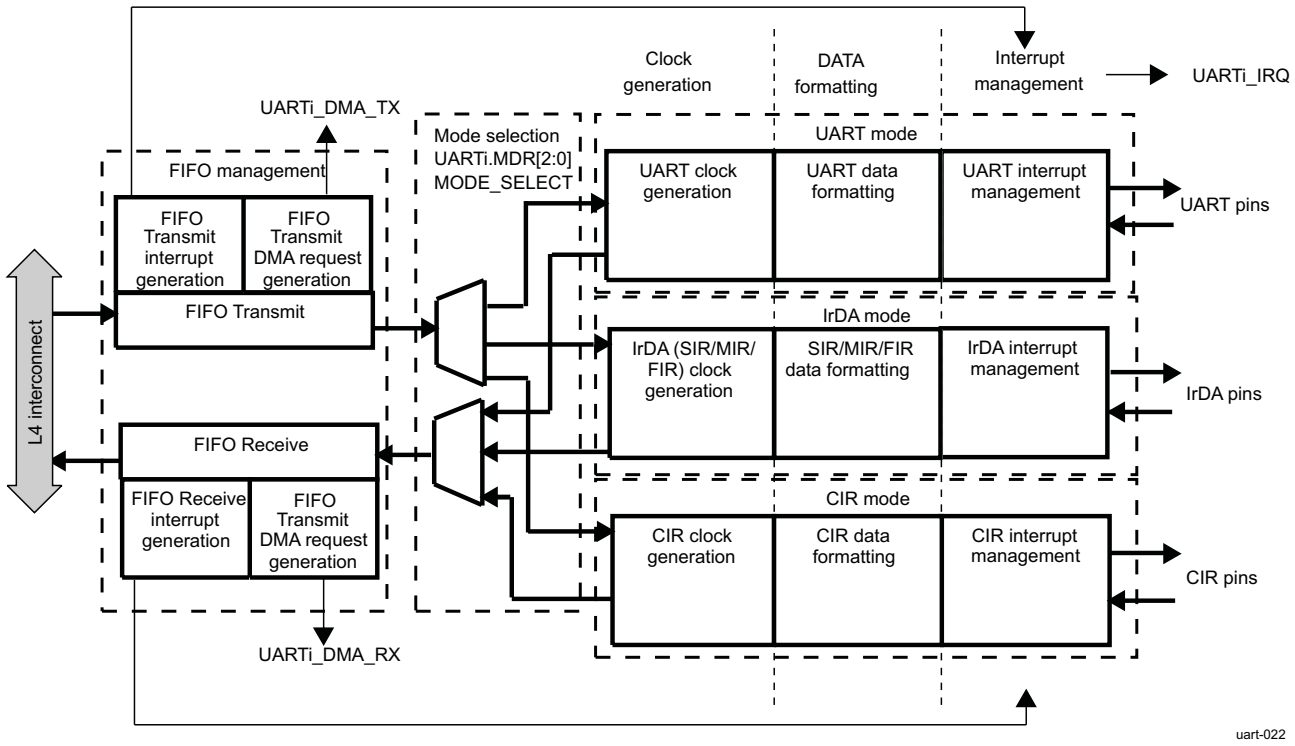
Protocol formatting has three subcategories:

- Clock generation: The 48-MHz input clock generates all necessary clocks.
- Data formatting: Each function uses its own state-machine, which assumes the transition between FIFO data and frame data.
- Interrupt management: Different interrupt types are generated depending on the chosen function.

In parallel with these functional blocks, a power-saving strategy exists for each function.

Figure 14-21 is the UART/IrDA/CIR block diagram.

Figure 14-21. UART/IrDA/CIR Block Diagram



14.4.2 FIFO Management

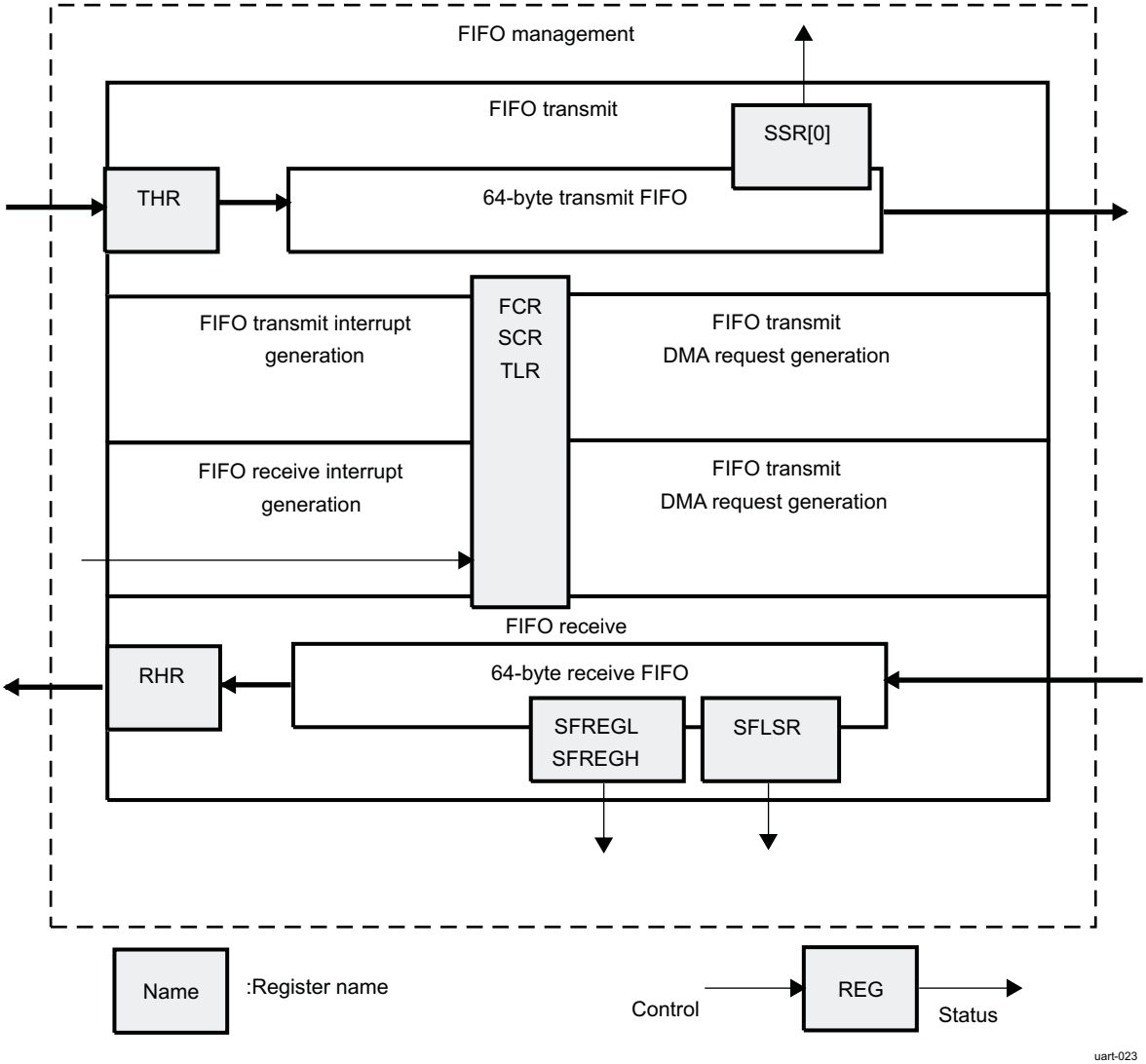
The FIFO is accessed by reading/writing the UARTi.RHR_REG and UARTi.THR_REG registers. Parameters are controlled using the FIFO control register (UARTi.FCR_REG) and supplementary control register (UARTi.SCR_REG). Reading the UARTi.SSR_REG[0] TX_FIFO_FULL bit at 1 means the FIFO is full.

The UARTi.TLR_REG register controls the FIFO trigger level, which enables the DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled; so, in effect, the trigger level is the default value of 1 byte. Figure 14-22 shows the FIFO management registers.

NOTE: Data in the UARTi.RHR_REG register is not overwritten when an overflow occurs.

NOTE: The UARTi.SFLSR_REG, UARTi.SFREGL_REG, and UARTi.SFREGH_REG status registers are used in IrDA mode only. For use, see Section 14.4.4.2.3, *IrDA Data Formatting*.

Figure 14-22. FIFO Management Registers



14.4.2.1 FIFO Trigger

14.4.2.1.1 Transmit FIFO Trigger

Table 14-16 summarizes the transmit FIFO trigger level settings.

Table 14-16. TX FIFO Trigger Level Setting Summary

SCR_REG[6]	TLR_REG[3:0]	TX FIFO Trigger Level
0	= 0x0	Defined by the UARTi.FCR_REG[5:4] TX_FIFO_TRIG field (8,16, 32, or 56 spaces)
0	≠ 0x0	Defined by the UARTi.TLR_REG[3:0] TX_FIFO_TRIG_DMA field (from 4 to 60 spaces with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TX_FIFO_TRIG_DMA and TX_FIFO_TRIG (from 1 to 63 spaces with a granularity of 1 space) **Note:** The combination of TX_FIFO_TRIG_DMA = 0x0 and TX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of one space required). All zeros result in unpredictable behavior.

14.4.2.1.2 Receive FIFO Trigger

Table 14-17 summarizes the receive FIFO trigger level settings.

Table 14-17. RX FIFO Trigger Level Setting Summary

SCR_REG[7]	TLR_REG[7:4]	RX FIFO Trigger Level
0	= 0x0	Defined by the UARTi.FCR_REG[7:6] RX_FIFO_TRIG field (8,16, 56, or 60 characters)
0	≠ 0x0	Defined by UARTi.TLR_REG[7:4] RX_FIFO_TRIG_DMA field (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of RX_FIFO_TRIG_DMA and RX_FIFO_TRIG (from 1 to 63 characters with a granularity of 1 character). **Note:** The combination of RX_FIFO_TRIG_DMA = 0x0 and RX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum 1 character required). All zeros result in unpredictable behavior.

The receive threshold is programmed using the UARTi.TCR_REG[7:4] RX_FIFO_TRIG_START and UARTi.TCR_REG[3:0] RX_FIFO_TRIG_HALT fields.

- Trigger levels from 0 to 60 bytes are available with a granularity of four. (Trigger level = 4 x [4-bit register value])
- To ensure correct device operation, ensure that RX_FIFO_TRIG_HALT > RX_FIFO_TRIG when auto-RTS is enabled.
- In the FIFO interrupt mode with flow control, ensure that the trigger level to HALT transmission is greater than or equal to the receive FIFO trigger level (either the UARTi.TCR_REG[7:4] RX_FIFO_TRIG_START field or the UARTi.FCR_REG[7:6] RX_FIFO_TRIG field); otherwise, FIFO operation stalls. In the FIFO DMA mode with flow control, this concept does not exist, because a DMA request is sent when a byte is received.

14.4.2.2 FIFO Interrupt Mode

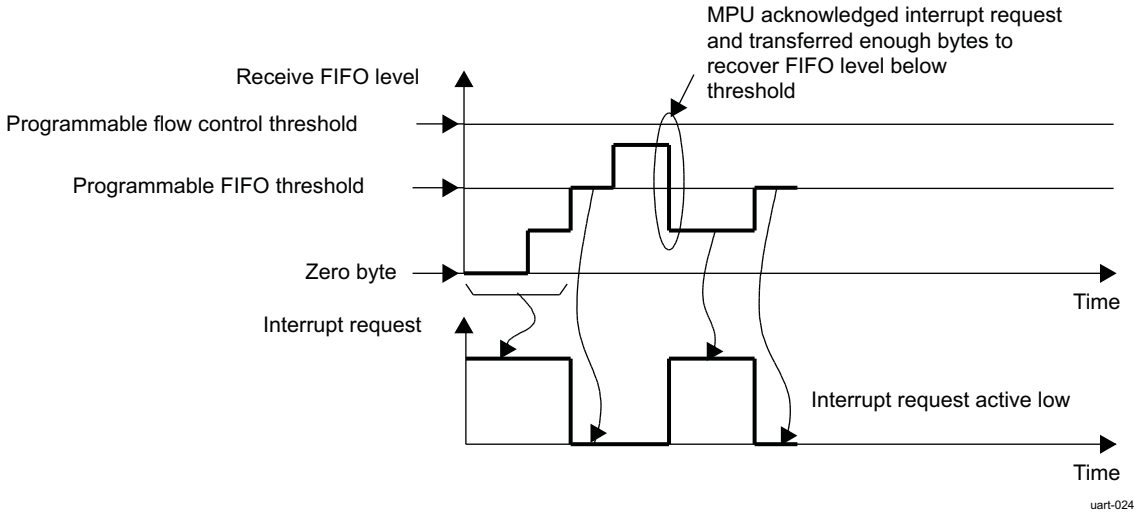
In the FIFO interrupt mode (the FIFO control register UARTi.FCR_REG[0] FIFO_EN bit is set to 1 and relevant interrupts are enabled by the UARTi.IER_REG register), an interrupt signal informs the processor of the status of the receiver and transmitter. These interrupts are raised when the receive/transmit FIFO threshold (the UARTi.TLR_REG[7:4] RX_FIFO_TRIG_DMA and UARTi.TLR_REG[3:0] TX_FIFO_TRIG_DMA fields or the UARTi.FCR_REG[7:6] RX_FIFO_TRIG and UARTi.FCR_REG[5:4] TX_FIFO_TRIG fields, respectively) is reached.

The interrupt signals instruct the MPU to transfer data to the destination (from the UART module in receive mode and/or from any source to the UART FIFO in transmit mode).

When the UART flow control is enabled with the interrupt capabilities, the UART flow control FIFO threshold (UARTi.TCR_REG[3:0] RX_FIFO_TRIG_HALT field) must be greater than or equal to the receive FIFO threshold.

Figure 14-23 shows the generation of the receive FIFO interrupt request.

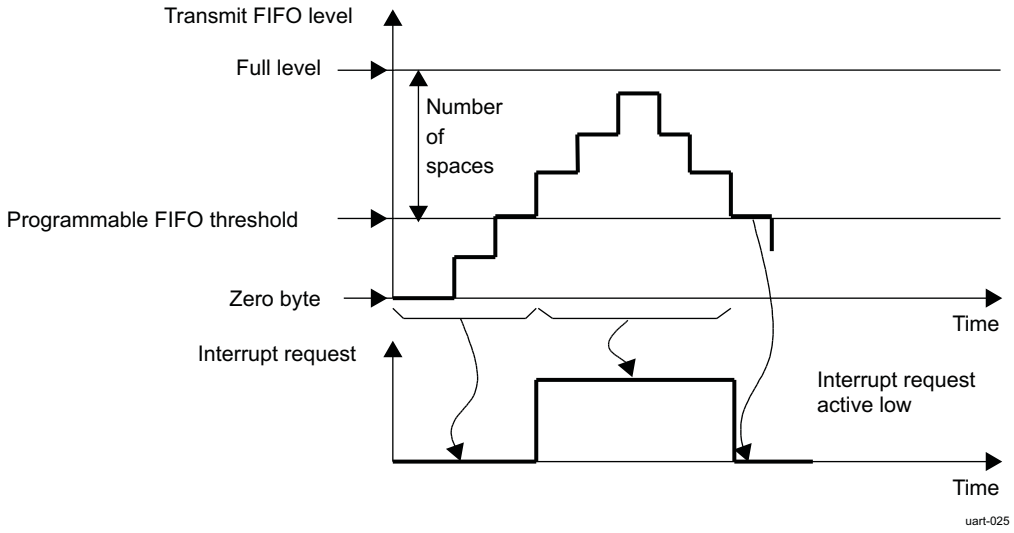
Figure 14-23. Receive FIFO Interrupt Request Generation



In receive mode, no interrupt is generated until the receive FIFO reaches its threshold. Once low, the interrupt can be deasserted only when the MPU has handled enough bytes to make the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

Figure 14-24 shows the generation of the transmit FIFO interrupt request.

Figure 14-24. Transmit FIFO Interrupt Request Generation



In transmit mode, an interrupt request is automatically asserted when the TX FIFO is empty. This request is deasserted when the TX FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements is transmitted to go below the TX FIFO threshold.

14.4.2.3 FIFO Polled Mode Operation

In the FIFO polled mode (the UARTi.FCR_REG[0] FIFO_EN bit is set to 0 and the relevant interrupts are disabled by the UARTi.IER_REG register), the status of the receiver and transmitter can be checked by polling the line status register (UARTi.LSR_REG).

This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically determined by sending interrupts to the MPU.

14.4.2.4 FIFO DMA Mode Operation

Although DMA operation includes four modes (DMA modes 0/1/2/3), the information in Table 14-14 assumes that mode 1 is used. (Mode 2 and mode 3 are legacy modes that use only one DMA request for each module.)

In mode 2, the remaining DMA request is used for RX. In mode 3, the remaining DMA request is used for TX.

The DMA requests in mode 2 and mode 3 use S_DMA_48, S_DMA_50, and S_DMA_52/D_DMA_10. S_DMA_49, S_DMA_51, and S_DMA_53/D_DMA_11 are not used by the module in mode 2 and mode 3 and can be selected as follows:

- When the UARTi.SCR_REG[0] DMA_MODE_CTL bit is set to 0, setting the UARTi.FCR_REG[3] DMA_MODE bit to 0 enables DMA mode 0. Setting the DMA_MODE bit to 1 enables DMA mode 1.
- When the DMA_MODE_CTL bit is set to 1, the UARTi.FCR_REG[2:1] DMA_MODE_2 field determines DMA mode 0 to 3 based on the supplementary control register (SCR_REG) description.

For example:

- If no DMA operation is desired, set the DMA_MODE_CTL bit to 1 and the DMA_MODE_2 field to 0x0. (The DMA_MODE bit is discarded.)
- If DMA mode 1 is desired, set either the DMA_MODE_CTL bit to 0 and the DMA_MODE bit to 1, or set the DMA_MODE_CTL bit to 1 and the DMA_MODE_2 field to 01. (The DMA_MODE bit is discarded.)

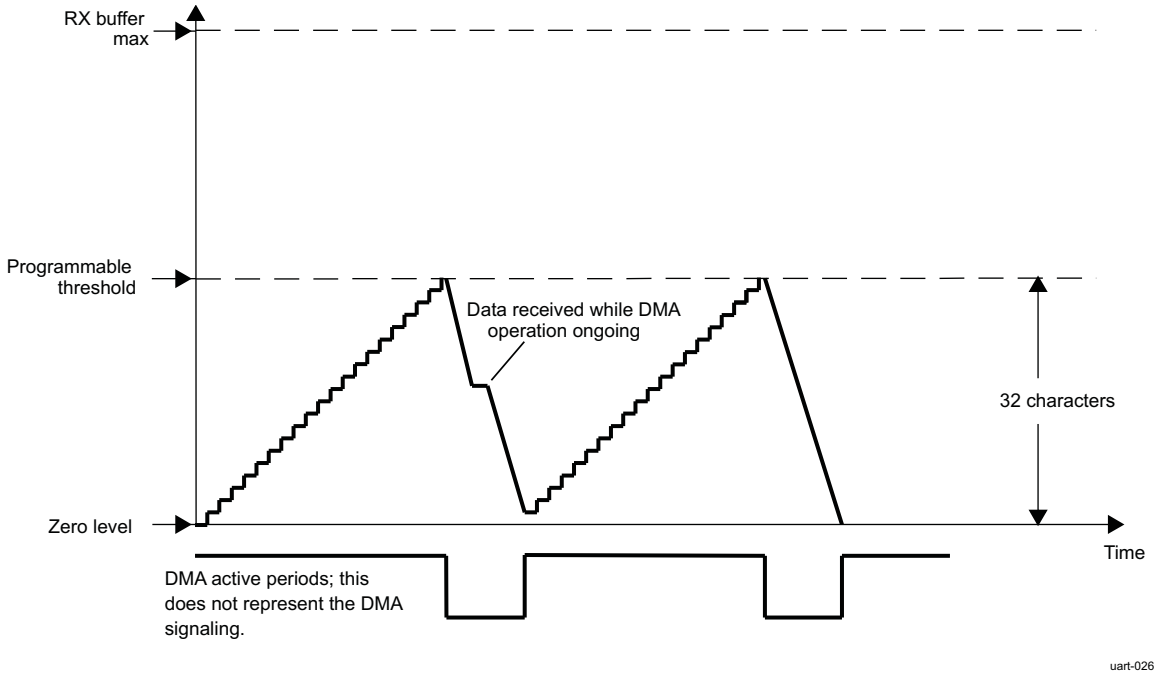
If the FIFOs are disabled (the UARTi.FCR_REG[0] FIFO_EN bit = 0), the DMA occurs in single-character transfers.

When DMA mode 0 is programmed, the signals associated with DMA operation are not active.

14.4.2.4.1 DMA Transfers (DMA Mode 1, 2, or 3)

Figure 14-25 through Figure 14-28 show the supported DMA operations.

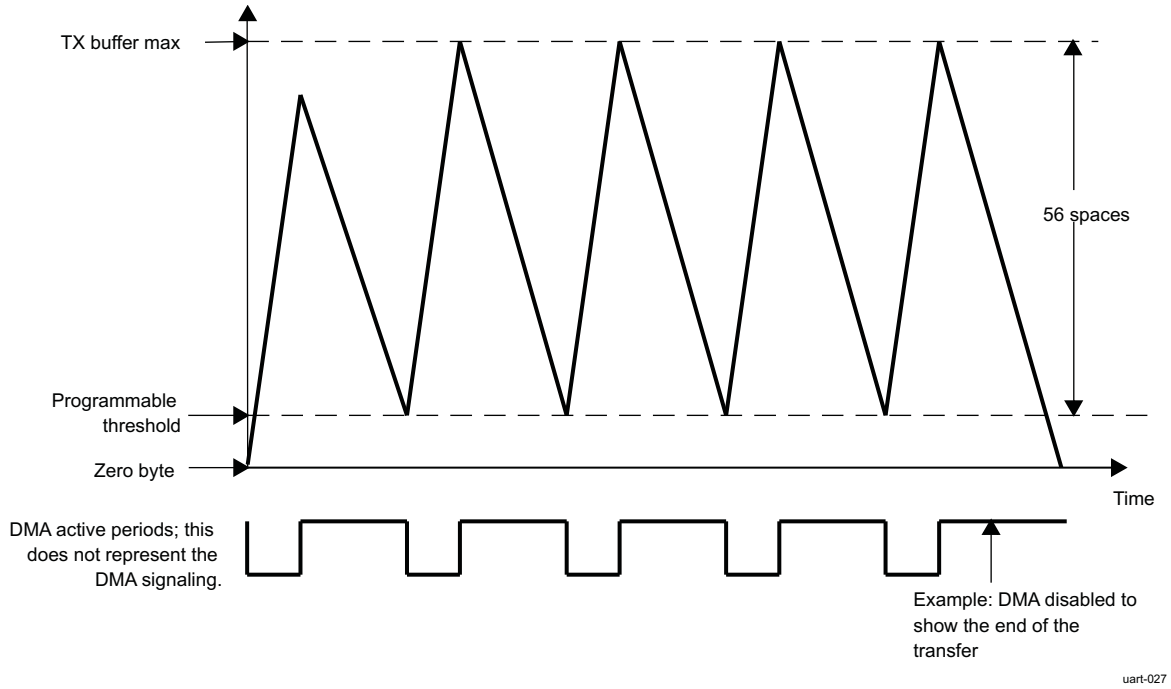
Figure 14-25. Receive FIFO DMA Request Generation (32 Characters)



In receive mode, a DMA request is generated when the receive FIFO reaches its threshold level defined in the trigger level register (UARTi.TLR_REG). This request is deasserted when the number of bytes defined by the threshold level is read by the system DMA (sDMA).

In transmit mode, a DMA request is automatically asserted when the transmit FIFO is empty. This request is deasserted when the number of bytes defined by the number of spaces in the trigger level register (UARTi.TLR_REG) is written by the sDMA. If an insufficient number of characters is written, the DMA request stays active.

Figure 14-26. Transmit FIFO DMA Request Generation (56 Spaces)



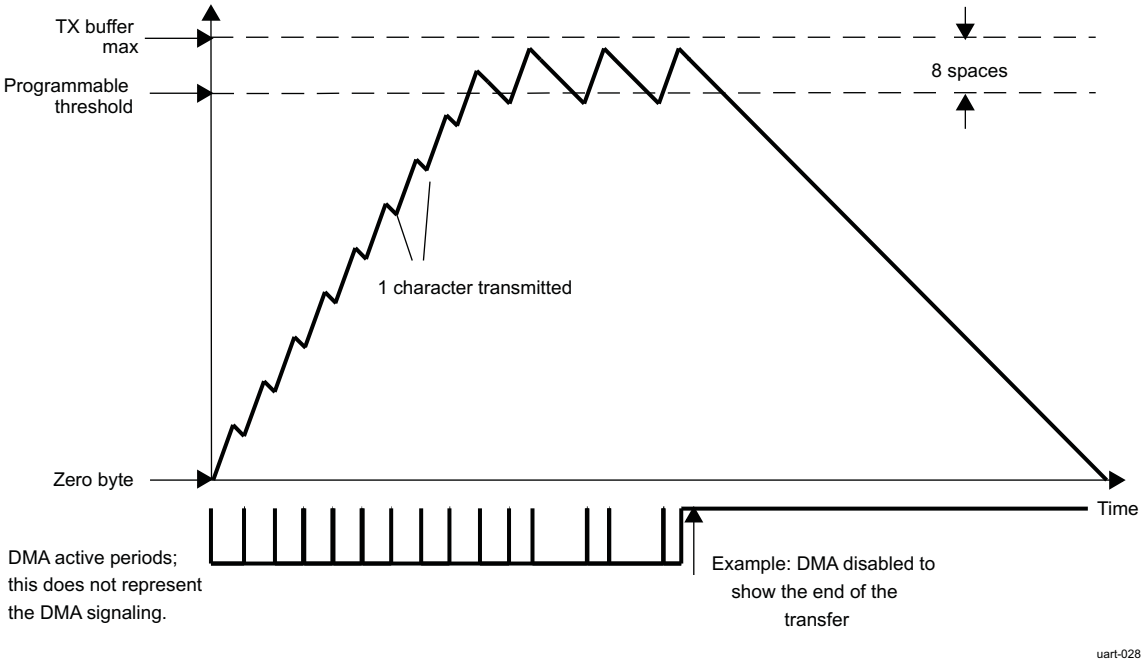
The DMA request is again asserted if the FIFO can receive the number of bytes defined by the UARTi.TLR_REG register.

The threshold can be programmed in a number of ways. Figure 14-26 shows a DMA transfer operating with a space setting of 56 that can arise from using the auto settings in the UARTi.FCR_REG[5:4] TX_FIFO_TRIG field or the UARTi.TLR_REG[3:0] TX_FIFO_TRIG_DMA field concatenated with the TX_FIFO_TRIG field.

The setting of 56 spaces in the UART/IrDA/CIR module must correlate with the settings of the sDMA so that the buffer does not overflow (program the DMA request size of the local host controller to be equal to the number of spaces value in the UART/IrDA/CIR module).

Figure 14-27 shows an example with eight spaces to show the buffer level crossing the space threshold. Again, the local host DMA controller settings must correspond to that of the UART/IrDA/CIR module.

Figure 14-27. Transmit FIFO DMA Request Generation (8 Spaces)

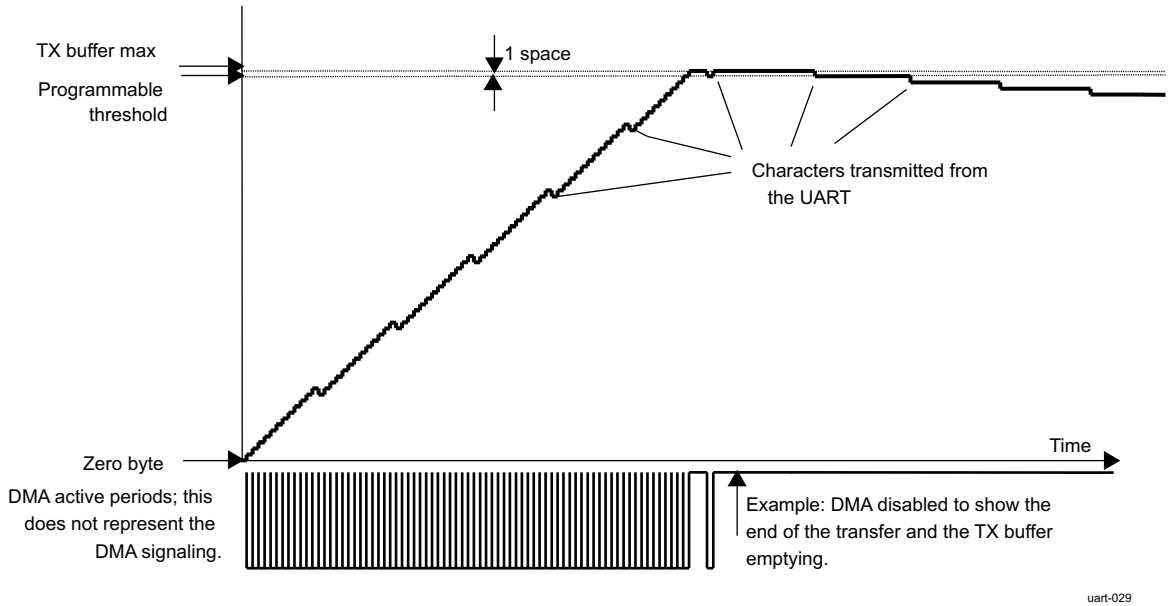


The final example shows the setting of one space that uses the DMA for each transfer of one character to the transmit buffer (see Figure 14-28). The buffer is filled at a faster rate than the baud rate at which data is transmitted to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer.

On two occasions the buffer holds the maximum amount of data words; shortly after this, the DMA is disabled to show the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud rate settings of the UARTi.DLL_REG and the UARTi.DLH_REG registers.

Again, the DMA settings must correspond to the system local host DMA controller settings to ensure correct operation of this logic.

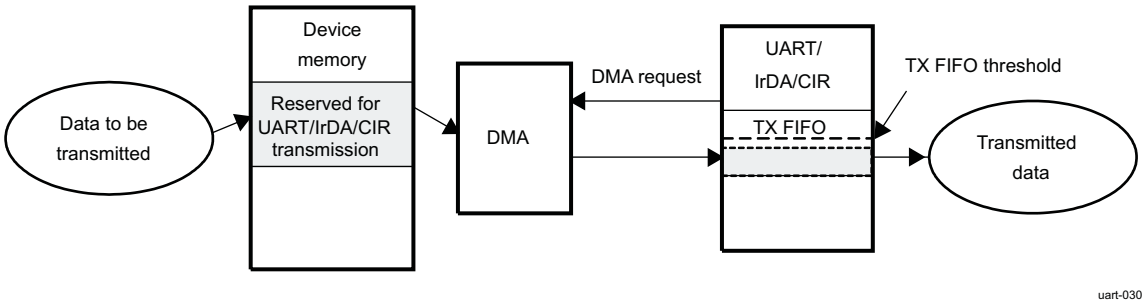
Figure 14-28. Transmit FIFO DMA Request Generation (1 Space)



14.4.2.4.2 DMA Transmission

Figure 14-29 shows the DMA transmission process.

Figure 14-29. Transmission Process



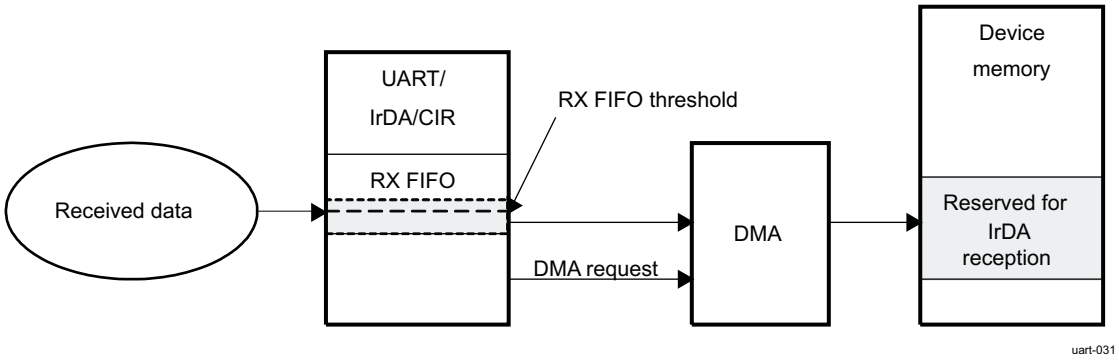
- Data to be transmitted are put in the device's memory reserved for UART/IrDA/CIR transmission by the DMA:
  - Until the TX FIFO trigger level is not reached, a DMA request is generated.
  - An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
- Data in the TX FIFO are automatically transmitted.
- The end of the transmission is signaled by the UARTi.THR_REG empty (TX FIFO empty).

NOTE: In IrDA mode, the transmission is not ended immediately after the TX FIFO empties, at which point there are still the last data byte, the CRC field, and the stop flag to be transmitted; thus, the end of transmission is a few milliseconds after the UARTi.THR_REG register empties.

14.4.2.4.3 DMA Reception

Figure 14-30 shows the DMA reception process.

Figure 14-30. Reception Process



1. Enable the reception.
2. Received data are put in the RX FIFO.
3. Data are transferred from the RX FIFO to the device memory by the DMA.
   - At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.
   - An element (1 byte) is transferred from the RX FIFO to the SDRAM at each DMA request (DMA element synchronization).
4. The end of the reception is signaled by the EOF interrupt.

14.4.3 Mode Selection

14.4.3.1 Register Access Modes

14.4.3.1.1 Operational Mode and Configuration Modes

Register access depends on the register access mode, although register access modes are not correlated to functional mode selection. Three different modes are available:

- Operational mode
- Configuration mode A
- Configuration mode B

Operational mode is the selected mode when the function is active; serial data transfer can be performed in this mode.

Both configuration mode A and configuration mode B are used during module initialization steps. These modes enable access to configuration registers, which are hidden in the operational mode. The modes are used when the module is inactive (no serial data transfer processed) and only in the initialization step or reconfiguration of the module.

The value of the UARTi.LCR_REG register determines the register access mode (see Table 14-18).

Table 14-18. UART/IrDA/CIR Register Access Mode Programming (Using LCR_REG)

Mode	Condition
Configuration_mode_A	LCR_REG[7] = 0x1 and LCR_REG[7:0]! = 0xBF
Configuration_mode_B	LCR_REG[7] = 0x1 and LCR_REG[7:0] = 0xBF
Operational_mode	LCR_REG[7] = 0x0

14.4.3.1.2 Register Access Submode

In each access register mode (operational mode or configuration mode A/B), some register accesses are conditional to the programming of a submode (MSR_SPR, TCR_TLR, and XOFF). These registers are identified in Section 14.6, *UART/IrDA/CIR Registers*.

Table 14-19 through Table 14-21 summarize the register access submodes.

Table 14-19. Sub-Configuration_Mode_A Mode Summary

Mode	Condition
MSR_SPR	(EFR_REG[4] = 0x0 or MCR_REG[6] = 0x0)
TCR_TLR	EFR_REG[4] = 0x1 and MCR_REG[6] = 0x1

Table 14-20. Sub-Configuration_Mode_B Mode Summary

Mode	Condition
TCR_TLR	EFR_REG[4] = 0x1 and MCR_REG[6] = 0x1
XOFF	(EFR_REG[4] = 0x0 or MCR_REG[6] = 0x0)

Table 14-21. Sub-Operational_Mode Mode Summary

Mode	Condition
MSR_SPR	EFR_REG[4] = 0x0 or MCR_REG[6] = 0x0)
TCR_TLR	EFR_REG[4] = 0x1 and MCR_REG[6] = 0x1

14.4.3.1.3 Registers Available for the Register Access Modes

Table 14-22 lists the names of the register hits in each access register mode. Gray-shaded registers do not depend on the register access mode (available in all modes).

Table 14-22. UART/IrDA/CIR Register Access Mode Overview

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x000	DLL_REG	DLL_REG	DLL_REG	DLL_REG	RHR_REG	THR_REG
0x004	DLH_REG	DLH_REG	DLH_REG	DLH_REG	IER_REG	IER_REG
0x008	IIR_REG	FCR_REG	EFR_REG	EFR_REG	IIR_REG	FCR_REG
0x00C	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG
0x010	MCR_REG	MCR_REG	XON1_ADDR1_REG	XON1_ADDR1_REG	MCR_REG	MCR_REG
0x014	LSR_REG	-	XON2_ADDR2_REG	XON2_ADDR2_REG	LSR_REG	-
0x018	MSR_REG(1)/TCR_REG(2)	TCR_REG(2)	TCR_REG(2)/XOFF1_REG(3)	TCR_REG(2)/XOFF1_REG(3)	MSR_REG(1)/TCR_REG(2)	TCR_REG(2)
0x01C	SPR_REG(1)/TLR_REG(2)	SPR_REG(1)/TLR_REG(2)	TLR_REG(2)/XOFF2_REG(3)	TLR_REG(2)/XOFF2_REG(3)	SPR_REG(1)/TLR_REG(2)	SPR_REG(1)/TLR_REG(2)
0x020	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG
0x024	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG
0x028	SFLSR_REG	TXFLL_REG	SFLSR_REG	TXFLL_REG	SFLSR_REG	TXFLL_REG
0x02C	RESUME_REG	TXFLH_REG	RESUME_REG	TXFLH_REG	RESUME_REG	TXFLH_REG
0x030	SFREGL_REG	RXFLL_REG	SFREGL_REG	RXFLL_REG	SFREGL_REG	RXFLL_REG
0x034	SFREGH_REG	RXFLH_REG	SFREGH_REG	RXFLH_REG	SFREGH_REG	RXFLH_REG
0x038	UASR_REG	-	UASR_REG	-	BLR_REG	BLR_REG
0x03C	-	-	-	-	ACREG_REG	ACREG_REG
0x040	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG
0x044	SSR_REG	-	SSR_REG	-	SSR_REG	-
0x048	-	-	-	-	EBLR_REG	EBLR_REG
0x050	-	-	-	-	-	-
0x054	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG
0x058	SYSS_REG	-	SYSS_REG	-	SYSS_REG	-
0x05C	WER_REG	WER_REG	WER_REG	WER_REG	WER_REG	WER_REG
0x060	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG

(1) if MSR_SPR mode is active (see Section 14.4.3.1.2, *Register Access Submode*)
(2) if TCR_TLR mode is active (see Section 14.4.3.1.2, *Register Access Submode*)
(3) if XOFF mode is active (see Section 14.4.3.1.2, *Register Access Submode*)

14.4.3.2 UART/IrDA (SIR, MIR, FIR)/CIR Mode Selection

To select a mode, set the UARTi.MDR1_REG[2:0] MODE_SELECT field (see Table 14-23).

Table 14-23. UART Mode Selection

Value	Mode
0x0:	UART 16x mode
0x1:	SIR mode (UART3 only)
0x2:	UART 16x auto-baud
0x3:	UART 13x mode
0x4:	MIR mode (UART3 only)
0x5:	FIR mode (UART3 only)
0x6:	CIR mode (UART3 only)

MODE_SELECT is effective when the module is in operational mode (see Section 14.4.3.1, *Register Access Modes*).

14.4.3.2.1 Registers Available for the UART Function

Only the registers listed in Table 14-24 are used for the UART function.

Table 14-24. UART Mode Register Overview(1) (2)

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x000	DLL_REG	DLL_REG	DLL_REG	DLL_REG	RHR_REG	THR_REG
0x004	DLH_REG	DLH_REG	DLH_REG	DLH_REG	IER_REG(UART)	IER_REG(UART)
0x008	IIR_REG	FCR_REG	EFR_REG[4]	EFR_REG[4]	IIR_REG(UART)	FCR_REG(UART)
0x00C	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG
0x010	MCR_REG	MCR_REG	XON1_ADDR1_REG	XON1_ADDR1_REG	MCR_REG	MCR_REG
0x014	LSR_REG(UART)	-	XON2_ADDR2_REG	XON2_ADDR2_REG	LSR_REG(UART)	-
0x018	MSR_REG/TCR_REG	TCR_REG	XOFF1_REG/TCR_REG	XOFF1_REG/TCR_REG	MSR_REG/TCR_REG	TCR_REG
0x01C	TLR_REG/SPR_REG	TLR_REG/SPR_REG	TLR_REG/XOFF2_REG	TLR_REG/XOFF2_REG	TLR_REG/SPR_REG	TLR_REG/SPR_REG
0x020	MDR1_REG	MDR1_REG[2:0]	MDR1_REG[2:0]	MDR1_REG[2:0]	MDR1_REG[2:0]	MDR1_REG[2:0]
0x024	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG
0x028	-	-	-	-	-	-
0x02C	-	-	-	-	-	-
0x030	-	-	-	-	-	-
0x034	-	-	-	-	-	-
0x038	UASR_REG	-	UASR_REG	-	-	-
0x03C	-	-	-	-	-	-
0x040	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG
0x044	SSR_REG	-	SSR_REG	-	SSR_REG	-
0x048	-	-	-	-	-	-

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions (described separately in Section 14.6, *UART/IrDA/CIR Registers*).

(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 14-24. UART Mode Register Overview[1] [2] (continued)

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x050	-	-	-	-	-	-
0x054	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG
0x058	SYSS_REG	-	SYSS_REG	-	SYSS_REG	-
0x05C	WER_REG	WER_REG	WER_REG	WER_REG	WER_REG	WER_REG
0x060	-	-	-	-	-	-

14.4.3.2.2 Registers Available for the IrDA Function (UART3 Only)

Only the registers listed in Table 14-25 are used for the IrDA function.

Table 14-25. IrDA Mode Register Overview[1] [2]

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x000	DLL_REG	DLL_REG	DLL_REG	DLL_REG	RHR_REG	THR_REG
0x004	DLH_REG	DLH_REG	DLH_REG	DLH_REG	IER_REG(IrDA)	IER_REG(IrDA)
0x008	IIR_REG	FCR_REG	EFR_REG[4]	EFR_REG[4]	IIR_REG(IrDA)	FCR_REG(IrDA)
0x00C	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG
0x010	-	-	XON1_ADDR1_REG	XON1_ADDR1_REG	MCR_REG	MCR_REG
0x014	LSR_REG(IrDA)	-	XON2_ADDR2_REG	XON2_ADDR2_REG	LSR_REG(IrDA)	-
0x018	MSR_REG/TCR_REG	TCR_REG	TCR_REG	TCR_REG	MSR_REG/TCR_REG	TCR_REG
0x01C	TLR_REG/SPR_REG	TLR_REG/SPR_REG	TLR_REG	TLR_REG	TLR_REG/SPR_REG	TLR_REG/SPR_REG
0x020	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG
0x024	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG
0x028	SFLSR_REG	TXFLL_REG	SFLSR_REG	TXFLL_REG	SFLSR_REG	TXFLL_REG
0x02C	RESUME_REG	TXFLH_REG	RESUME_REG	TXFLH_REG	RESUME_REG	TXFLH_REG
0x030	SFREGL_REG	RXFLL_REG	SFREGL_REG	RXFLL_REG	SFREGL_REG	RXFLL_REG
0x034	SFREGH_REG	RXFLH_REG	SFREGH_REG	RXFLH_REG	SFREGH_REG	RXFLH_REG
0x038	-	-	-	-	BLR_REG	BLR_REG
0x03C	-	-	-	-	ACREG_REG	ACREG_REG
0x040	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG
0x044	SSR_REG	-	SSR_REG	-	SSR_REG	-
0x048	-	-	-	-	EBLR_REG	EBLR_REG
0x050	-	-	-	-	-	-
0x054	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG
0x058	SYSS_REG	-	SYSS_REG	-	SYSS_REG	-

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions (described separately in Section 14.6, *UART/IrDA/CIR Registers*).

(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 14-25. IrDA Mode Register Overview[1] [2] (continued)

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x05C	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]
0x060	-	-	-	-	-	-

14.4.3.2.3 Registers Available for the CIR Function (UART3 Only)

Only the registers listed in Table 14-26 are used for CIR function.

Table 14-26. CIR Mode Register Overview[1] [2]

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x000	DLL_REG	DLL_REG	DLL_REG	DLL_REG	-	THR_REG
0x004	DLH_REG	DLH_REG	DLH_REG	DLH_REG	IER_REG(CIR)	IER_REG(CIR)
0x008	IIR_REG	FCR_REG	EFR_REG	EFR_REG	IIR_REG(CIR)	FCR_REG(CIR)
0x00C	LCR_REG	LCR_REG[7]	LCR_REG[7]	LCR_REG[7]	LCR_REG[7]	LCR_REG[7]
0x010	-	-	-	-	-	-
0x014	LSR_REG(IrDA)	-	-	-	LSR_REG(IrDA)	-
0x018	MSR_REG/TCR_REG	TCR_REG	TCR_REG	TCR_REG	MSR_REG/TCR_REG	TCR_REG
0x01C	TLR_REG/SPR_REG	TLR_REG/SPR_REG	TLR_REG	TLR_REG	TLR_REG/SPR_REG	TLR_REG/SPR_REG
0x020	MDR1_REG[3:0]	MDR1_REG[3:0]	MDR1_REG[3:0]	MDR1_REG[3:0]	MDR1_REG[3:0]	MDR1_REG[3:0]
0x024	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG
0x028	-	-	-	-	-	-
0x02C	RESUME_REG	-	RESUME_REG	-	RESUME_REG	-
0x030	-	-	-	-	-	-
0x034	-	-	-	-	-	-
0x038	-	-	-	-	-	-
0x03C	-	-	-	-	ACREG_REG	ACREG_REG
0x040	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG
0x044	SSR_REG	-	SSR_REG	-	SSR_REG	-
0x048	-	-	-	-	EBLR_REG	EBLR_REG
0x050	-	-	-	-	-	-
0x054	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG
0x058	SYSS_REG	-	SYSS_REG	-	SYSS_REG	-
0x05C	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]	WER_REG[6:4]
0x060	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions (described separately in Section 14.6, *UART/IrDA/CIR Registers*).

(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

14.4.4 Protocol Formatting

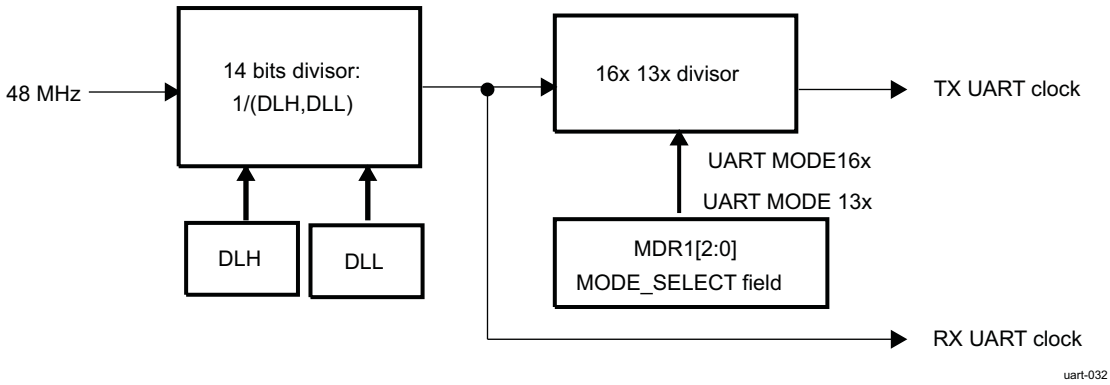
14.4.4.1 UART Mode

14.4.4.1.1 UART Clock Generation: Baud Rate Generation

The UART function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 14-31 shows the baud rate generator and associated controls.

Figure 14-31. Baud Rate Generation



CAUTION

It is mandatory that MODE_SELECT = DISABLE (UARTi.MDR1_REG[2:0] = 0x7) before initializing or modifying clock parameter controls (UARTi.DLH_REG, UARTi.DLL_REG). Failure to observe this rule can result in unpredictable module behavior.

14.4.4.1.2 Choosing the Appropriate Divisor Value

Two divisor values are:

- UART 16x mode: Divisor value = Operating frequency/(16x baud rate)
- UART 13x mode: Divisor value = Operating frequency/(13x baud rate)

Table 14-27 describes the UART baud rate settings.

Table 14-27. UART Baud Rate Settings (48-MHz Clock)

Baud Rate	Baud Multiple	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
0.3 Kbps	16x	10000	0x27, 0x10	0.3 Kbps	0
0.6 Kbps	16x	5000	0x13, 0x88	0.6 Kbps	0
1.2 Kbps	16x	2500	0x09, 0xC4	1.2 Kbps	0
2.4 Kbps	16x	1250	0x04, 0xE2	2.4 Kbps	0
4.8 Kbps	16x	625	0x02, 0x71	4.8 Kbps	0
9.6 Kbps	16x	312	0x01, 0x38	9.6153 Kbps	+0.16
14.4 Kbps	16x	208	0x00, 0xD0	14.423 Kbps	+0.16
19.2 Kbps	16x	156	0x00, 0x9C	19.231 Kbps	+0.16
28.8 Kbps	16x	104	0x00, 0x68	28.846 Kbps	+0.16
38.4 Kbps	16x	78	0x00, 0x4E	38.462 Kbps	+0.16
57.6 Kbps	16x	52	0x00, 0x34	57.692 Kbps	+0.16
115.2 Kbps	16x	26	0x00, 0x1A	115.38 Kbps	+0.16

Table 14-27. UART Baud Rate Settings (48-MHz Clock) (continued)

Baud Rate	Baud Multiple	DLH, DLL (Decimal)	DLH, DLL (Hex)	Actual Baud Rate	Error (%)
230.4 Kbps	16x	13	0x00, 0x0D	230.77 Kbps	+0.16
460.8 Kbps	13x	8	0x00, 0x08	461.54 Kbps	+0.16
921.6 Kbps	13x	4	0x00, 0x04	923.08 Kbps	+0.16
1.843 Mbps	13x	2	0x00, 0x02	1.846 Mbps	+0.16
3.6884 Mbps	13x	1	0x00, 0x01	3.6923 Mbps	+0.16

14.4.4.1.3 UART Data Formatting

The UART module can use hardware flow control to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the RTS output and CTS input signals.

The UART module is enhanced with the autobauding function. In control mode, autobauding allows the speed, the number of bits per character, and the parity selected to be set automatically.

14.4.4.1.3.1 Frame Formatting

When autobauding is not used, frame format attributes must be defined in the UARTi.LCR_REG register.

Character length is specified using the UARTi.LCR_REG[1:0] CHAR_LENGTH field.

The number of stop bits is specified using the UARTi.LCR_REG[2] NB_STOP register bit.

The parity bit is programmed using the UARTi.LCR_REG[5:3] PARITY_EN, PARITY_TYPE_1, and PARITY_TYPE_2 register bits (see Table 14-28).

Table 14-28. UART Parity Bit Encoding

PARITY_EN	PARITY_TYPE1	PARITY_TYPE2	Parity
0	N/A	N/A	No parity
1	0	0	Odd parity
1	1	0	Even parity
1	0	1	Forced 1
1	1	1	Forced 0

14.4.4.1.3.2 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled/disabled independently by programming the UARTi.EFR_REG[7:6] AUTO_CTS_EN and AUTO_RTS_EN bits, respectively.

With auto-CTS, uarti_cts must be active before the module can transmit data.

Auto-RTS activates the uarti_rts output only when there is enough room in the RX FIFO to receive data. It deactivates the uarti_rts output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the UARTi.TCR_REG register determine the levels at which uarti_rts is activated/deactivated.

If both auto-CTS and auto-RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If auto-CTS and auto-RTS are not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

- **Auto-RTS**

  Auto-RTS data flow control originates in the receiver block. The receiver FIFO trigger levels used in auto-RTS are stored in the UARTi.TCR_REG register. uarti_rts is active if the RX FIFO level is below the HALT trigger level in the UARTi.TCR_REG[3:0] RX_FIFO_TRIG_HALT field. When the receiver FIFO HALT trigger level is reached, uarti_rts is deasserted. The sending device (for example, another UART) might send an additional byte after the trigger level is reached because it might not recognize the deassertion of RTS until it begins sending the additional byte.

uarti_rts is automatically reasserted when the receiver FIFO reaches the RESUME trigger level programmed by the UARTi.TCR_REG[7:4] RX_FIFO_TRIG_START field. This reassertion requests the sending device to resume transmission.

In this case, uarti_rts is an active-low signal.

- **Auto-CTS**

  The transmitter circuitry checks uarti_cts before sending the next data byte. When uarti_cts is active, the transmitter sends the next byte. To stop the transmitter from sending the next byte, uarti_cts must be deasserted before the middle of the last stop bit currently sent.

  The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the uarti_cts state changes do not have to trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO, and a receiver overrun error can result.

  In this case, uarti_cts is an active-low signal.

14.4.4.1.3.3 Software Flow Control

Software flow control is enabled through the enhanced feature register (UARTi.EFR_REG) and the modem control register (UARTi.MCR_REG). Different combinations of software flow control can be enabled by setting different combinations of UARTi.EFR_REG[3:0] (see Table 14-29).

Two other enhanced features relate to software flow control:

- XON any function (UARTi.MCR_REG[5]): Operation resumes after receiving any character after recognition of the XOFF character.

NOTE: The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (UARTi.EFR_REG[5]): Incoming data is compared to XOFF2. When the special character is detected, the XOFF interrupt (UARTi.IIR_REG[4]) is set, but it does not halt transmission. The XOFF interrupt is cleared by a read of UARTi.IIR_REG. The special character is transferred to the RX FIFO.

Table 14-29. EFR_REG[0-3] Software Flow Control Options

Bit 3	Bit 2	Bit 1	Bit 0	Tx, Rx Software Flow Controls
0	0	X	X	No transmit flow control
1	0	X	X	Transmit XON1, XOFF1
0	1	X	X	Transmit XON2, XOFF2
1	1	X	X	Transmit XON1, XON2: XOFF1, XOFF2 (1)
X	X	0	0	No receive flow control
X	X	1	0	Receiver compares XON1, XOFF1
X	X	0	1	Receiver compares XON2, XOFF2
X	X	1	1	Receiver compares XON1, XON2: XOFF1, XOFF2 (1)

(1) In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.
XON1 is defined in the UARTi.XON1_ADDR1_REG[7:0] XON_WORD1 field, XON2 is defined in UARTi.XON2_ADDR2_REG[7:0] XON_WORD2.
XOFF1 is defined in the UARTi.XOFF1_REG[7:0] XOFF_WORD1 field, XOFF2 is defined in UARTi.XOFF2_REG[7:0] XOFF_WORD2.

14.4.4.1.3.3.1 Receive (RX)

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases, XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission stops after transmission of the current character is complete. XOFF detection also sets UARTi.IIR_REG[4] (if enabled by UARTi.IER_REG[5]) and causes the interrupt line to go low.

To resume transmission, an XON1/2 character must be received (in certain cases, XON1 and XON2 must be received sequentially). When the correct XON characters are received, UARTi.IIR_REG[4] is cleared and the XOFF interrupt disappears.

NOTE: When a parity, framing, or break error occurs while receiving a software flow control character, this character is treated as normal data and is written to the RX FIFO.

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, when UARTi.EFR_REG[1:0] = 0x2, if XON1 and XOFF1 characters are received, they do not get written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (UARTi.EFR_REG[1:0] = 0x3), the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

14.4.4.1.3.3.2 Transmit (TX)

XOFF1: Two characters are transmitted when the RX FIFO passes the trigger level programmed by UARTi.TCR_REG[3:0].

XON1: Two characters are transmitted when the RX FIFO reaches the trigger level programmed by UARTi.TCR_REG[7:4].

NOTE: If software flow control is disabled after an XOFF character is sent, the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF(s)/XON(s) follows the same protocol as transmission of an ordinary byte from the TX FIFO. This means that even if the word length is set to 5, 6, or 7 characters, the 5, 6, or 7 LSBs of XOFF1/2 and XON1/2 are transmitted. The 5, 6, or 7 bits of a character are seldom transmitted, but this function is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

14.4.4.1.3.4 Autobauding Modes

In autobauding mode, the UART can extract transfer characteristics (speed, length, and parity) from an AT command (ASCII code). These characteristics are used to receive data after an “at” (AT) and to send data.

The valid AT commands follow:

AT	DATA	<CR>
at	DATA	<CR>
A/		
a/		

A line break during the acquisition of the sequence AT is not recognized, and echo functionality is not implemented in hardware.

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. Either A/ or a/ is used to instruct the software to repeat the last received AT command; therefore, an a/ always follows an AT, and transfer characteristics are not expected to change between an AT and an a/.

When a valid AT is received, AT and all subsequent data, including the final <CR> (0x0D), are saved to RX FIFO. The autobaud state-machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved into RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The UARTi.UASR_REG register reflects the correct settings for the baud rate detected. The interrupt activity can continue in this fashion when a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed: 115.2k baud, 57.6k baud, 38.4k baud, 28.8k baud, 19.2k baud, 14.4k baud, 9.6k baud, 4.8k baud, 2.4k baud, or 1.2k baud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

NOTE: The combination of 7-bit character + space parity is not supported.

Autobauding mode is selected when the UARTi.MDR1_REG[2:0] MODE_SELECT field is set to 0x2. In the UART autobauding mode, UARTi.DLL_REG, UARTi.DLH_REG, UARTi.LCR_REG[5:0] settings are not used; instead, UASR_REG is updated with the configuration detected by the autobauding logic.

UASR_REG Autobauding Status Register Use

This register is used to set up transmission according to the characteristics of the previous reception instead of the UARTi.LCR_REG, UARTi.DLL_REG, and UARTi.DLH_REG registers when the UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), the UARTi.MDR1_REG[2:0] MODE_ SELECT field must be set to reset state (0x7) and then to the UART in autobauding mode (0x2) or to the UART in standard mode (0x0).

Use limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1,200 and 115,200 bps (10 possibilities)

14.4.4.1.3.5 Error Detection

When the UARTi.LSR_REG register is read, UARTi.LSR_REG[4:2] reflects the error bits (BI: break condition, FE: framing error, PE: parity error) of the character at the top of the RX FIFO (next character to be read). Therefore, reading the UARTi.LSR_REG register and then reading the UARTi.RHR_REG register identifies errors in a character.

Reading the UARTi.RHR_REG register updates the BI, FE, and PE bits (see Table 14-30 for the UART mode interrupts).

The UARTi.LSR_REG [7] RX_FIFO_STS bit is set when there is an error in the RX FIFO and is cleared only when no errors remain in the RX FIFO.

NOTE: Reading UARTi.LSR_REG does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading UARTi.RHR_REG.

Reading UARTi.LSR_REG clears the OE bit if it is set (see Table 14-30 for the UART mode interrupts).

14.4.4.1.3.6 Overrun During Receive

Overrun during receive occurs if the RX state-machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the MPU with UARTi.IIR_REG[5:1] IT_TYPE = 0x3 (receiver line status error) and discards the remaining portion of the frame.

Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received, the system (MPU) must:

- Reset the RX FIFO

- Read the UARTi.RESUME_REG register (which clears the internal flag)

14.4.4.1.3.7 Time-Out and Break Conditions

14.4.4.1.3.7.1 Time-Out Counter

An RX idle condition is detected when the receiver line (uarti_rx) is high for a time equivalent to 4x programmed word length + 12 bits. uarti_rx is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on uarti_rx.

For the time-out interrupt, the counter counts only when there is data in the RX FIFO, and the count is reset when there is activity on uarti_rx or when the UARTi.RHR_REG is read.

14.4.4.1.3.7.2 Break Condition

When a break condition occurs, uarti_tx is pulled low. A break condition is activated by setting the UARTi.LCR_REG[6] BREAK_EN bit. The break condition is not aligned on word stream (that is, a break condition can occur in the middle of a character). The only way to send a break condition on a full character is as follows:

1. Reset the transmit FIFO (if enabled).
2. Wait for the transmit shift register to empty (UARTi.LSR_REG[6] TX_SR_E = 1).
3. Take a guard time according to stop-bit definition.
4. Set the BREAK_EN bit to 1.

The break condition is asserted while the BREAK_EN bit is set to 1.

The time-out counter and break condition apply only to UART modem operation and not to IrDA/CIR mode operation.

14.4.4.1.4 UART Mode Interrupt Management

The UART mode includes seven possible interrupts prioritized to six levels.

When an interrupt is generated, the interrupt identification register (UARTi.IIR_REG) sets the UARTi.IIR_REG[0] IT_PENDING bit to 0 to indicate that an interrupt is pending, and provides the type of interrupt through UARTi.IIR_REG[5:1]. Table 14-30 summarizes the interrupt control functions.

Table 14-30. UART Mode Interrupts

IIR_REG[5:0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
000001	None	None	None	None
000110	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE, PE, BI: Read RHR_REG. OE: Read LSR_REG.
001100	2	RX time-out	Stale data in RX FIFO	Read RHR_REG.
000100	2	RHR interrupt	DRDY (data ready) (FIFO disable)	Read RHR_REG until interrupt condition disappears.
			RX FIFO above trigger level (FIFO enable)	
000010	3	THR interrupt	TFE (THR_REG empty) (FIFO disable)	Write to THR_REG until interrupt condition disappears.
			TX FIFO below trigger level (FIFO enable)	
000000	4	Modem status	See the MSR_REG register.	Read MSR_REG.
010000	5	XOFF interrupt/special character interrupt	Receive XOFF characters/special character	Receive XON character(s), if XOFF interrupt/read of IIR_REG, if special character interrupt.
100000	6	CTS, RTS	RTS pin or CTS pin change state from active (low) to inactive (high).	Read IIR_REG.

For the receiver-line status interrupt, the RX_FIFO_STS bit (UARTi.LSR_REG[7]) generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, the interrupt is cleared by an XON flow character detection. If special character detection caused the interrupt, the interrupt is cleared by a read of the UARTi.IIR_REG register.

14.4.4.1.4.1 Wake-Up Interrupt

Wake-up interrupt is a special interrupt that is not designed the same way as other interrupts. This interrupt is enabled when the UARTi.SCR_REG[4] RX_CTS_WU_EN bit is set to 1. The UARTi.IIR_REG register is not modified when it occurs; the UART3.SSR_REG[1] RX_CTS_WU_STS bit must be checked to detect a wake-up event.

When a wake-up interrupt occurs, the only way to clear it is to reset the UARTi.SCR_REG[4] RX_CTS_WU_EN bit. This bit must be re-enabled (set to 1) after the current wake-up interrupt event is processed to detect the next incoming wake-up event.

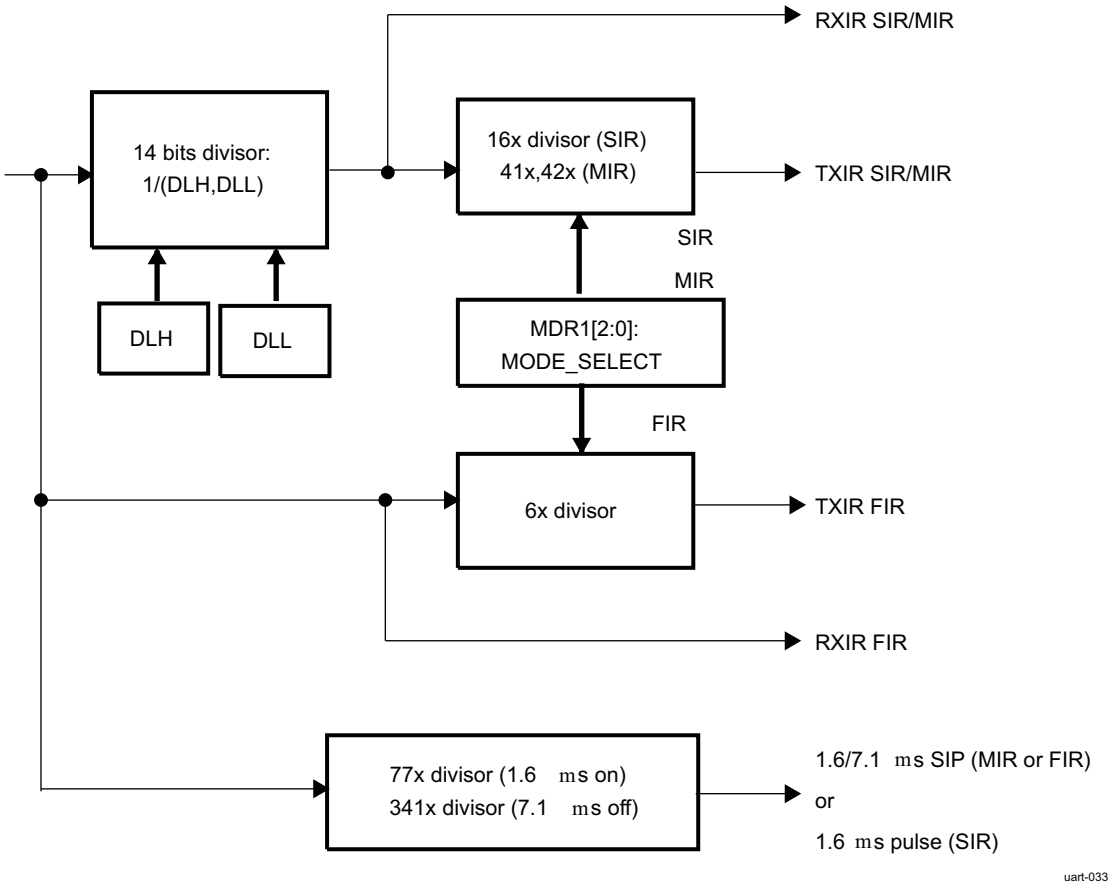
14.4.4.2 IrDA Mode (UART3 Only)

14.4.4.2.1 IrDA Clock Generation: Baud Generator

The IrDA function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 14-32 shows the baud rate generator and associated controls.

Figure 14-32. Baud Rate Generator



CAUTION

Before trying to initialize or modify clock parameter controls (UARTi.DLH_REG, UARTi.DLL_REG), it is mandatory to set MODE_SELECT=DISABLE (UARTi.MDR1_REG[2:0]=0x7).

Failure to observe this rule can result in unpredictable module behavior.

14.4.4.2.2 Choosing the Appropriate Divisor Value

- SIR mode: Divisor value = Operating frequency/(16x baud rate)
- MIR mode: Divisor value = Operating frequency/(41x/42x baud rate)
- FIR mode: Divisor value = None

Table 14-31 lists the IrDA baud rate settings.

Table 14-31. IrDA Baud Rates Settings

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)$	Source Jitter (%)	Pulse duration
2.4 Kbps	SIR	16x	3/16	1250	2.4 Kbps	0	0	78.1 µs
9.6 Kbps	SIR	16x	3/16	312	9.6153 Kbps	+0.16	0	19.5 µs
19.2 Kbps	SIR	16x	3/16	156	19.231 Kbps	+0.16	0	9.75 µs
38.4 Kbps	SIR	16x	3/16	78	38.462 Kbps	+0.16	0	4.87 µs
57.6 Kbps	SIR	16x	3/16	52	57.692 Kbps	+0.16	0	3.25 µs
115.2 Kbps	SIR	16x	3/16	26	115.38 Kbps	+0.16	0	1.62 µs
0.576 Mbps	MIR	41x/42x	1/4	2	0.5756 Mbps(1)	0	+1.63/- 0.80	416 ns
1.152 Mbps	MIR	41x/42x	1/4	1	1.1511 Mbps(1)	0	+1.63/- 0.80	208 ns
4 Mbps	FIR	6x	4 PPM	-	4 Mbps	0	0	125 ns

(1) Average value

NOTE: Baud rate error and source jitter table values do not include 48-MHz reference clock error and jitter.

14.4.4.2.3 IrDA Data Formatting

The methods described in this section apply to all IrDA modes (SIR, MIR, and FIR).

14.4.4.2.3.1 IRRX Polarity Control

The UART3.MDR2_REG[6] IRRXINVERT bit provides the flexibility to invert the uart3_rx_irrx pin in the UART module to ensure that the protocol at the output of the transceiver module has the same polarity at module level. By default, the uart3_rx_irrx pin is inverted because most transceivers invert the IR receive pin.

14.4.4.2.3.2 IrDA Reception Control

Data can be transferred both ways by the module, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

Operation of the uart3_rx_irrx input can be disabled by the UART3.ACREG_REG[5] DIS_IR_RX bit.

14.4.4.2.3.3 IR Address Checking

In all IR modes, when address checking is enabled, only frames intended for the device are written to the RX FIFO. This restriction avoids receiving frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives with the UART3.XON1_ADDR1_REG[7:0] XON_WORD1 and UART3.XON2_ADDR2_REG[7:0] XON_WORD2 fields.

Setting the EFR_REG[0] bit to 1 selects address1 checking. Setting the EFR_REG[1] bit to 1 selects address2 checking. Setting the EFR_REG[1:0] bit to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for both private and public addresses.

If address checking is disabled, all received frames write to the reception FIFO.

14.4.4.2.3.4 Frame Closing

A transmission frame can be correctly terminated in two ways:

- Frame-length method: The frame-length method is selected by setting the UART3.MDR1_REG[7] FRAME_END_MODE bit to 0. The MPU writes the frame-length value to the UART3.TXFLH_REG and UART3.TXFLL_REG registers. The device automatically attaches end flags to the frame when the number of bytes transmitted equals the frame-length value.
- Set-EOT bit method: The set-EOT bit method is selected by setting the FRAME_END_MODE bit to 1. The MPU writes 1 to the UART3.ACREG_REG[0] EOT bit just before it writes the last byte to the TX FIFO. When the MPU writes the last byte to the TX FIFO, the device internally sets the tag bit for that character in the TX FIFO. As the TX state-machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and correctly terminate the frame.

14.4.4.2.3.5 Store and Controlled Transmission

In store and controlled transmission (SCT) mode, the MPU starts writing data to the TX FIFO. Then, after writing a part of a frame (for a bigger frame) or a whole frame (a small frame; that is, a supervisory frame), the MPU writes 1 to the UART3.ACREG_REG[2] SCTX_EN bit (deferred TX start) to start transmission.

SCT mode is enabled by setting the UART3.MDR1_REG[5] SCT bit to 1. This transmission method is different from the normal mode, in which data transmission starts immediately after data is written to the TX FIFO. SCT mode is useful for sending short frames without TX underrun.

14.4.4.2.3.6 Error Detection

When the UART3.LSR_REG register is read, the UART3.LSR_REG[4:2] field reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (the next frame status to be read).

The error is triggered by an interrupt (see Table 14-32 for IrDA mode interrupts). STATUS FIFO must be read until empty (a maximum of eight reads is required).

14.4.4.2.3.7 Underrun During Transmission

Underrun during transmission occurs when the TX FIFO is empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission.

Underrun also causes an internal flag to be set, which disables additional transmissions. Before the next frame can be transmitted, the system (MPU) must:

- Reset the TX FIFO.
- Read the UART3.RESUME_REG register (which clears the internal flag).

This functionality can be disabled by the UART3.ACREG_REG[4] DIS_TX_UNDERRUN bit, compensated by the extension of the stop bit in transmission if the TX FIFO is empty.

14.4.4.2.3.8 Overrun During Receive

Overrun during receive for the IrDA mode has the same functionality as that for the UART mode (see Section 14.4.4.1.3.6, *Overrun During Receive*).

14.4.4.2.3.9 Status FIFO

In IrDA modes, a status FIFO records the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written to the status FIFO.

Reading UART3.SFREGH_REG[3:0] (MSB) and UART3.SFREGL_REG[3:0] (LSB) obtains the frame length. The frame error status is read in the UART3.SFLSR_REG register. Reading the UART3.SFLSR_REG register increments the status FIFO read pointer. The status FIFO is eight entries deep and, therefore, can hold the status of eight frames.

The MPU uses the frame-length information to locate the frame boundary in the received frame data. The MPU can screen bad frames using the error status information and can later request the sender to resend only the bad frames.

This status FIFO can be used effectively in DMA mode because the MPU must be interrupted only when the programmed status FIFO trigger level is reached, not each time a frame is received.

14.4.4.2.4 SIR Mode DATA Formatting

This section provides specific instructions for SIR mode programming.

14.4.4.2.4.1 Abort Sequence

When the transmitter prematurely closes a frame, it sends the following sequence to abort: 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

A transmission frame can be aborted by setting the UART3.ACREG_REG[1] ABORT_EN bit to 1.

When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

The receiver treats a frame as an aborted frame when a 0x7D character followed immediately by a 0xC1 character is received without transparency.

CAUTION

When the transmit FIFO is not empty and the UART3.MDR1_REG[5] SCT bit is set to 1, the UART IrDA starts a new transfer with data of a previous frame when the abort frame is sent. Therefore, TX FIFO must be reset before sending an abort frame.

14.4.4.2.4.2 Pulse Shaping

The SIR mode supports both the 3/16th or the 1.6-µs pulse duration methods. The UART3.ACREG_REG[7] PULSE_TYPE bit selects the pulse width method in the transmit mode.

14.4.4.2.4.3 SIR Free Format Programming

The SIR FF mode is selected by setting the module in the UART mode (UART3.MDR1_REG[2:0] MODE_SELECT = 0x0) and the UART3.MDR2_REG[3] UART_PULSE bit to 1 to allow pulse shaping.

Because the bit format remains the same, some UART mode configuration registers must be set at specific values:

- UART3.LCR_REG[1:0] CHAR_LENGTH field = 0x3 (8 data bits)
- UART3.LCR_REG[2] NB_STOP bit = 0x0 (1 stop-bit)
- UART3.LCR_REG[3] PARITY_EN bit = 0x0 (no parity)

The UART mode interrupts are used for the SIR FF mode, but many of them are not relevant (XOFF, RTS, CTS, modem status register, etc.).

14.4.4.2.5 MIR and FIR Mode Data Formatting

This section describes common instructions for FIR and MIR mode programming.

At the end of a frame reception, the MPU reads the line status register (UART3.LSR_REG) to detect possible errors in the received frame.

When the UART3.MDR1_REG[6] SIP_MODE bit is set to 1, the TX state-machine always sends one SIP at the end of a transmission frame. However, when the SIP_MODE bit is set to 0, SIP transmission depends on the UART3.ACREG_REG[3] SEND_SIP bit.

The system (MPU) can set the SEND_SIP bit at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not have to send the SIP at the end of each frame, which can reduce the overhead required.

14.4.4.2.6 IrDA Mode Interrupt Management

14.4.4.2.6.1 IrDA Interrupts

The IrDA function generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (UART3.IER_REG). The interrupt status of the device can be checked at any time by reading the interrupt identification register (UART3.IIR_REG).

The UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different UART3.IER_REG and UART3.IIR_REG mappings, depending on the selected mode.

The IrDA modes have eight possible interrupts (see Table 14-32). The interrupt line is activated when any of the eight interrupts is generated (there is no priority).

Table 14-32. IrDA Mode Interrupts

IIR_REG Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read RHR_REG until interrupt condition disappears.
1	THR interrupt	TFE (THR_REG empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR_REG until interrupt condition disappears.
2	Last byte in RX FIFO	Last byte of frame in RX FIFO is available to be read at the RHR port.	Read RHR_REG.
3	RX overrun	Write to RHR_REG when RX FIFO full.	Read RESUME_REG register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read STATUS FIFO.
5	TX status	1. THR_REG empty before EOF sent. Last bit of transmission of the IrDA frame occurred, but with an underrun error. OR 2. Transmission of the last bit of the IrDA frame completed successfully.	1. Read RESUME_REG register. OR 2. Read IIR_REG.
6	Receiver line status interrupt	CRC, ABORT, or frame-length error is written into STATUS FIFO.	Read STATUS FIFO (read until empty - maximum of eight reads required).
7	Received EOF	Received end-of-frame.	Read IIR_REG.

14.4.4.2.6.2 Wake-Up Interrupts

The wake-up interrupt for the IrDA mode has the same functionality as that for the UART mode (see Section 14.4.4.1.4.1, *Wake-Up Interrupt*).

CAUTION

Wake-up interface implementation in this mode is based on the UARTi_SIDLEACK low-to-high transition instead of the UARTi_SIDLEACK state.

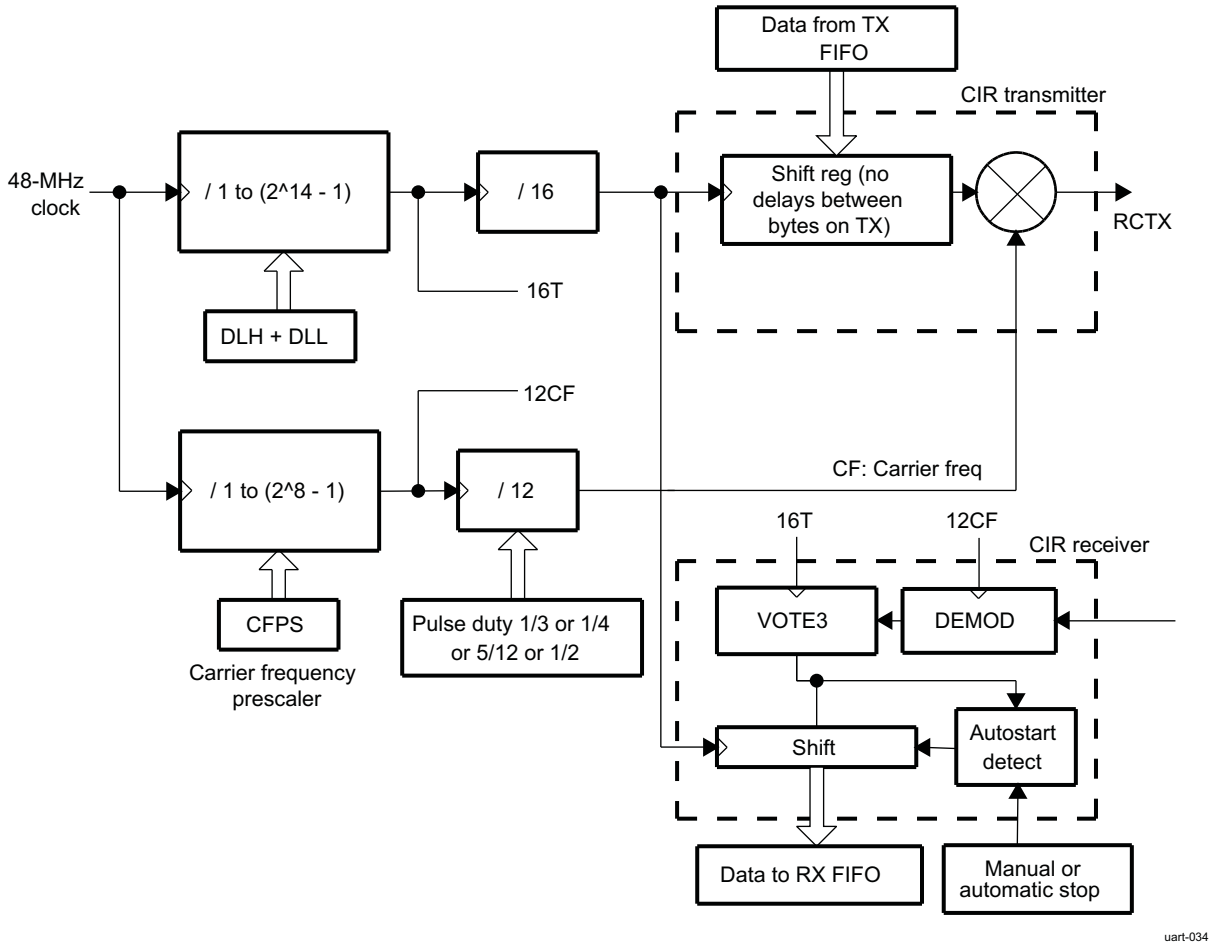
This does not ensure wake-up event generation as expected when configured in smart-idle mode and the system wakes up for a short period.

14.4.4.3 CIR Mode (UART3 Only)

14.4.4.3.1 CIR Mode Clock Generation

Depending on the encoding method (variable pulse distance/biphase), the MPU must develop a data structure that combines 1 and 0 with a t period to encode the complete frame to transmit. This can then be transmitted to the infrared output with a modulation method, as shown in Figure 14-33.

Figure 14-33. CIR Mode Block Components



Based on the requested modulation frequency, the UART3.CFPS_REG register must be set with the correct dividing value to provide the more accurate pulse frequency:

Dividing value = (FCLK/12)/MODfreq

Where:

FCLK = System clock frequency (48 MHz)

12 = Real value of baud multiple

MODfreq = Effective frequency of the modulation (MHz)

Example: For a targeted modulation frequency of 36 kHz, the CFPS_REG value must be set to 111 (decimal), which provides a modulation frequency of 36.04 kHz.

NOTE: The UART3.CFPS_REG register starts with a reset value of 105 (decimal), which translates to a frequency of 38.1 kHz.

The duty cycle of these pulses is user-defined by the pulse duty register bits in the UART3.MDR2_REG configuration register. Table 14-33 shows the duty cycle.

Table 14-33. Duty Cycle

MDR2_REG[5:4]	Duty Cycle (High Level)
00	1/4
01	1/3
10	5/12
11	½

14.4.4.3.2 CIR Data Formatting

The methods described in this section apply to all CIR modes.

14.4.4.3.2.1 IRRX Polarity Control

The IRRX polarity control for the CIR mode has the same functionality as that for the IrDA mode (see Section 14.4.4.2.3.1, *IRRX Polarity Control*).

14.4.4.3.2.2 CIR Transmission

In transmission, the MPU software must exercise an element of real-time control to transmit data packets, each of which must be emitted at a constant delay from the start bits of each individual packet. Thus, when sending a series of packets, the packet-to-packet delay must respect a specific delay. Two methods can be used to control this delay:

- Filling the TX FIFO with a number of zero bits that are transmitted with a *t* period
- Using an external system timer to control the delay either between each start-of-frame or between the end of a frame and the start of the next one. This can be performed by:
  - Controlling the start of the frame using the UART3.MDR1_REG[5] SCT bit and UART3.ACREG_REG[2] SCTX_EN bit, depending on the timer status
  - Using the UART3.IIR_REG[5] TX_STATUS_IT interrupt to preload the next frame in the TX FIFO and to control the start of the timer (in case of control delay between the end of a frame and the start of the next frame).

NOTE: There's a limitation when receiving data in UART CIR mode. The consumer IR transceivers on the market have a common characteristic that shrinks the hold time of the received modulation pulse. The UART filtering schema on receiving is based on the same encoding mechanism used in transmission.

Consider the following scenario:

- shift register period : 0.9 µs
- modulation frequency : 36 kHz
- duty cycle : 1/4 of a modulation frequency period

So the data sent in those conditions would look like 7µs pulses within 28µs period. The UART expects to receive similar incoming data on receive, but available transceiver timing characteristics typically send 2µs modulated pulses. Those will be filtered out and RX FIFO will not receive any data.

The transmission of data in UART CIR mode is not affected by this limitation.

14.4.4.3.3 CIR Mode Interrupt Management

14.4.4.3.3.1 CIR Interrupts

The CIR function generates interrupts that can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (UART3.IER_REG). The interrupt status of the device can be checked at any time by reading the interrupt identification register (UART3.IIR_REG).

The UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different UART3.IER_REG and UART3.IIR_REG mappings, depending on the selected mode.

Table 14-34 lists the interrupt modes to be maintained. In CIR mode, the sole purpose of the UART3.IIR_REG[5] bit is to indicate that the last bit of infrared data was passed to the uart3_cts_rctx pin.

Table 14-34. CIR Mode Interrupts

IIR_REG Bit Number	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
1	THR interrupt	TFE (THR_REG empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to THR_REG until interrupt condition disappears.
2:4	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
5	TX status	Transmission of the last bit of the frame is completed successfully.	Read IIR_REG.
6:7	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode

14.4.4.3.3.2 Wake-Up Interrupts

The wake-up interrupt for the IrDA mode has the same functionality as that for the UART mode (see Section 14.4.4.1.4.1, *Wake-Up Interrupt*).

14.4.5 Power Management

14.4.5.1 UART Mode Power Management

14.4.5.1.1 Module Power Saving

In UART modes, sleep mode is enabled by setting the UARTi.IER_REG[4] SLEEP_MODE bit to 1 (when the UARTi.EFR_REG[4] ENHANCED_EN bit is set to 1).

Sleep mode is entered when all the following conditions exist:

- The serial data input line, uarti_rx, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- No interrupts are pending except THR interrupts.

Sleep mode is a good way to lower power consumption of the UART, but this state can be achieved only when the UART is set in modem mode. Therefore, even if the UART has no functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked using these clocks, this greatly reduces power consumption. The module wakes up when a change is detected on the uarti_rx line, when data is written to the TX FIFO, and when there is a change in the state of the modem input pins.

An interrupt can be generated on a wake-up event by setting the UARTi.SCR_REG[4] RX_CTS_WU_EN bit to 1. See Section 14.4.4.1.4.1, *Wake-Up Interrupt*, to understand how to manage the interrupt.

NOTE: There must be no writing to the divisor latches, UARTi.DLL_REG and UARTi.DLH_REG, to set the baud clock, BCLK, while in sleep mode. It is advisable to disable sleep mode using the UARTi.IER_REG[4] SLEEP_MODE bit before writing to the UARTi.DLL_REG register or the UARTi.DLH_REG register.

14.4.5.1.2 System Power Saving

Sleep and auto-idle modes are embedded power-saving features. At the system level, power-reduction techniques can be applied by shutting down certain internal clock and power domains of the device.

The UART supports an idle req/idle ack handshaking protocol. This protocol is used at the system level to shut down clocks of the UART in a clean and controlled manner and to switch the UART from interrupt-generation mode to wake-up generation mode for unmasked events (see the UARTi.SYSC_REG[2] ENAWAKEUP bit and the UARTi.WER_REG register).

For more information, see the *Power, Reset, and Clock Management* chapter.

14.4.5.2 IrDA Mode Power Management (UART3 Only)

14.4.5.2.1 Module Power Saving

In IrDA modes, sleep mode is enabled by setting the UART3.MDR[3] IR_SLEEP bit to 1.

Sleep mode is entered when all the following conditions exist:

- The serial data input line, uart3.rx_irrx, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- No interrupts are pending except THR interrupts.

The module wakes up when a change is detected on the uart3_rx_irrx line or when data is written to the TX FIFO.

14.4.5.2.2 System Power Saving

System power saving for the IrDA mode has the same functionality as that for the UART mode (see Section 14.4.5.1.2, *System Power Saving*).

14.4.5.3 CIR Mode Power Management (UART3 Only)

14.4.5.3.1 Module Power Saving

Module power saving for the CIR mode has the same functionality as that for the IrDA mode (see Section 14.4.5.2.1, *Module Power Saving*).

14.4.5.3.2 System Power Saving

System power saving for the CIR mode has the same functionality as that for the UART mode (see Section 14.4.5.2.2, *System Power Saving*).

14.5 UART/IrDA/CIR Basic Programming Model

14.5.1 UART Programming Model

14.5.1.1 Quick Start

This section outlines the procedure for operating the UART module with FIFO and DMA or interrupts. This 3-part procedure ensures the quick start of the UART module. It does not cover every UART module feature.

The first programming model covers software reset of the module. The second programming model deals with FIFO and DMA configuration. The last programming model deals with protocol, baud rate and interrupt configuration.

NOTE: Each programming model can be used independently of the other two; for instance, reconfiguring the FIFOs and DMA settings only.

Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information see Section 14.4.3.1, *Register Access Modes*.

14.5.1.1.1 Software Reset

To clear the UART registers, perform the following steps:

1. Initiate a software reset:
   Set the UARTi.SYSC_REG[1] SOFTRESET bit to 1.
2. Wait for the end of the reset operation:
   Poll the UARTi.SYSS_REG[0] RESETDONE bit until it equals 1.

14.5.1.1.2 FIFOs and DMA Settings

To enable and configure the receive and transmit FIFOs and program the DMA mode, perform the following steps:

1. Switch to register configuration mode B to access the UARTi.EFR_REG register:
   (a) Save the current UARTi.LCR_REG value.
   (b) Set UARTi.LCR_REG to 0x00BF.
2. Enable register submode TCR_TLR to access UARTi.TLR_REG (part 1 of 2):
   (a) Save the UARTi.EFR_REG[4] ENHANCED_EN value.
   (b) Set the UARTi.EFR_REG[4] ENHANCED_EN bit to 1.
3. Switch to register configuration mode A to access the UARTi.MCR_REG register:
   Set UARTi.LCR_REG to 0x0080.
4. Enable register submode TCR_TLR to access UARTi.TLR_REG (part 2 of 2):
   (a) Save the UARTi.MCR_REG[6] TCR_TLR value.
   (b) Set the UARTi.MCR_REG[6] TCR_TLR bit to 1.
5. Enable FIFO, load the new FIFO triggers (part 1 of 3) and the new DMA mode (part 1 of 2):
   Set the following bits to the desired values:
   - UARTi.FCR_REG[7:6] RX_FIFO_TRIG
   - UARTi.FCR_REG[5:4] TX_FIFO_TRIG
   - UARTi.FCR_REG[3] DMA_MODE
   - UARTi.FCR_REG[0] FIFO_ENABLE (0: Disable the FIFO/1: Enable the FIFO)

NOTE: The UARTi.FCR_REG register is not readable.

6. Switch to register configuration mode B to access the UARTi.EFR_REG register:
   Set UARTi.LCR_REG to 0x00BF.
7. Load the new FIFO triggers (part 2 of 3):
   Set the following bits to the desired values:
   - UARTi.TLR_REG[7:4] RX_FIFO_TRIG_DMA
   - UARTi.TLR_REG[3:0] TX_FIFO_TRIG_DMA
8. Load the new FIFO triggers (part 3 of 3) and the new DMA mode (part 2 of 2):
   Set the following bits to the desired values:
   - UARTi.SCR_REG[7] RX_TRIG_GRANU1
   - UARTi.SCR_REG[6] TX_TRIG_GRANU1
   - UARTi.SCR_REG[2:1] DMA_MODE_2
   - UARTi.SCR_REG[0] DMA_MODE_CTL
9. Restore the UARTi.EFR_REG[4] ENHANCED_EN value saved in Step 2a.
10. Switch to register configuration mode A to access the UARTi.MCR_REG register:
    Set UARTi.LCR_REG to 0x0080.
11. Restore the UARTi.MCR_REG[6] TCR_TLR value saved in Step 4a.
12. Restore the UARTi.LCR_REG value saved in Step 1a.

Triggers are used to generate interrupt and DMA requests. See Section 14.4.2.1.1, *Transmit FIFO Trigger,* to choose the following values:

- UARTi.FCR_REG[5:4] TX_FIFO_TRIG
- UARTi.TLR_REG[3:0] TX_FIFO_TRIG_DMA
- UARTi.SCR_REG[6] TX_TRIG_GRANU1

Triggers are used to generate interrupt and DMA requests. See Section 14.4.2.1.2, *Receive FIFO Trigger*, to choose the following values:

- UARTi.FCR_REG[7:6] RX_FIFO_TRIG
- UARTi.TLR_REG[7:4] RX_FIFO_TRIG_DMA
- UARTi.SCR_REG[7] RX_TRIG_GRANU1

DMA mode enables the different DMA requests. See Section 14.4.2.4, *FIFO DMA Mode Operation*, to choose the following values:

- UARTi.FCR_REG[3] DMA_MODE
- UARTi.SCR_REG[2:1] DMA_MODE_2
- UARTi.SCR_REG[0] DMA_MODE_CTL

14.5.1.1.3 Protocol, Baud Rate, and Interrupt Settings

To program the protocol, baud rate and interrupt settings, perform the following steps:

1. Disable UART to access UARTi.DLL_REG and UARTi.DLH_REG:
   Set UARTi.MDR1_REG[2:0] MODE_SELECT to 0x7.
2. Switch to register configuration mode B to access the UARTi.EFR_REG register:
   Set UARTi.LCR_REG to 0x00BF.
3. Enable access to UARTi.IER_REG[7:4]:
   (a) Save the UARTi.EFR_REG[4] ENHANCED_EN value.
   (b) Set the UARTi.EFR_REG[4] ENHANCED_EN bit to 1.
4. Switch to register operational mode to access the UARTi.IER_REG register:
   Set UARTi.LCR_REG to 0x0000.

5. Clear the UARTi.IER_REG (UARTi.IER_REG[4] SLEEP_MODE bit to 0 to change UARTi.DLL_REG and UARTi.DLH_REG).Set UARTi.IER_REG to 0x0000.
6. Switch to register configuration mode B to access the UARTi.DLL_REG and UARTi.DLH_REG registers:

   Set UARTi.LCR_REG to 0x00BF.
7. Load the new divisor value:

   Set the UARTi.DLL_REG[7:0] CLOCK_LSB and UARTi.DLH_REG[5:0] CLOCK_MSB fields to the desired value.
8. Switch to register operational mode to access the UARTi.IER_REG register:

   Set UARTi.LCR_REG to 0x0000.
9. Load the new interrupt configuration.(0: Disable the interrupt/1: Enable the interrupt):

   Set the following bits to the desired values:
   - UARTi.IER_REG[7] CTS_IT
   - UARTi.IER_REG[6] RTS_IT
   - UARTi.IER_REG[5] XOFF_IT
   - UARTi.IER_REG[4] SLEEP_MODE
   - UARTi.IER_REG[3] MODEM_STS_IT
   - UARTi.IER_REG[2] LINE_STS_IT
   - UARTi.IER_REG[1] THR_IT
   - UARTi.IER_REG[0] RHR_IT
10. Switch to register configuration mode B to access the UARTi.EFR_REG register:

    Set UARTi.LCR_REG to 0x00BF.
11. Restore the UARTi.EFR_REG[4] ENHANCED_EN value saved in Step 3a.
12. Load the new protocol formatting (parity, stop bit, char length) and switch to register operational mode:

    Set UARTi.LCR_REG[7] DIV_EN to 0.

    Set UARTi.LCR_REG[6] BREAK_EN to 0.

    Set the following bits to the desired values:
    - UARTi.LCR_REG[5] PARITY_TYPE_2
    - UARTi.LCR_REG[4] PARITY_TYPE_1
    - UARTi.LCR_REG[3] PARITY_EN
    - UARTi.LCR_REG[2] NB_STOP
    - UARTi.LCR_REG[1:0] CHAR_LENGTH
13. Load the new UART mode:

    Set UARTi.MDR1_REG[2:0] MODE_SELECT to the desired value.

See Section 14.4.4.1.2, *Choosing the Appropriate Divisor Value*, to choose the following values:

- UARTi.DLL_REG[7:0] CLOCK_LSB
- UARTi.DLH_REG[5:0] CLOCK_MSB
- UARTi.MDR1_REG[2:0] MODE_SELECT

See Section 14.4.4.1.3.1, *Frame Formatting*, to choose the following values:

- UARTi.LCR_REG[5] PARITY_TYPE_2
- UARTi.LCR_REG[4] PARITY_TYPE_1
- UARTi.LCR_REG[3] PARITY_EN
- UARTi.LCR_REG[2] NB_STOP
- UARTi.LCR_REG[1:0] CHAR_LENGTH

14.5.1.2 Hardware and Software Flow Control Configuration

This section outlines the programming steps to enable and configure hardware and software flow control. Hardware and software flow control cannot be used at the same time.

NOTE: Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see Section 14.4.3.1, *Register Access Modes*.

14.5.1.2.1 Hardware Flow Control Configuration

To enable and configure hardware flow control, perform the following procedure:

1. Switch to register configuration mode A to access the UARTi.MCR_REG register:
   (a) Save the current UARTi.LCR_REG.
   (b) Set UARTi.LCR_REG to 0x0080.
2. Enable register submode TCR_TLR to access UARTi.TCR_REG (part 1 of 2):
   (a) Save the UARTi.MCR_REG[6] TCR_TLR value.
   (b) Set UARTi.MCR_REG[6] TCR_TLR = 1.
3. Switch to register configuration mode B to access the UARTi.EFR_REG register:
   Set UARTi.LCR_REG to 0x00BF.
4. Enable register submode TCR_TLR to access the UARTi.TCR_REG register (part 2 of 2):
   (a) Save the UARTi.EFR_REG[4] ENHANCED_EN value.
   (b) Set the UARTi.EFR_REG[4] ENHANCED_EN bit to 1.
5. Load the new start and halt trigger values for hardware flow control:
   Set the following bits to the desired values:
   - UARTi.TCR_REG[7:4] AUTO_RTS_START
   - UARTi.TCR_REG[3:0] AUTO_RTS_HALT
6. Enable or disable receive and transmit hardware flow control mode and restore the UARTi.EFR_REG[4] ENHANCED_EN value saved in Step 4a.
   Set the following bits to the desired values:
   - UARTi.EFR_REG[7] AUTO_CTS_EN (0: Disable/1: Enable)
   - UARTi.EFR_REG[6] AUTO_RTS_EN (0: Disable/1: Enable)

   Restore UARTi.EFR_REG[4] ENHANCED_EN to the saved value.
7. Switch to register configuration mode A to access UARTi.MCR_REG:
   Set UARTi.LCR_REG to 0x0080.
8. Restore the UARTi.MCR_REG[6] TCR_TLR value saved in Step 2a.
9. Restore the UARTi.LCR_REG value saved in Step 1a.

See Section 14.4.4.1.3.2, *Hardware Flow Control*, to choose the following values:

- UARTi.EFR_REG[7] AUTO_CTS_EN
- UARTi.EFR_REG[6] AUTO_RTS_EN
- UARTi.TCR_REG[7:4] AUTO_RTS_START
- UARTi.TCR_REG[3:0] AUTO_RTS_HALT

14.5.1.2.2 Software Flow Control Configuration

To enable and configure software flow control, perform the following procedure:

1. Switch to register configuration mode B to access the UARTi.EFR_REG register.

(a) Save the current UARTi.LCR_REG.

(b) Set UARTi.LCR_REG to 0x00BF.

2. Enable register submode XOFF to access the UARTi.XOFF1_REG and UARTi.XOFF2_REG registers:

   (a) Save the UARTi.EFR_REG[4] ENHANCED_EN value.

   (b) Set the UARTi.EFR_REG[4] ENHANCED_EN bit to 0.

3. Load the new software flow control characters:

   Set the following bits to the desired values:

   - UARTi.XON1_ADDR1_REG[7:0] XON_WORD1
   - UARTi.XON2_ADDR2_REG[7:0] XON_WORD2
   - UARTi.XOFF1_REG[7:0] XOFF_WORD1
   - UARTi.XOFF2_REG[7:0] XOFF_WORD2

4. Enable access to UARTi.MCR_REG[7:5] and enable register submode TCR_TLR to access UARTi.TCR_REG (part 1 of 2):

   Set the UARTi.EFR_REG[4] ENHANCED_EN bit to 1.

5. Switch to register configuration mode A to access the UARTi.MCR_REG register:

   Set UARTi.LCR_REG to 0x0080.

6. Enable register submode TCR_TLR to access UARTi.TCR_REG (part 2 of 2) and enable or disable XON any function:

   (a) Save the UARTi.MCR_REG[6] TCR_TLR value.

   (b) Set the UARTi.MCR_REG[6] TCR_TLR to 1.

   Set UARTi.MCR_REG[5] XON_EN to the desired value (0: Disable/1: Enable).

7. Switch to register configuration mode B to access the UARTi.EFR_REG register:

   Set UARTi.LCR_REG to 0x00BF.

8. Load the new start and halt trigger values for software flow control:

   Set the following bits to the desired values:

   - UARTi.TCR_REG[7:4] AUTO_RTS_START
   - UARTi.TCR_REG[3:0] AUTO_RTS_HALT

9. Enable or disable special char function and load the new software flow control mode and restore the UARTi.EFR_REG[4] ENHANCED_EN value saved in Step 2a:

   Set the following bits to the desired values:

   - UARTi.EFR_REG[5] SPEC_CHAR (0: Disable/1: Enable)
   - UARTi.EFR_REG[3:0] SW_FLOW_CONTROL

   Restore UARTi.EFR_REG[4] ENHANCED_EN to the saved value.

10. Switch to register configuration mode A to access the UARTi.MCR_REG register:

    Set UARTi.LCR_REG to 0x0080.

11. Restore the UARTi.MCR_REG[6] TCR_TLR value saved in Step 6a.

12. Restore the UARTi.LCR_REG value saved in Step 1a.

See Section 14.4.4.1.3.3, *Software Flow Control*, to choose the following values:

- UARTi.EFR_REG[5] SPEC_CHAR
- UARTi.EFR_REG[3:0] SW_FLOW_CONTROL
- UARTi.TCR_REG[7:4] AUTO_RTS_START
- UARTi.TCR_REG[3:0] AUTO_RTS_HALT
- UARTi.XON1_ADDR1_REG[7:0] XON_WORD1
- UARTi.XON2_ADDR2_REG[7:0] XON_WORD2
- UARTi.XOFF1_REG[7:0] XOFF_WORD1
- UARTi.XOFF2_REG[7:0] XOFF_WORD2

14.5.2 IrDA Programming Model (UART3 Only)

14.5.2.1 SIR Mode

14.5.2.1.1 Receive

The following programming model explains how to program the module in order to receive IrDA frame with parity forced to '1', baud rate = 112.5Kbs, FIFOs disable, 2 stop bits, 8 bits word length

1. **Disable UART before accessing UARTi.DLL_REG and UARTi.DLH_REG:**
   Set UARTi.MDR1_REG[2:0] MODE_SELECT to 0x7.
2. **Grant access to DLL_REG and DLH_REG (LCR_REG[7] DIV_EN = 0x1)**
   UART3.LCR_REG = 0x80 (Note: Data format is unaffected by the use and settings of the UART3.LCR_REG register in IrDA mode)
3. **Load the new baud rate (115.2Kbs)**
   UART3.DLL_REG = 0x1A
   UART3.DLH_REG = 0x00
4. **Set SIR Mode**
   UART3.MDR1_REG[2:0] MODE_SELECT = 0x1
5. **Disable access to DLL_REG and DLH_REG and Switch to register operational mode**
   UART3.LCR_REG = 0x00
6. **Optional: Enable RHR interrupt**
   UART3.IER_REG[0] RHR_IT = 0x1

14.5.2.1.2 Transmit

The following programming model explains how to program the module in order to transmit IrDA 6 bytes frame with no parity, baud rate = 112.5Kbs, FIFOs disable, 3/16 encoding, 2 stop bits, 7 bits word length

1. **Disable UART before accessing UARTi.DLL_REG and UARTi.DLH_REG:**
   Set UARTi.MDR1_REG[2:0] MODE_SELECT to 0x7.
2. **Grant access to EFR_REG**
   UART3.LCR_REG = 0xBF
3. **Enable the enhanced features (EFR_REG[4] ENAHNCED_EN = 0x1)**
   UART3.EFR_REG = 0x10
4. **Grant access to DLL_REG and DLH_REG (LCR_REG[7] DIV_EN = 0x1)**
   UART3.LCR_REG = 0x80 (Note: Data format is unaffected by the use and settings of the UART3.LCR_REG register in IrDA mode)
5. **Load the new baud rate (115.2Kbs)**
   UART3.DLL_REG = 0x1A
   UART3.DLH_REG = 0x00
6. **Set SIR Mode (MDR1_REG[2:0] MODE_SELECT = 0x1)**
   UART3.MDR1_REG = 0x01
7. **Disable access to DLL_REG and DLH_REG and Switch to register operational mode**
   UART3.LCR_REG = 0x00
8. **Force DTR output to active**
   UART3.MCR_REG[0] DTR = 0x1
9. **Optional: Enable THR interrupt**
   UART3.IER_REG[1] THR_IT = 0x1
10. **Set transmit frame length to 6 bytes**
    UART3.TXFLL_REG = 0x06
11. **Set 7 starts of frame transmission**
    UART3.EBLR_REG = 0x08
12. **Optional: Set SIR pulse width to be 1.6 us**
    UART3.ACREG_REG[7] PULSE_TYPE = 0x1
13. **Load THR_REG with the desired data to be transmitted**

14.5.2.2 MIR Mode

14.5.2.2.1 Receive

The following programming model explains how to program the module in order to receive IrDA frame with no parity, baud rate = 1.152Mbs, FIFOs disable

1. **Disable UART before accessing UARTi.DLL_REG and UARTi.DLH_REG:**
   Set UARTi.MDR1_REG[2:0] MODE_SELECT to 0x7.
2. **Grant access to DLL_REG and DLH_REG (LCR_REG[7] DIV_EN = 0x1)**
   UART3.LCR_REG = 0x80 (Note: Data format is unaffected by the use and settings of the UART3.LCR_REG register in IrDA mode)
3. **Load the new baud rate (1.152Mbs)**
   UART3.DLL_REG = 0x01
   UART3.DLH_REG = 0x00
4. **Set MIR Mode**
   UART3.MDR1_REG[2:0] MODE_SELECT = 0x4
5. **Disable access to DLL_REG and DLH_REG and Switch to register operational mode**
   UART3.LCR_REG = 0x00
6. **Force DTR output to active (MCR_REG[0] DTR = 0x1)**
   Force RTS output to active (MCR_REG[1] RTS = 0x1)
   UART3.MCR_REG = 0x3
7. **Optional: Enable RHR interrupt**
   UART3.IER_REG[0] RHR_IT = 0x1

14.5.2.2.2 Transmit

The following programming model explains how to program the module in order to transmit IrDA 60 bytes frame with no parity, baud rate = 1.152Mbs, FIFOs disable

1. **Disable UART before accessing UARTi.DLL_REG and UARTi.DLH_REG:**
   Set UARTi.MDR1_REG[2:0] MODE_SELECT to 0x7.
2. **Grant access to DLL_REG and DLH_REG (LCR_REG[7] DIV_EN = 0x1)**
   UART3.LCR_REG = 0x80 (Note: Data format is unaffected by the use and settings of the UART3.LCR_REG register in IrDA mode)
3. **Load the new baud rate (1.152Mbs)**
   UART3.DLL_REG = 0x01
   UART3.DLH_REG = 0x00
4. **Set MIR Mode**
   UART3.MDR1_REG[2:0] MODE_SELECT = 0x4
5. **Disable access to DLL_REG and DLH_REG and Switch to register operational mode**
   UART3.LCR_REG = 0x00
6. **Force DTR output to active**
   UART3.MCR_REG[0] DTR = 0x1
7. **Optional: Enable THR interrupt**
   UART3.IER_REG[1] THR_IT = 0x1
8. **Set frame length to 60 bytes**
   UART3.TXFLL_REG = 0x3C
9. **Optional: Transmit 8 additional starts of frame (MIR mode requires 2 starts anyway)**
   UART3.EBLR_REG = 0x08
10. **SIP will be send at the end of transmission**
    UART3.ACREG_REG[3] = 0x1
11. **Load THR_REG with the desired data to be transmitted**

14.5.2.3 FIR Mode

14.5.2.3.1 Receive

The following programming model explains how to program the module to receive IrDA frame with no parity, baud rate = 4M bits/s, FIFOs enable, 8-bit word length.

1. **Disable UART before accessing UARTi.DLL_REG and UARTi.DLH_REG:**
   Set UARTi.MDR1_REG[2:0] MODE_SELECT to 0x7.
2. **Grant access to DLL_REG and DLH_REG (LCR_REG[7] DIV_EN = 0x1**
   UART3.LCR_REG = 0x80 (Note: Data format is unaffected by the use and settings of the UART3.LCR_REG register in IrDA mode.)
3. **FIFO clear and enable**
   FCR_REG = 0x7 (Tx/Rx FIFO trigger : FCR_REG[7:6] & FCR_REG[5:4])
   LCR_REG[7]=0
4. **Set FIR Mode**
   UART3.MDR1_REG[2:0] MODE_SELECT = 0x5
5. **Set frame length**
   RXFLL_REG = 0xA (Data + CRC + STO)
6. **Disable access to DLL_REG and DLH_REG and switch to register operational mode**
   UART3.LCR_REG[7] DIV_EN = 0x0.
7. **Optional: Enable RHR interrupt**
   UART3.IER_REG[0] RHR_IT = 0x1.

14.5.2.3.2 Transmit

The following programming model explains how to program the module to transmit IrDA 4 bytes frame with no parity, baud rate = 4M bits/s, FIFOs enable, 8-bit word length.

1. **Disable UART before accessing UARTi.DLL_REG and UARTi.DLH_REG:**
   Set MDR1_REG[2:0] MODE_SELECT = 0x7).
2. **Grant access to EFR_REG**
   UART3.LCR_REG = 0xBF
3. **Enable the enhanced features (EFR_REG[4] ENAHNCED_EN = 0x1)**
   UART3.EFR_REG = 0x10.
4. **FIFO clear and enable**
   FCR_REG = 0x7 (Tx/Rx FIFO trigger: FCR_REG[7:6] and FCR_REG[5:4])
   LCR_REG[7]=0
5. **Set FIR mode and enable auto-SIP mode**
   MDR1_REG = 0x45
6. **Set Frame Length**
   TXFLL_REG = 0x4
   TXFLH_REG = 0x0
   RXFLL_REG = 0xA (Data + CRC + STO)
   RXFLH_REG = 0x0
7. **Force DTR output to active**
   UART3.MCR_REG[0] DTR = 0x1
8. **Optional: Enable THR interrupt**
   UART3.IER_REG[1] THR_IT = 0x1
9. **Load THR_REG with the desired data to be transmitted**

14.6 UART/IrDA/CIR Registers

Table 14-35 shows the base address and address space for the UART/IrDA/CIR module instances.

Table 14-35. Instance Summary

Module Name	Base Address	Size
UART1 (UART only)	0x4806 A000	1K byte
UART2 (UART only)	0x4806 C000	1K byte
UART3 (UART/IrDA/CIR)	0x4902 0000	1K byte
UART4	0x4809 E000	1K byte

14.6.1 UART/IrDA/CIR Register Mapping Summary

This section provides information about the UART/IrDA/CIR module instance in the device. Each register in the module instance is described separately in this section. This module has multiple modes. Table 14-36 summarizes register access for different modes.

Table 14-36. UART Mode Overview

Address Offset	Registers					
	Configuration_Mode_A		Configuration_Mode_B		Operational_Mode	
	Read	Write	Read	Write	Read	Write
0x000	DLL_REG	DLL_REG	DLL_REG	DLL_REG	RHR_REG	THR_REG
0x004	DLH_REG	DLH_REG	DLH_REG	DLH_REG	IER_REG	IER_REG
0x008	IIR_REG	FCR_REG	EFR_REG	EFR_REG	IIR_REG	FCR_REG
0x00C	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG	LCR_REG
0x010	MCR_REG	MCR_REG	XON1_ADDR1_REG	XON1_ADDR1_REG	MCR_REG	MCR_REG
0x014	LSR_REG	-	XON2_ADDR2_REG	XON2_ADDR2_REG	LSR_REG	-
0x018	MSR_REG(1)/TCR_REG(2)	TCR_REG(2)	TCR_REG(2)/XOFF1_REG(3)	TCR_REG(2)/XOFF1_REG(3)	MSR_REG(1)/TCR_REG(2)	TCR_REG(2)
0x01C	SPR_REG(1)/TLR_REG(2)	SPR_REG(1)/TLR_REG(2)	TLR_REG(2)/XOFF2_REG(3)	TLR_REG(2)/XOFF2_REG(3)	SPR_REG(1)/TLR_REG(2)	SPR_REG(1)/TLR_REG(2)
0x020	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG	MDR1_REG
0x024	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG	MDR2_REG
0x028	SFLSR_REG	TXFLL_REG	SFLSR_REG	TXFLL_REG	SFLSR_REG	TXFLL_REG
0x02C	RESUME_REG	TXFLH_REG	RESUME_REG	TXFLH_REG	RESUME_REG	TXFLH_REG
0x030	SFREGL_REG	RXFLL_REG	SFREGL_REG	RXFLL_REG	SFREGL_REG	RXFLL_REG
0x034	SFREGH_REG	RXFLH_REG	SFREGH_REG	RXFLH_REG	SFREGH_REG	RXFLH_REG
0x038	UASR_REG	-	UASR_REG	-	BLR_REG	BLR_REG
0x03C	-	-	-	-	ACREG_REG	ACREG_REG
0x040	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG	SCR_REG
0x044	SSR_REG	-	SSR_REG	-	SSR_REG	-
0x048	-	-	-	-	EBLR_REG	EBLR_REG
0x050	-	-	-	-	-	-
0x054	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG	SYSC_REG
0x058	SYSS_REG	-	SYSS_REG	-	SYSS_REG	-
0x05C	WER_REG	WER_REG	WER_REG	WER_REG	WER_REG	WER_REG
0x060	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG	CFPS_REG

(1) if MSR_SPR mode is active
(2) if TCR_TLR mode is active
(3) if XOFF mode is active

Table 14-37. UART1/2/3/4 Mode Summary

Mode	Condition
Configuration_Mode_A	LCR_REG[7] = 0x1 and LCR_REG[7:0]! = 0xBF
Configuration_Mode_B	LCR_REG[7] = 0x1 and LCR_REG[7:0] = 0xBF
Operational_Mode	LCR_REG[7] = 0x0

Table 14-38. UART1/2/3/4 Register Summary for Configuration_Mode_A Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
DLL_REG	RW	8	0x0000 0000	Section 14.6.2.1
DLH_REG	RW	8	0x0000 0004	Section 14.6.2.5
IIR_REG	R	8	0x0000 0008	Section 14.6.2.7
FCR_REG	W	8	0x0000 0008	Section 14.6.2.6
LCR_REG	RW	8	0x0000 000C	Section 14.6.2.9
MCR_REG	RW	8	0x0000 0010	Section 14.6.2.10
LSR_REG	R	8	0x0000 0014	Section 14.6.2.12
See Table 14-39 and either Table 14-40 or Table 14-41.			0x0000 0018	
See Table 14-39 and either Table 14-40 or Table 14-41.			0x0000 001C	
MDR1_REG	RW	8	0x0000 0020	Section 14.6.2.20
MDR2_REG	RW	8	0x0000 0024	Section 14.6.2.21
SFLSR_REG	R	8	0x0000 0028	Section 14.6.2.23
TXFLL_REG	W	8	0x0000 0028	Section 14.6.2.22
RESUME_REG	R	8	0x0000 002C	Section 14.6.2.24
TXFLH_REG	W	8	0x0000 002C	Section 14.6.2.25
RXFLL_REG	W	8	0x0000 0030	Section 14.6.2.26
SFREGL_REG	R	8	0x0000 0030	Section 14.6.2.27
RXFLH_REG	W	8	0x0000 0034	Section 14.6.2.29
SFREGH_REG	R	8	0x0000 0034	Section 14.6.2.28
UASR_REG	R	8	0x0000 0038	Section 14.6.2.31
SCR_REG	RW	8	0x0000 0040	Section 14.6.2.33
SSR_REG	R	8	0x0000 0044	Section 14.6.2.34
SYSC_REG	RW	8	0x0000 0054	Section 14.6.2.36
SYSS_REG	R	8	0x0000 0058	Section 14.6.2.37
WER_REG	RW	8	0x0000 005C	Section 14.6.2.38
CFPS_REG	RW	8	0x0000 0060	Section 14.6.2.39
Condition: LCR_REG[7]=0x1 and LCR_REG[7:0]!=0xBF				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-39. UART1/2/3/4 Subconfiguration_Mode_A Mode Summary

Mode	Condition
MSR_SPR	(EFR_REG[4]=0x0 or MCR_REG[6]=0x0)
TCR_TLR	EFR_REG[4]=0x1 and MCR_REG[6]=0x1

Table 14-40. UART1/2/3/4 Register Summary for Sub-Configuration_Mode_A Mode: MSR_SPR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
MSR_REG	R	8	0x0000 0018	Section 14.6.2.16
SPR_REG	RW	8	0x0000 001C	Section 14.6.2.17
Condition: (EFR_REG[4]=0x0 or MCR_REG[6]=0x0)				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-41. UART1/2/3/4 Register Summary for Sub-Configuration_Mode_A Mode: TCR_TLR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
TCR_REG	RW	8	0x0000 0018	Section 14.6.2.15
TLR_REG	RW	8	0x0000 001C	Section 14.6.2.19
Condition: EFR_REG[4]=0x1 and MCR_REG[6]=0x1				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-42. UART1/2/3/4 Register Summary for Configuration_Mode_B Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
DLL_REG	RW	8	0x0000 0000	Section 14.6.2.1
DLH_REG	RW	8	0x0000 0004	Section 14.6.2.5
EFR_REG	RW	8	0x0000 0008	Section 14.6.2.8
LCR_REG	RW	8	0x0000 000C	Section 14.6.2.9
XON1_ADDR1_REG	RW	8	0x0000 0010	Section 14.6.2.11
XON2_ADDR2_REG	RW	8	0x0000 0014	Section 14.6.2.13
See Table 14-43 and either Table 14-44 or Table 14-45.			0x0000 0018	
See Table 14-43 and either Table 14-44 or Table 14-45.			0x0000 001C	
MDR1_REG	RW	8	0x0000 0020	Section 14.6.2.20
MDR2_REG	RW	8	0x0000 0024	Section 14.6.2.21
SFLSR_REG	R	8	0x0000 0028	Section 14.6.2.23
TXFLL_REG	W	8	0x0000 0028	Section 14.6.2.22
RESUME_REG	R	8	0x0000 002C	Section 14.6.2.24
TXFLH_REG	W	8	0x0000 002C	Section 14.6.2.25
RXFLL_REG	W	8	0x0000 0030	Section 14.6.2.26
SFREGL_REG	R	8	0x0000 0030	Section 14.6.2.27
RXFLH_REG	W	8	0x0000 0034	Section 14.6.2.29
SFREGH_REG	R	8	0x0000 0034	Section 14.6.2.28
UASR_REG	R	8	0x0000 0038	Section 14.6.2.31
SCR_REG	RW	8	0x0000 0040	Section 14.6.2.33
SSR_REG	R	8	0x0000 0044	Section 14.6.2.34
SYSC_REG	RW	8	0x0000 0054	Section 14.6.2.36
SYSS_REG	R	8	0x0000 0058	Section 14.6.2.37
WER_REG	RW	8	0x0000 005C	Section 14.6.2.38
CFPS_REG	RW	8	0x0000 0060	Section 14.6.2.39
Condition: LCR_REG[7:0]=0xBF				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-43. UART1/2/3/4 Sub-Configuration_Mode_B Mode Summary

Mode	Condition
TCR_TLR	EFR_REG[4]=0x1 and MCR_REG[6]=0x1
XOFF	(EFR_REG[4]=0x0 or MCR_REG[6]=0x0)

Table 14-44. UART1/2/3/4 Register Summary for Sub-Configuration_Mode_B Mode: TCR_TLR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
TCR_REG	RW	8	0x0000 0018	Section 14.6.2.15
TLR_REG	RW	8	0x0000 001C	Section 14.6.2.19
Condition: EFR_REG[4]=0x1 and MCR_REG[6]=0x1				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-45. UART1/2/3/4 Register Summary for Sub-Configuration_Mode_B Mode: XOFF Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
XOFF1_REG	RW	8	0x0000 0018	Section 14.6.2.14
XOFF2_REG	RW	8	0x0000 001C	Section 14.6.2.18
Condition: (EFR_REG[4]=0x0 or MCR_REG[6]=0x0)				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-46. UART1/2/3/4 Register Summary for Operational_Mode Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
RHR_REG	R	8	0x0000 0000	Section 14.6.2.2
THR_REG	W	8	0x0000 0000	Section 14.6.2.3
IER_REG	RW	8	0x0000 0004	Section 14.6.2.4
IIR_REG	R	8	0x0000 0008	Section 14.6.2.7
FCR_REG	W	8	0x0000 0008	Section 14.6.2.6
LCR_REG	RW	8	0x0000 000C	Section 14.6.2.9
MCR_REG	RW	8	0x0000 0010	Section 14.6.2.10
LSR_REG	R	8	0x0000 0014	Section 14.6.2.12
See Table 14-47 and either Table 14-48 or Table 14-49.			0x0000 0018	
See Table 14-47 and either Table 14-48 or Table 14-49.			0x0000 001C	
MDR1_REG	RW	8	0x0000 0020	Section 14.6.2.20
MDR2_REG	RW	8	0x0000 0024	Section 14.6.2.21
SFLSR_REG	R	8	0x0000 0028	Section 14.6.2.23
TXFLL_REG	W	8	0x0000 0028	Section 14.6.2.22
RESUME_REG	R	8	0x0000 002C	Section 14.6.2.24
TXFLH_REG	W	8	0x0000 002C	Section 14.6.2.25
RXFLL_REG	W	8	0x0000 0030	Section 14.6.2.26
SFREGL_REG	R	8	0x0000 0030	Section 14.6.2.27
RXFLH_REG	W	8	0x0000 0034	Section 14.6.2.29
SFREGH_REG	R	8	0x0000 0034	Section 14.6.2.28
BLR_REG	RW	8	0x0000 0038	Section 14.6.2.30
ACREG_REG	RW	8	0x0000 003C	Section 14.6.2.32

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-46. UART1/2/3/4 Register Summary for Operational_Mode Mode Active (continued)

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
SCR_REG	RW	8	0x0000 0040	Section 14.6.2.33
SSR_REG	R	8	0x0000 0044	Section 14.6.2.34
EBLR_REG	RW	8	0x0000 0048	Section 14.6.2.35
SYSC_REG	RW	8	0x0000 0054	Section 14.6.2.36
SYSS_REG	R	8	0x0000 0058	Section 14.6.2.37
WER_REG	RW	8	0x0000 005C	Section 14.6.2.38
CFPS_REG	RW	8	0x0000 0060	Section 14.6.2.39
Condition: LCR_REG[7]=0x0				

Table 14-47. UART1/2/3/4 Sub-Operational_Mode Mode Summary

Mode	Condition
MSR_SPR	(EFR_REG[4] = 0x0 or MCR_REG[6] = 0x0)
TCR_TLR	EFR_REG[4] = 0x1 and MCR_REG[6]= 0x1

Table 14-48. UART1/2/3/4 Register Summary for Sub-Operational_Mode Mode: MSR_SPR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
MSR_REG	R	8	0x0000 0018	Section 14.6.2.16
SPR_REG	RW	8	0x0000 001C	Section 14.6.2.17
Condition: (EFR_REG[4] = 0x0 or MCR_REG[6] = 0x0)				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

Table 14-49. UART1/2/3/4 Register Summary for Suboperational_Mode Mode: TCR_TLR Mode Active

Register Name	Type	Register Width (Bits)	Physical Address[1]	Section
TCR_REG	RW	8	0x0000 0018	Section 14.6.2.15
TLR_REG	RW	8	0x0000 001C	Section 14.6.2.19
Condition: EFR_REG[4] = 0x1 and MCR_REG[6] = 0x1				

(1) To get the physical address for a specific instance of a register, add the corresponding module base address.

14.6.2 UART/IrDA/CIR Register Descriptions

14.6.2.1 DLL_REG

Table 14-50. DLL_REG

Address Offset	0x000		
Physical Address	0x4806 A000	**Instance**	UART1
	0x4806 C000		UART2
	0x4902 0000		UART3
	0x4809 E000		UART4
Description	Divisor latches low		
	This register, with DLH_REG, stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH_REG stores the most-significant part of the divisor. DLL_REG stores the least-significant part of the divisor.		
	Note: DLL_REG and DLH_REG can be written to only before sleep mode is enabled (before IER_REG[4] is set).		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CLOCK_LSB							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	CLOCK_LSB	Stores the 8-bit LSB divisor value	RW	0x00

14.6.2.2 RHR_REG

Table 14-51. RHR_REG

Address Offset	0x000		
Physical Address	0x4806 A000	**Instance**	UART1
	0x4806 C000		UART2
	0x4902 0000		UART3
	0x4809 E000		UART4
Description	Receive holding register		
	The receiver section consists of the receiver holding register (RHR_REG) and the receiver shift register. The RHR_REG is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR_REG. If the FIFO is disabled, location zero of the FIFO is used to store the single data character.		
	Note: If an overflow occurs the data in the RHR_REG is not overwritten.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RHR							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	RHR	Receive holding register	R	0x–

14.6.2.3 THR_REG

Table 14-52. THR_REG

Address Offset	0x000		
Physical Address	0x4806 A000	**Instance**	UART1
	0x4806 C000		UART2
	0x4902 0000		UART3
	0x4809 E000		UART4
Description	Transmit holding register		
	The transmitter section consists of the transmit holding register (THR_REG) and the transmit shift register. The transmit holding register is a 64-byte FIFO. The MPU writes data to the THR_REG. The data is placed in the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location zero of the FIFO is used to store the data.		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no functional effect.	W	0x00
7:0	THR	Transmit holding register	W	0x–

14.6.2.4 IER_REG

Table 14-53. IER_REG

Address Offset	0x004		
Physical Address	0x4806 A004	**Instance**	UART1
	0x4806 C004		UART2
	0x4902 0004		UART3
	0x4809 E000		UART4
Description	Interrupt enable register		
Type	RW		

14.6.2.4.1 UART Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CTS_IT	RTS_IT	XOFF_IT	SLEEP_MODE	MODEM_STS_IT	LINE_STS_IT	THR_IT	RHR_IT

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00. Write has no functional effect.	RW	0x00
7	CTS_IT	Can be written only when EFR_REG[4] = 1	RW	0
		0x0: Disables the nCTS interrupt		
		0x1: Enables the nCTS interrupt		
6	RTS_IT	Can be written only when EFR_REG[4] = 1	RW	0
		0x0: Disables the interrupt		
		0x1: Enables the nRTS interrupt		
5	XOFF_IT	Can be written only when EFR_REG[4] = 1	RW	0
		0x0: Disables the XOFF interrupt		
		0x1: Enables the XOFF interrupt		
4	SLEEP_MODE	Can be only written when EFR_REG[4] = 1	RW	0
		0x0: Disables sleep mode		
		0x1: Enables sleep mode (stop baud rate clock when the module is inactive)		
3	MODEM_STS_IT		RW	0
		0x0: Disables the modem status register interrupt		
		0x1: Enables the modem status register interrupt		
2	LINE_STS_IT		RW	0
		0x0: Disables the receiver line status interrupt		
		0x1: Enables the receiver line status interrupt		
1	THR_IT		RW	0
		0x0: Disables the THR interrupt		
		0x1: Enables the THR interrupt		
0	RHR_IT		RW	0
		0x0: Disables the RHR interrupt and time out interrupt.		
		0x1: Enables the RHR interrupt and time out interrupt.		

14.6.2.4.2 IrDA Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EOF_IT	LINE_STS_IT_I	TX_STATUS_IT	STS_FIFO_ TRIG_IT	RX_OVERRUN_ IT	LAST_RX_ BYTE_IT	THR_IT	RHR_IT

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0. Write has no functional effect.	RW	0x00
7	EOF_IT			
		0x0: Disables the received EOF interrupt		
		0x1: Enables the received EOF interrupt		
6	LINE_STS_IT_I		RW	0
		0x0: Disables the receiver line status interrupt		
		0x1: Enables the receiver line status interrupt		
5	TX_STATUS_IT	TX_STATUS_IT interrupt reflects two possible conditions. The MDR2_REG[0] must be read to determine the status in the event of this interrupt.	RW	0
		0x0: Disables the TX status interrupt		
		0x1: Enables the TX status interrupt		
4	STS_FIFO_ TRIG_IT		RW	0
		0x0: Disables status FIFO trigger level interrupt		
		0x1: Enables status FIFO trigger level interrupt.		
3	RX_OVERRUN_ IT		RW	0
		0x0: Disables the RX overrun interrupt		
		0x1: Enables the RX overrun interrupt		
2	LAST_RX_ BYTE_IT		RW	0
		0x0: Disables the last byte of frame in RX FIFO interrupt		
		0x1: Enables the last byte of frame in RX FIFO interrupt		
1	THR_IT		RW	0
		0x0: Disables the THR interrupt		
		0x1: Enables the THR interrupt		
0	RHR_IT		RW	0
		0x0: Disables the RHR interrupt		
		0x1: Enables the RHR interrupt		

14.6.2.4.3 CIR Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TX_STATUS_IT	Reserved			THR_IT	Reserved

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Read returns 0x00. Write has no functional effect.	RW	0x00
5	TX_STATUS_IT	In IR-CIR mode, contrary to the IR-IrDA mode, the TX_STATUS_IT has only one meaning corresponding to the case MDR2_REG[0] = 0.	RW	0
4:2	Reserved	Read returns 0. Write has no functional effect.	R	0
1	THR_IT	0x0: Disables the THR interrupt 0x1: Enables the THR interrupt	RW	0
0	Reserved	Reserved	R	0

14.6.2.5 DLH_REG

Table 14-54. DLH_REG

Address Offset	0x004		
Physical Address	0x4806 A004	**Instance**	UART1
	0x4806 C004		UART2
	0x4902 0004		UART3
	0x4809 E000		UART4
Description	Divisor latches high		
	This register, with DLL_REG, stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH_REG stores the most-significant part of the divisor. DLL_REG stores the least-significant part of the divisor.		
	Note: DLL_REG and DLH_REG can be written to only before sleep mode is enabled (before IER_REG[4] is set).		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved		CLOCK_MSB					

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:6	Reserved	Read returns 0x0.	R	0x0
5:0	CLOCK_MSB	Stores the 6-bit most-significant bit (MSB) divisor value	RW	0x00

14.6.2.6 FCR_REG

Table 14-55. FCR_REG

Address Offset	0x008		
Physical Address	0x4806 A008	**Instance**	UART1
	0x4806 C008		UART2
	0x4902 0008		UART3
	0x4809 E000		UART4
Description	FIFO control register		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG		TX_FIFO_TRIG		DMA_MODE	TX_FIFO_CLEAR	RX_FIFO_CLEAR	FIFO_EN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no functional effect.	W	0x00
7:6	RX_FIFO_TRIG	Sets the trigger level for the RX FIFO: If SCR_REG[7] = 0 and TLR_REG[7:4] ≠ 0000 RX_FIFO_TRIG is not considered. If SCR_REG[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with the granularity 1. If SCR_REG[7] = 0 and TLR_REG[7:4] = 0000: 0x0: 8 characters 0x1: 16 characters 0x2: 56 characters 0x3: 60 characters	W	0x0
5:4	TX_FIFO_TRIG	Can be written only if EFR_REG[4] = 1. Sets the trigger level for the TX FIFO: If SCR_REG[6] = 0 and TLR_REG[3:0] ≠ 0000, TX_FIFO_TRIG is not considered. If SCR_REG[6] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with a granularity of 1. If SCR_REG[6] = 0 and TLR_REG[3:0] = 0000: 0x0: 8 characters 0x1: 16 characters 0x2: 32 characters 0x3: 56 characters	W	0x0
3	DMA_MODE	Can be changed only when the baud clock is not running (DLL_REG and DLH_REG set to 0). This register is considered if SCR_REG[0] = 0. 0x0: DMA_MODE 0 (No DMA) 0x1: DMA_MODE 1 (UART_NDMA_REQ[0] in TX, UART_NDMA_REQ[1] in RX)	W	0
2	TX_FIFO_CLEAR	0x0: No change 0x1: Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.	W	0

Bits	Field Name	Description	Type	Reset
1	RX_FIFO_CLEAR		W	0
		0x0: No change		
		0x1: Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.		
0	FIFO_EN	Can be changed only when the baud clock is not running (DLL_REG and DLH_REG set to 0).	W	0
		0x0: Disables the transmit and receive FIFOs. The transmit and receive holding registers are 1-byte FIFOs.		
		0x1: Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.		

14.6.2.7 IIR_REG

Table 14-56. IIR_REG

Address Offset	0x008		
Physical Address	0x4806 A008	**Instance**	UART1
	0x4806 C008		UART2
	0x4902 0008		UART3
	0x4809 E000		UART4
Description	Interrupt Identification register		
	The IIR_REG is a read-only register that provides the source of the interrupt in a prioritized manner.		
	Note: An interrupt source can be flagged only if enabled in the IER_REG register.		
Type	R		

14.6.2.7.1 UART Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FCR_MIRROR		IT_TYPE					IT_PENDING

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Read returns 0x00.		R	0x00
7:6	FCR_MIRROR	Mirror the contents of FCR_REG[0] on both bits.		R	0x0
5:1	IT_TYPE	Seven possible interrupts in UART mode; other combinations never occur:		R	0x00
		0x0:	Modem interrupt. Priority = 4		
		0x1:	THR interrupt. Priority = 3		
		0x2:	RHR interrupt. Priority = 2		
		0x3:	Receiver line status error. Priority = 1		
		0x6:	Rx timeout. Priority = 2		
		0x8:	Xoff/special character. Priority = 5		
		0x10:	CTS, RTS change state from active (low) to inactive (high). Priority = 6		
0	IT_PENDING	0x0:	An interrupt is pending.	R	1
		0x1:	No interrupt is pending.		

14.6.2.7.2 IrDA Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EOF_IT	LINE_STS_IT	TX_STATUS_IT	STS_FIFO_IT	RX_OE_IT	RX_FIFO_LB_IT	THR_IT	RHR_IT

Bits	Field Name	Description		Type	Reset
15:8	Reserved	Read returns 0x00.		R	0x00
7	EOF_IT				
		0x0:	Received EOF interrupt inactive		
		0x1:	Received EOF interrupt active		
6	LINE_STS_IT			R	0
		0x0:	Receiver line status interrupt inactive		
		0x1:	Receiver line status interrupt active		
5	TX_STATUS_IT			R	0
		0x0:	TX status interrupt inactive		
		0x1:	TX status interrupt active		
4	STS_FIFO_IT			R	0
		0x0:	Status FIFO trigger level interrupt inactive		
		0x1:	Status FIFO trigger level interrupt active		
3	RX_OE_IT			R	0
		0x0:	RX overrun interrupt inactive		
		0x1:	RX overrun interrupt active		
2	RX_FIFO_LB_IT		Receive FIFO last byte interrupt	R	0
		0x0:	Last byte of frame in RX FIFO interrupt inactive		
		0x1:	Last byte of frame in RX FIFO interrupt active		
1	THR_IT			R	0
		0x0:	THR interrupt inactive		
		0x1:	THR interrupt active		
0	RHR_IT			R	0
		0x0:	RHR interrupt inactive		
		0x1:	RHR interrupt active		

14.6.2.7.3 CIR Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TX_STATUS_IT	Reserved			THR_IT	Reserved

Bits	Field Name	Description	Type	Reset
15:6	Reserved	Read returns 0x00.	R	0x00
5	TX_STATUS_IT		R	0
		0x0: TX status interrupt inactive		
		0x1: TX status interrupt active		
4:2	Reserved	Read returns 0	R	0
1	THR_IT		R	0
		0x0: THR interrupt inactive		
		0x1: THR interrupt active		
0	Reserved	Reserved	R	0

14.6.2.8 EFR_REG

Table 14-57. EFR_REG

Address Offset	0x008		
Physical Address	0x4806 A008	**Instance**	UART1
	0x4806 C008		UART2
	0x4902 0008		UART3
	0x4809 E000		UART4
Description	Enhanced feature register		
	This register enables or disables enhanced features. Most enhanced functions apply only to UART modes, but EFR_REG[4] enables write accesses to FCR_REG[5:4], the TX trigger level, which is also used in IrDA modes.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								AUTO_CTS_EN	AUTO_RTS_EN	SPEC_CHAR	ENHANCED_EN	SW_FLOW_CONTROL			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	AUTO_CTS_EN	Auto-CTS enable bit (UART mode only) 0x0: Normal operation 0x1: Auto-CTS flow control is enabled; transmission is halted when the nCTS pin is high (inactive).	RW	0
6	AUTO_RTS_EN	Auto-RTS enable bit (UART mode only) 0x0: Normal operation 0x1: Auto-RTS flow control is enabled; nRTS pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR_REG[3:0], is reached and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.	RW	0
5	SPEC_CHAR	(UART mode only) Special character detect 0x0: Normal operation 0x1: Special character detect enable. Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to RX FIFO and IIR_REG bit 4 is set to 1 to indicate that a special character was detected.	RW	0
4	ENHANCED_EN	Enhanced functions write enable bit 0x0: Disables writing to IER_REG bits [7:4], FCR_REG bits [5:4], and MCR_REG bits [7:5] 0x1: Enables writing to IER_REG bits [7:4], FCR_REG bits [5:4], and MCR_REG bits [7:5]	RW	0
3:0	SW_FLOW_CONTROL	Combinations of software flow control can be selected by programming bit [3:0]. See Table 14-29. In IrDA mode, bits [1:0] select IR address to check. See Section 14.2.5.2.1.7, *IR Address Checking*.	RW	0x0

14.6.2.9 LCR_REG

Table 14-58. LCR_REG

Address Offset	0x00C		
Physical Address	0x4806 A00C	**Instance**	UART1
	0x4806 C00C		UART2
	0x4902 000C		UART3
	0x4809 E000		UART4
Description	Line control register		
	LCR_REG[6:0] define parameters of the transmission and reception for UART mode. LCR_REG[7] is used to put the module in operational mode or Configuration_Mode_A/B.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DIV_EN	BREAK_EN	PARITY_TYPE2	PARITY_TYPE1	PARITY_EN	NB_STOP	CHAR_LENGTH	

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00	R	0x00
7	DIV_EN	0x0: Operational mode 0x1: Divisor latch enable; put the module in Configuration_Mode_A/B. Allows access to DLL_REG, DLH_REG, and other registers (see Section 14.6.1, *UART/IrDA/CIR Register Mapping Summary*). Configuration_Mode_B: LCR_REG[7:0] = 0xBF Else, Configuration_Mode_A	RW	0
6	BREAK_EN	Break control bit. UART mode only. **Note:** When LCR_REG[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR_REG[6] = 1. 0x0: Normal operating condition 0x1: Forces the transmitter output to go low to alert the communication terminal. TX line is forced to 0 and remains in this state while BREAK_EN = 1.	RW	0
5	PARITY_TYPE2	Selects the forced parity format (if LCR_REG[3] = 1). UART mode only. If LCR_REG[5] = 1and LCR_REG[4] = 0, the parity bit is forced to 1 in the transmitted and received data. If LCR_REG[5] = 1 and LCR_REG[4] = 1, the parity bit is forced to 0 in the transmitted and received data.	RW	0
4	PARITY_TYPE1	UART mode only 0x0: Odd parity is generated (if LCR_REG[3] = 1). 0x1: Even parity is generated (if LCR_REG[3] = 1).	RW	0
3	PARITY_EN	UART mode only 0x0: No parity 0x1: A parity bit is generated during transmission, and the receiver checks for received parity.	RW	0
2	NB_STOP	Specifies the number of stop bits. UART mode only. 0x0: 1 stop bit (word length = 5, 6, 7, 8) 0x1: 1.5 stop bits (word length = 5)1 - 2 stop bits (word length = 6, 7, 8)	RW	0

Bits	Field Name	Description	Type	Reset
1:0	CHAR_LENGTH	Specifies the word length to be transmitted or received. UART mode only. 0x0: 5 bits 0x1: 6 bits 0x2: 7 bits 0x3: 8 bits	RW	0x0

14.6.2.10 MCR_REG

Table 14-59. MCR_REG

Address Offset	0x010		
Physical Address	0x4806 A010	**Instance**	UART1
	0x4806 C010		UART2
	0x4902 0010		UART3
	0x4809 E000		UART4
Description	Modem control register		
	MCR_REG[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved	TCR_TLR	XON_EN	LOOPBACK_EN	CD_STS_CH	RI_STS_CH	RTS	DTR

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	Reserved	Read returns 0. Write has no functional effect.	RW	0
6	TCR_TLR	Can be written only when EFR_REG[4] ENHANCED_EN = 1 0x0: No action 0x1: Enables access to the TCR_REG and TLR_REG registers	RW	0
5	XON_EN	Can be written only when EFR_REG[4] ENHANCED_EN = 1 0x0: Disable XON any function 0x1: Enable XON any function	RW	0
4	LOOPBACK_EN	0x0: Normal operating mode 0x1: Enable local loopback mode (internal). In this mode, the MCR_REG[3:0] signals are looped back into MSR_REG[7:4]. The transmit output is looped back to the receive input internally.	RW	0
3	CD_STS_CH	0x0: In loopback, forces nDCD input high and IRQ outputs to INACTIVE state. 0x1: In loopback, forces nDCD input low and IRQ outputs to INACTIVE state.	RW	0
2	RI_STS_CH	0x0: In loopback, forces nRI input inactive (high). 0x1: In loopback, forces nRI input active (low).	RW	0
1	RTS	In loop back, controls MSR_REG[4].If auto-RTS is enabled, the nRTS output is controlled by hardware flow control. 0x0: Force nRTS output to inactive (high). 0x1: Force nRTS output to active (low).	RW	0
0	DTR	0x0: Force DTR output (used in loop back mode) to inactive (high). 0x1: Force DTR output (used in loop back mode) to active (low).	RW	0

14.6.2.11 XON1_ADDR1_REG

Table 14-60. XON1_ADDR1_REG

Address Offset	0x010		
Physical Address	0x4806 A010	**Instance**	UART1
	0x4806 C010		UART2
	0x4902 0010		UART3
	0x4809 E000		UART4
Description	UART mode: XON1 character, IrDA mode: ADDR1 address		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XON_WORD1							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00	R	0x00
7:0	XON_WORD1	Used to store the 8-bit XON1 character in UART modes and ADDR1 address 1 for IrDA modes	RW	0x00

14.6.2.12 LSR_REG

Table 14-61. LSR_REG

Address Offset	0x014		
Physical Address	0x4806 A014	**Instance**	UART1
	0x4806 C014		UART2
	0x4902 0014		UART3
	0x4809 E000		UART4
Description	Line status register		
Type	R		

14.6.2.12.1 UART Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_STS	TX_SR_E	TX_FIFO_E	RX_BI	RX_FE	RX_PE	RX_OE	RX_FIFO_E

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	RX_FIFO_STS		R	0
		0x0: Normal operation		
		0x1: At least one parity error, framing error, or break indication in the RX FIFO. Bit 7 is cleared when no errors are present in the RX FIFO.		
6	TX_SR_E		R	1
		0x0: Transmitter hold (TX FIFO) and shift registers are not empty.		
		0x1: Transmitter hold (TX FIFO) and shift registers are empty		
5	TX_FIFO_E		R	1
		0x0: Transmit hold register (TX FIFO) is not empty.		
		0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily complete.		
4	RX_BI		R	0
		0x0: No break condition		
		0x1: A break was detected while the data being read from the RX FIFO was being received (RX input was low for one character + 1 bit time frame).		
3	RX_FE		R	0
		0x0: No framing error in data being read from RX FIFO		
		0x1: Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).		
2	RX_PE		R	0
		0x0: No parity error in data being read from RX FIFO		
		0x1: Parity error in data being read from RX FIFO		
1	RX_OE		R	0
		0x0: No overrun error		
		0x1: Overrun error occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full.		
0	RX_FIFO_E		R	0

Bits	Field Name	Description	Type	Reset
		0x0: No data in the receive FIFO		
		0x1: At least one data character in the RX FIFO		

14.6.2.12.2 IrDA Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR_EMPTY	STS_FIFO_FUL	RX_LAST_BYTE	FRAME_TOO_LONG	ABORT	CRC	STS_FIFO_E	RX_FIFO_E

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	THR_EMPTY		R	0
		0x0: Transmit holding register (TX FIFO) is not empty.		
		0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily complete.		
6	STS_FIFO_FUL		R	1
		0x0: Status FIFO not full		
		0x1: Status FIFO full		
5	RX_LAST_BYTE	Receive last byte	R	1
		0x0: The RX FIFO (RHR_REG) does not contain the last byte of the frame to be read.		
		0x1: The RX FIFO (RHR_REG) contains the last byte of the frame to be read. This bit is set only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR_REG register.		
4	FRAME_TOO_LONG	Frame too long	R	0
		0x0: No frame-too-long error in frame		
		0x1: Frame-too-long error in the frame at the top of the STATUS FIFO (next character to be read). This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH_REG and RXFLL_REG registers) is received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.		
3	ABORT		R	0
		0x0: No abort pattern error in frame		
		0x1: Abort pattern received. SIR and MIR: abort pattern. FIR: Illegal symbol.		
2	CRC		R	0
		0x0: No CRC error in frame		
		0x1: CRC error in the frame at the top of the STATUS FIFO (next character to be read)		
1	STS_FIFO_E		R	0
		0x0: Status FIFO not empty		
		0x1: Status FIFO empty		
0	RX_FIFO_E		R	1
		0x0: At least one data character in the RX FIFO		

Bits	Field Name	Description	Type	Reset
		0x1: No data in the receive FIFO		

14.6.2.12.3 CIR Bitfield Details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								THR_EMPTY	Reserved						

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	THR_EMPTY		R	0
		0x0: Transmit holding register (TX FIFO) is not empty.		
		0x1: Transmit hold register (TX FIFO) is empty. The transmission is not necessarily complete.		
6:0	Reserved	Reserved	R	1

14.6.2.13 XON2_ADDR2_REG

Table 14-62. XON2_ADDR2_REG

Address Offset	0x014		
Physical Address	0x4806 A014	**Instance**	UART1
	0x4806 C014		UART2
	0x4902 0014		UART3
	0x4809 E000		UART4
Description			
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XON_WORD2							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	XON_WORD2	Stores the 8-bit XON2 character in UART modes and ADDR2 address 2 for IrDA modes	RW	0x00

14.6.2.14 XOFF1_REG

Table 14-63. XOFF1_REG

Address Offset	0x018		
Physical Address	0x4806 A018	**Instance**	UART1
	0x4806 C018		UART2
	0x4902 0018		UART3
	0x4809 E000		UART4
Description	UART mode XOFF1 character		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XOFF_WORD1							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00	R	0x00
7:0	XOFF_WORD1	Stores the 8-bit XOFF1 character used in UART modes	RW	0x00

14.6.2.15 TCR_REG

Table 14-64. TCR_REG

Address Offset	0x018		
Physical Address	0x4806 A018	**Instance**	UART1
	0x4806 C018		UART2
	0x4902 0018		UART3
	0x4809 E000		UART4
Description	Transmission control register.		
	This register stores the receive FIFO threshold levels to start/stop transmission during hardware flow control.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG_ START				RX_FIFO_TRIG_ HALT			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:4	RX_FIFO_TRIG_ START	RX FIFO trigger level to RESTORE transmission (0 to 60)	RW	0x0
3:0	RX_FIFO_TRIG_ HALT	RX FIFO trigger level to HALT transmission (0 to 60)	RW	0xF

14.6.2.16 MSR_REG

Table 14-65. MSR_REG

Address Offset	0x018		
Physical Address	0x4806 A018	**Instance**	UART1
	0x4806 C018		UART2
	0x4902 0018		UART3
	0x4809 E000		UART4
Description	Modem status register. UART mode only.		
	This register provides information about the current state of the control lines from the modem, data set, or peripheral device to the MPU. It also indicates when a control input from the modem changes state.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								NCD_STS	NRI_STS	NDSR_STS	NCTS_STS	DCD_STS	RI_STS	DSR_STS	CTS_STS

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	NCD_STS	In loopback mode, it is equivalent to MCR_REG[3].	R	-
6	NRI_STS	This bit is the complement of the nRI input. In loopback mode, it is equivalent to MCR_REG[2].	R	-
5	NDSR_STS	In loopback mode, it is equivalent to MCR_REG[0].	R	-
4	NCTS_STS	This bit is the complement of the nCTS input. In loopback mode, it is equivalent to MCR_REG[1].	R	-
3	DCD_STS	Indicates that MCR_REG[3] in loopback changed. Cleared on a read.	R	0
2	RI_STS	Indicates that nRI input (or MCR_REG[2] in loopback) changed state from low to high. Cleared on a read.	R	0
1	DSR_STS	0x1: Indicates that MCR_REG[0] in loopback changed state. Cleared on a read.	R	0
0	CTS_STS	0x1: Indicates that nCTS input (or MCR_REG[1] in loopback) changed state. Cleared on a read.	R	0

14.6.2.17 SPR_REG

Table 14-66. SPR_REG

Address Offset	0x01C		
Physical Address	0x4806 A01C	**Instance**	UART1
	0x4806 C01C		UART2
	0x4902 001C		UART3
	0x4809 E000		UART4
Description	Scratchpad register		
	This read/write register does not control the module. It is a scratchpad register used by the programmer to hold temporary data.		
Type			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SPR_WORD							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	SPR_WORD	Scratchpad register	RW	0x00

14.6.2.18 XOFF2_REG

Table 14-67. XOFF2_REG

Address Offset	0x01C		
Physical Address	0x4806 A01C	**Instance**	UART1
	0x4806 C01C		UART2
	0x4902 001C		UART3
	0x4809 E000		UART4
Description	UART mode XOFF2 character.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								XOFF_WORD2							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	XOFF_WORD2	Stores the 8-bit XOFF2 character used in UART modes	RW	0x00

14.6.2.19 TLR_REG

Table 14-68. TLR_REG

Address Offset	0x01C		
Physical Address	0x4806 A01C	**Instance**	UART1
	0x4806 C01C		UART2
	0x4902 001C		UART3
	0x4809 E000		UART4
Description	Trigger level register.		
	Stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_FIFO_TRIG_ DMA				TX_FIFO_TRIG_ DMA			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:4	RX_FIFO_TRIG_ DMA	Receive FIFO trigger level	RW	0x0
3:0	TX_FIFO_TRIG_ DMA	Transmit FIFO trigger level	RW	0x0

14.6.2.20 MDR1_REG

Table 14-69. MDR1_REG

Address Offset	0x020		
Physical Address	0x4806 A020	**Instance**	UART1
	0x4806 C020		UART2
	0x4902 0020		UART3
	0x4809 E000		UART4
Description	Mode definition register 1.		
	The mode of operation can be programmed by writing to MDR1_REG[2:0]; therefore, the MDR1_REG must be programmed on startup after configuration of the configuration registers (DLL_REG, DLH_REG, LCR_REG). The value of MDR1_REG[2:0] must not be changed again during normal operation.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FRAME_END_M ODE	SIP_MODE	SCT	SET_TXIR	IR_SLEEP	MODE_SELECT		

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	FRAME_END_ MODE	IrDA mode only	RW	0
		0x0: Frame-length method		
		0x1: Set EOT bit method		
6	SIP_MODE	MIR/FIR modes only. IrDA only.	RW	0
		0x0: Manual SIP mode: SIP is generated with the control of ACREG_REG[3].		
		0x1: Automatic SIP mode: SIP is generated after each transmission.		
5	SCT	Store and control the transmission.	RW	0
		0x0: Starts the infrared transmission when a value is written to THR_REG		
		0x1: Starts the infrared transmission with the control of ACREG_REG[2]		
		Note: Before starting any transmission, there must be no reception ongoing.		
4	SET_TXIR	Used to configure the infrared transceiver. IrDA only.	RW	0
		0x0: No action		
		0x1: IRTX pin output is forced high.		
3	IR_SLEEP		RW	0
		0x0: IrDA/CIR sleep mode disabled		
		0x1: IrDA/CIR sleep mode enabled		
2:0	MODE_SELECT	UART-IrDA-CIR mode selection	RW	0x7
		0x0: UART 16x mode		
		0x1: SIR mode		
		0x2: UART 16x auto-baud		
		0x3: UART 13x mode		

Bits	Field Name	Description	Type	Reset
		0x4: MIR mode		
		0x5: FIR mode		
		0x6: CIR mode		
		0x7: Disable (default state)		

14.6.2.21 MDR2_REG

Table 14-70. MDR2_REG

Address Offset	0x024		
Physical Address	0x4806 A024	**Instance**	UART1
	0x4806 C024		UART2
	0x4902 0024		UART3
	0x4809 E000		UART4
Description	Mode definition register 2		
	IR-IrDA and IR-CIR modes only		
	MDR2_REG[0] describes the status of the interrupt in IIR_REG[5]. The IRTX_UNDERRUN bit must be read after an IIR_REG[5] TX_STATUS_IT interrupt occurs. The bits [2:1] of this register set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1_REG[2:0].		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved	IRRXINVERT	CIR_PULSE_ MODE		UART_PULSE	STS_FIFO_TRIG		IRTX_UNDERRUN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	Reserved		R	0
6	IRRXINVERT	Only for IR mode (IrDA and CIR). Invert RX pin in the module before the voting or sampling system logic of the infrared block. This does not affect the RX path in UART modem modes. 0x0: Inversion is performed. 0x1: No inversion is performed.	RW	0
5:4	CIR_PULSE_ MODE	CIR pulse modulation definition. Defines high level of the pulse width associated with a digit: 0x0: Pulse width of 3 from 12 cycles 0x1: Pulse width of 4 from 12 cycles 0x2: Pulse width of 5 from 12 cycles 0x3: Pulse width of 6 from 12 cycles	RW	0x00
3	UART_PULSE	UART mode only. Used to allow pulse shaping in UART mode. 0x0: Normal UART mode 0x1: UART mode with pulse shaping	RW	0
2:1	STS_FIFO_TRIG	Only for IR-IrDA mode Frame status FIFO threshold select: 0x0: 1 entry 0x1: 4 entries 0x2: 7 entries 0x3: 8 entries	RW	0x00
0	IRTX_UNDERRUN	IrDA transmission status interrupt. When the IIR_REG[5] interrupt occurs, the meaning of the interrupt is: 0x0: The last bit of the frame was transmitted successfully without error.	R	0

Bits	Field Name	Description	Type	Reset
		0x1: An underrun occurred. The last bit of the frame was transmitted but with an underrun error. The bit is reset to 0 when the RESUME_REG register is read.		

14.6.2.22 TXFLL_REG

Table 14-71. TXFLL_REG

Address Offset	0x028		
Physical Address	0x4806 A028	**Instance**	UART1
	0x4806 C028		UART2
	0x4902 0028		UART3
	0x4809 E000		UART4
Description	Transmit frame length register low		
	IrDA modes only		
	The registers TXFLL_REG and TXFLH_REG hold the 13-bit transmit frame length (expressed in bytes). TXFLL_REG holds the LSBs and TXFLH_REG holds the MSBs. The frame length value is used if the frame length method of frame closing is used.		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXFLL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no functional effect.	W	0x00
7:0	TXFLL	LSB register used to specify the frame length	W	0x00

14.6.2.23 SFLSR_REG

Table 14-72. SFLSR_REG

Address Offset	0x028		
Physical Address	0x4806 A028	**Instance**	UART1
	0x4806 C028		UART2
	0x4902 0028		UART3
	0x4809 E000		UART4
Description	Status FIFO line status register		
	IrDA modes only		
	Reading this register effectively reads frame status information from the status FIFO (this register does not physically exist). Reading this register increments the status FIFO read pointer (SFREGL_REG and SFREGH_REG must be read first).		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved			OE_ERROR	FTL_ERROR	ABORT_DETECT	CRC_ERROR	Reserved

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:5	Reserved		R	0x0
4	OE_ERROR		R	-
		0x1: Overrun error in RX FIFO when frame at top of RX FIFO was received. Top of RX FIFO = next frame to be read from RX FIFO.		
3	FTL_ERROR	Frame-too-long error	R	-
		0x1: Frame-length too long error in frame at top of RX FIFO		
2	ABORT_DETECT		R	-
		0x1: Abort pattern detected in frame at top of RX FIFO		
		0x1: CRC error in frame at top of RX FIFO		
1	CRC_ERROR		R	-
0	Reserved	Read returns 0.	R	0

14.6.2.24 RESUME_REG

Table 14-73. RESUME_REG

Address Offset	0x02C		
Physical Address	0x4806 A02C	**Instance**	UART1
	0x4806 C02C		UART2
	0x4902 002C		UART3
	0x4809 E000		UART4
Description	IR-IrDA and IR-CIR modes only		
	This register is used to clear internal flags, which halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and always reads as 0x00.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RESUME							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00	R	0x00
7:0	RESUME	Dummy read to restart the TX or RX	R	0x00

14.6.2.25 TXFLH_REG

Table 14-74. TXFLH_REG

Address Offset	0x02C		
Physical Address	0x4806 A02C	**Instance**	UART1
	0x4806 C02C		UART2
	0x4902 002C		UART3
	0x4809 E000		UART4
Description	Transmit frame length register low		
	IrDA modes only		
	The registers TXFLL_REG and TXFLH_REG hold the 13-bit transmit frame length (expressed in bytes). TXFLL_REG holds the LSBs and TXFLH_REG holds the MSBs. The frame length value is used if the frame length method of frame closing is used.		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved			TXFLH				

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:5	Reserved		R	0x0
4:0	TXFLH	MSB register used to specify the frame length	W	0x00

14.6.2.26 RXFLL_REG

Table 14-75. RXFLL_REG

Address Offset	0x030		
Physical Address	0x4806 A030	**Instance**	UART1
	0x4806 C030		UART2
	0x4902 0030		UART3
	0x4809 E000		UART4
Description	Received frame length register low		
	IrDA modes only		
	The registers RXFLL_REG and RXFLH_REG hold the 12-bit receive maximum frame length. RXFLL_REG holds the LSBs, and RXFLH_REG holds the MSBs. If the intended maximum receive frame length is n bytes, program RXFLL_REG and RXFLH_REG to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag).		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RXFLL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no functional effect.	W	0x00
7:0	RXFLL	LSB register used to specify the frame length in reception	W	0x00

14.6.2.27 SFREGL_REG

Table 14-76. SFREGL_REG

Address Offset	0x030		
Physical Address	0x4806 A030	**Instance**	UART1
	0x4806 C030		UART2
	0x4902 0030		UART3
	0x4809 E000		UART4
Description	Status FIFO register low		
	IrDA modes only		
	The frame lengths of received frames are written into the status FIFO. This information can be read by reading the SFREGL_REG and SFREGH_REG registers (these registers do not physically exist). The LSBs are read from SFREGL_REG, and the MSBs are read from SFREGH_REG. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR_REG.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SFREGL							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	SFREGL	LSB part of the frame length	R	0x–

14.6.2.28 SFREGH_REG

Table 14-77. SFREGH_REG

Address Offset	0x034		
Physical Address	0x4806 A034	**Instance**	UART1
	0x4806 C034		UART2
	0x4902 0034		UART3
	0x4809 E000		UART4
Description	Status FIFO register high		
	IrDA modes only		
	The frame lengths of received frames are written into the status FIFO. This information can be read by reading the SFREGL_REG and SFREGH_REG registers (these registers do not physically exist). The LSBs are read from SFREGL_REG, and the MSBs are read from SFREGH_REG. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR_REG.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved				SFREGH			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:4	Reserved	Read returns 0x0.	R	0x0
3:0	SFREGH	MSB part of the frame length	R	0x-

14.6.2.29 RXFLH_REG

Table 14-78. RXFLH_REG

Address Offset	0x034		
Physical Address	0x4806 A034	**Instance**	UART1
	0x4806 C034		UART2
	0x4902 0034		UART3
	0x4809 E000		UART4
Description	Received frame length register high		
	IrDA modes only		
	The registers RXFLL_REG and RXFLH_REG hold the 12-bit receive maximum frame length. RXFLL_REG holds the LSBs, and RXFLH_REG holds the MSBs. If the intended maximum receive frame length is n bytes, program RXFLL_REG and RXFLH_REG to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; there are two bytes associated with the FIR stop flag).		
Type	W		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved				RXFLH			

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Write has no functional effect.	W	0x00
7:4	Reserved	Write has no functional effect.	W	0x0
3:0	RXFLH	MSB register used to specify the frame length in reception	W	0x0

14.6.2.30 BLR_REG

Table 14-79. BLR_REG

Address Offset	0x038		
Physical Address	0x4806 A038	**Instance**	UART1
	0x4806 C038		UART2
	0x4902 0038		UART3
	0x4809 E000		UART4
Description	BOF control register		
	IrDA modes only		
	BLR_REG[6] is used to select whether 0xC0 or 0xFF start patterns are to be used, when multiple start flags are required in SIR mode. If only one start flag is required, this is always 0xC0. If n start flags are required, either (–1) 0xC0 or (–1) 0xFF flags are sent, followed by a single 0xC0 flag (immediately preceding the first data byte).		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								STS_FIFO_RESET	XBOF_TYPE	Reserved					

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	STS_FIFO_RESET	Status FIFO reset. This bit is self-clearing.	RW	0
6	XBOF_TYPE	SIR xBOF select 0x0: 0xFF 0x1: 0xC0	RW	1
5:0	Reserved	Read returns 0x00.	R	0x00

14.6.2.31 UASR_REG

Table 14-80. UASR_REG

Address Offset	0x038		
Physical Address	0x4806 A038	**Instance**	UART1
	0x4806 C038		UART2
	0x4902 0038		UART3
	0x4809 E000		UART4
Description	UART autobauding status register		
	UART autobauding mode only		
	This status register returns the speed, the number of bits by characters, and the type of parity in UART autobauding mode.		
	In autobauding mode, the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in incorrect baud rate recognition.		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PARITY_TYPE		BIT_BY_CHAR	SPEED				

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:6	PARITY_TYPE		R	0x0
		0x0: No parity identified		
		0x1: Parity space		
		0x2: Even parity		
		0x3: Odd parity		
5	BIT_BY_CHAR		R	0
		0x0: 7-bit character identified		
		0x1: 8-bit character identified		
4:0	SPEED	Used to report the speed identified	R	0x00
		0x0: No speed identified		
		0x1: 115 200 baud		
		0x2: 57 600 baud		
		0x3: 38 400 baud		
		0x4: 28 800 baud		
		0x5: 19 200 baud		
		0x6: 14 400 baud		
		0x7: 9 600 baud		
		0x8: 4 800 baud		
		0x9: 4 800 baud		
		0xA: 1 200 baud		

14.6.2.32 ACREG_REG

Table 14-81. ACREG_REG

Address Offset	0x03C		
Physical Address	0x4806 A03C	**Instance**	UART1
	0x4806 C03C		UART2
	0x4902 003C		UART3
	0x4809 E000		UART4
Description	Auxiliary control register		
	IrDA-CIR mode only		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PULSE_TYPE	SD_MOD	DIS_IR_RX	DIS_TX_UNDERRUN	SEND_SIP	SCTX_EN	ABORT_EN	EOT_EN

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	PULSE_TYPE	SIR pulse-width select: 0x0: 3/16 of baud-rate pulse width 0x1: 1.6 µs	RW	0
6	SD_MOD	Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers. 0x0: SD pin is set to high. 0x1: SD pin is set to low.	RW	0
5	DIS_IR_RX	0x0: Normal operation (RX input automatically disabled during transmit, but enabled outside of transmit operation). 0x1: Disables RX input (permanent state; independent of transmit)	RW	0
4	DIS_TX_ UNDERRUN	0x0: Long stop bits cannot be transmitted. TX underrun is enabled. 0x1: Long stop bits can be transmitted. TX underrun is disabled.	RW	0
3	SEND_SIP	MIR/FIR modes only. Send serial infrared interaction pulse (SIP). If this bit is set during an MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission. 0x0: No action 0x1: Send SIP pulse.	RW	0
2	SCTX_EN	Store and control TX start. When MDR1_REG[5] = 1 and the MPU writes 1 to this bit, the TX state-machine starts frame transmission. This bit is self-clearing.	RW	0
1	ABORT_EN	Frame abort. The MPU can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame.	RW	0

Bits	Field Name	Description	Type	Reset
0	EOT_EN	EOT (end-of-transmission) bit. The MPU writes 1 to this bit just before it writes the last byte to the TX FIFO in the set-EOT bit frame-closing method. This bit is automatically cleared when the MPU writes to the THR_REG (TX FIFO).	RW	0

14.6.2.33 SCR_REG

Table 14-82. SCR_REG

Address Offset	0x040	
Physical Address	0x4806 A040	UART1
	0x4806 C040	UART2
	0x4902 0040	UART3
	0x4809 E000	UART4
Description	Supplementary control register	
Type	RW	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RX_TRIG_GRANU1	TX_TRIG_GRANU1	Reserved	RX_CTS_WU_EN	TX_EMPTY_CTL_IT	DMA_MODE_2		DMA_MODE_CTL

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	RX_TRIG_GRANU1		RW	0
		0x0: Disables the granularity of 1 for TRIGGER RX level		
		0x1: Enables the granularity of 1 for TRIGGER RX level		
6	TX_TRIG_GRANU1		RW	0
		0x0: Disables the granularity of 1 for TRIGGER TX level		
		0x1: Enables the granularity of 1 for trigger TX level		
5	Reserved	Read returns 0. Write has no functional effect.	RW	0
4	RX_CTS_WU_EN	RX CTS wake-up enable	RW	0
		0x0: Disables the WAKE UP interrupt and clears SSR_REG[1]		
		0x1: Waits for a falling edge of pins RX, nCTS, or nDSR to generate an interrupt		
3	TX_EMPTY_CTL_IT		RW	0
		0x0: Normal mode for THR interrupt (see Table 14-30 for details about UART mode interrupts)		
		0x1: The THR interrupt is generated when TX FIFO and TX shift register are empty.		
2:1	DMA_MODE_2	Specifies the DMA mode valid if SCR_REG[0] = 1	RW	0x0
		0x0: DMA mode 0 (no DMA)		
		0x1: DMA mode 1 (UARTi_DMA_TX, UARTi_DMA_RX)		
		0x2: DMA mode 2 (UARTi_DMA_RX)		
		0x3: DMA mode 3 (UARTi_DMA_TX)		
0	DMA_MODE_ CTL		RW	0
		0x0: The DMA_MODE is set with FCR_REG[3].		
		0x1: The DMA_MODE is set with SCR_REG[2:1].		

14.6.2.34 SSR_REG

Table 14-83. SSR_REG

Address Offset	0x044		
Physical Address	0x4806 A044	**Instance**	UART1
	0x4806 C044		UART2
	0x4902 0044		UART3
	0x4809 E000		UART4
Description	Supplementary status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved						RX_CTS_WU_STS	TX_FIFO_FULL

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:2	Reserved	Read returns 0x00.	R	0x00
1	RX_CTS_WU_STS	Pin falling edge detection: Reset only when SCR_REG[4] is reset to 0. 0x0: No falling-edge event on RX, nCTS, and nDSR 0x1: A falling edge occurred on RX, nCTS, or nDSR.	R	0
0	TX_FIFO_FULL	 0x0: TX FIFO is not full. 0x1: TX FIFO is full.	R	0

14.6.2.35 EBLR_REG

Table 14-84. EBLR_REG

Address Offset	0x048		
Physical Address	0x4806 A048	**Instance**	UART1
	0x4806 C048		UART2
	0x4902 0048		UART3
	0x4809 E000		UART4
Description	BOF length register.		
	IR-IrDA mode only		
	In IR-IrDA SIR operation, this register specifies the number of BOF + xBOFs to transmit. Value set into this register must consider the BOF character; therefore, to send only one BOF with no XBOF, this register must be set to 1. To send one BOF with N XBOF, this register must be set to N + 1. Furthermore, the value 0 sends 1 BOF plus 255 XBOF.		
	In IR-IrDA MIR mode, this register specifies the number of additional start flags (MIR protocol mandates a minimum of 2 start flags).		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EBLR							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	EBLR	IR-IrDA mode: This register allows definition of up to 176 xBOFs, the maximum required by IrDA specification. IR-CIR mode: N/A 0x00: Feature disabled 0x01: Generate RX_STOP interrupt after receiving one zero bit. 0xFF: Generate RX_STOP interrupt after receiving 255 zero bits.	RW	0x00

14.6.2.36 SYSC_REG

Table 14-85. SYSC_REG

Address Offset	0x054		
Physical Address	0x4806 A054	**Instance**	UART1
	0x4806 C054		UART2
	0x4902 0054		UART3
	0x4809 E000		UART4
Description	System configuration register.		
	The auto idle bit controls a power-saving technique to reduce the logic power consumption of the module interface; that is, when the feature is enabled, the interface clock is gated off until the module interface is accessed. When the software reset bit is set high, it causes a full device reset.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved			IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:5	Reserved	Read returns 0x0.	R	0x0
4:3	IDLEMODE	Power management req/ack control	RW	0x0
		0x0: Force idle: Idle request is acknowledged unconditionally.		
		0x1: No-idle: Idle request is never acknowledged.		
		0x2: Smart idle: Idle request is acknowledged based on the internal module activity.		
		0x3: Reserved		
2	ENAWAKEUP	Wakeup control	RW	0
		0x0: Wakeup is disabled.		
		0x1: Wakeup capability is enabled.		
1	SOFTRESET	Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. Read returns 0.	RW	0
		0x0: Normal mode		
		0x1: The module is reset.		
0	AUTOIDLE	Internal interface clock-gating strategy	RW	0
		0x0: Clock is running.		
		0x1: Automatic interface clock-gating strategy is applied based on interface activity.		

14.6.2.37 SYSS_REG

Table 14-86. SYSS_REG

Address Offset	0x058		
Physical Address	0x4806 A058	**Instance**	UART1
	0x4806 C058		UART2
	0x4902 0058		UART3
	0x4809 E000		UART4
Description	System status register		
Type	R		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved							RESETDONE

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:1	Reserved	Read returns 0x00.	R	0x00
0	RESETDONE	Internal reset monitoring	R	0
		0x0: Internal module reset is ongoing.		
		0x1: Reset complete		

14.6.2.38 WER_REG

Table 14-87. WER_REG

Address Offset	0x05C		
Physical Address	0x4806 A05C	**Instance**	UART1
	0x4806 C05C		UART2
	0x4902 005C		UART3
	0x4809 E000		UART4
Description	Wake-up enable register		
	The UART wake-up enable register is used to mask and unmask a UART event that subsequently notifies the system. An event is any activity in the logic that can cause an interrupt and/ or an activity that requires the system to wake up. Even if wakeup is disabled for certain events, if these events are also an interrupt to the UART, the UART still registers the interrupt as such.		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Reserved	EVENT_6_RLS_ INTERRUPT	EVENT_5_RHR_ INTERRUPT	EVENT_4_RX_ ACTIVITY	Reserved	EVENT_2_RI_ ACTIVITY	Reserved	EVENT_0_CTS_ ACTIVITY

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7	Reserved	Read returns 0.	R	0
6	EVENT_6_RLS_ INTERRUPT	Receiver line status interrupt	RW	1
		0x0: Event is not allowed to wake up the system.		
		0x1: Event can wake up the system.		
5	EVENT_5_RHR_ INTERRUPT		RW	-
		0x0: Event is not allowed to wake up the system.		
		0x1: Event can wake up the system.		
4	EVENT_4_RX_ ACTIVITY		RW	1
		0x0: Event is not allowed to wake up the system.		
		0x1: Event can wake up the system.		
3	Reserved	Read returns 1. Must be set to 1 for correct behavior.	RW	1
2	EVENT_2_RI_ ACTIVITY		RW	1
		0x0: Event is not allowed to wake up the system.		
		0x1: Event can wake up the system.		
1	Reserved	Read returns 1. Must be set to 1 for correct behavior.	RW	1
0	EVENT_0_CTS_ ACTIVITY	UART mode only	RW	1
		0x0: Event is not allowed to wake up the system.		
		0x1: Event can wake up the system.		

14.6.2.39 CFPS_REG

Table 14-88. CFPS_REG

Address Offset	0x060		
Physical Address	0x4806 A060	**Instance**	UART1
	0x4806 C060		UART2
	0x4902 0060		UART3
	0x4809 E000		UART4
Description	Carrier frequency prescaler		
Type	RW		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CFPS							

Bits	Field Name	Description	Type	Reset
15:8	Reserved	Read returns 0x00.	R	0x00
7:0	CFPS	Because the consumer IR works at modulation rates of 30-56.8 kHz, the 48-MHz clock must be prescaled before the clock can drive the IR logic. This register sets the divisor rate to give a range to accommodate remote-control requirements in BAUD multiples of 12x. The value of the CFPS at reset is 0105 (decimal), which equates to a 38.1-kHz output from starting conditions. The 48-MHz carrier is prescaled by the CFPS, which is then divided by the 12x BAUD multiple.	RW	0x69

Example:

Target Freq (kHz)	CFPS (decimal)	Actual Freq(kHz)
30	133	30.08
32.75	122	32.79
36	111	36.04
36.7	109	36.69
38	105	38.1
40	100	40
56.8	70	57.14

CFPS = 0 is not supported.

I^2C

This chapter describes the four high-speed I^2C controller modules.

Topic **Page**

15.1 High-Speed I²C Controller Overview

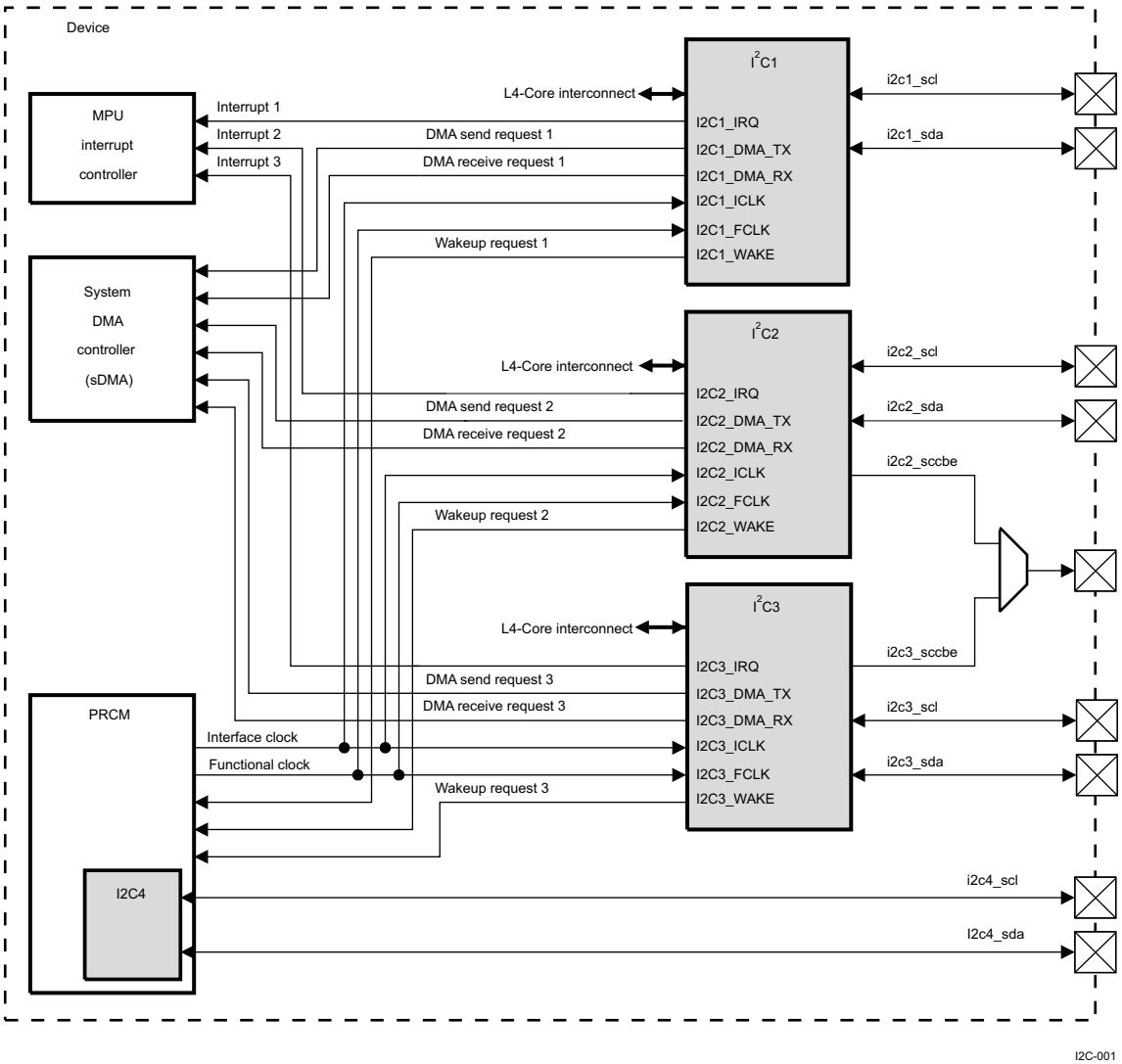
The device contains three multimaster high-speed (HS) inter-integrated circuit (I²C) controllers (I2Ci modules, where i = 1, 2, 3), each of which provides an interface between a local host (LH), such as the MPU subsystem, and any I²C-bus-compatible device that connects through the I²C serial bus. External components attached to the I²C bus can serially transmit/receive up to 8 bits of data to/from the LH device through the 2-wire I²C interface.

Each multimaster HS I²C controller can be configured to act like a slave or master I²C-compatible device. Moreover, each multimaster HS I²C controller can be configured in serial camera control bus (SCCB) mode (the SCCB is a serial bus developed by Omnivision Technologies, Inc.) to act as a master on a 2-wire SCCB bus. Only multimaster HS I²C controllers I2C2 and I2C3 can be configured in SCCB mode to act as a master device on a 3-wire SCCB bus.

The device contains an additional master transmitter HS I²C interface (I2C4) in the power, reset, and clock management (PRCM) module to perform dynamic voltage control and power sequencing. Texas Instruments provides a global solution with the device connected to power chips.

Figure 15-1 shows the HS I²C controllers in the device.

Figure 15-1. HS I²C Controllers



The three multimaster HS I²C controllers have the following features:

- Compliance with Philips I²C specification version 2.1

- Support for standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s)
- Support for HS mode for transfer up to 3.4M bits/s
- Support for 3-wire/2-wire SCCB master mode for I2C2 and I2C3 modules, 2-wire SCCB master mode for I2C1 module, up to 100K bits/s
- 7-bit and 10-bit device addressing modes
- General call
- Start/restart/stop
- Multimaster transmitter/slave receiver mode
- Multimaster receiver/slave transmitter mode
- Combined master transmit/receive and receive/transmit mode
- Built-in FIFO for buffered read or write
  - 8 bytes for I2C1 and I2C2
  - 64 bytes for I2C3
- Module enable/disable capability
- Programmable clock generation
- 8-bit-wide data access
- Low-power consumption design
- Two DMA channels
- Wide interrupt capability

The master transmitter HS I²C controller I2C4 has the following features:

- Support of HS and fast modes
- 7-bit addressing mode only
- Master transmitter mode only
- Start/restart/stop

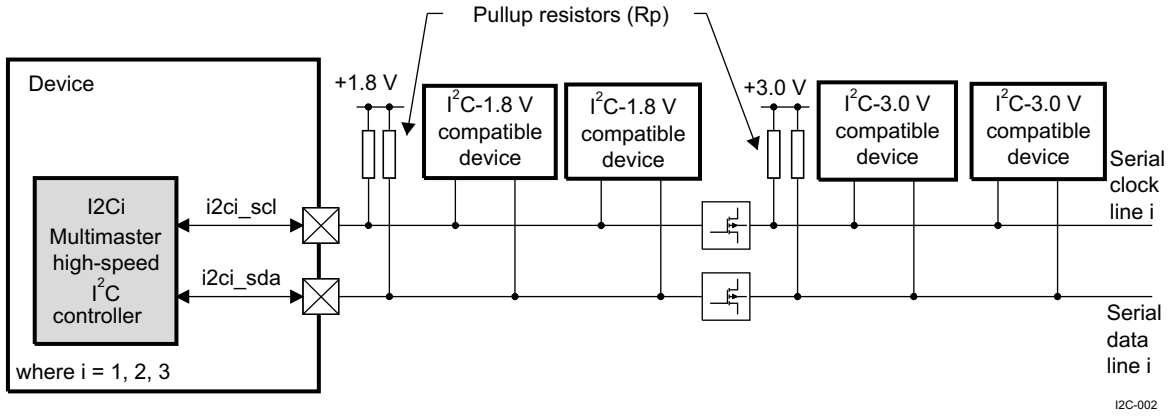
NOTE: Before using HS I²C mode, determine that the target device supports this mode.

15.2 High-Speed I²C Controller Environment

15.2.1 Multimaster HS I²C Controllers in I²C Mode

Figure 15-2 shows the multimaster HS I²C controllers and their related connections with I²C-compliant devices in I²C mode.

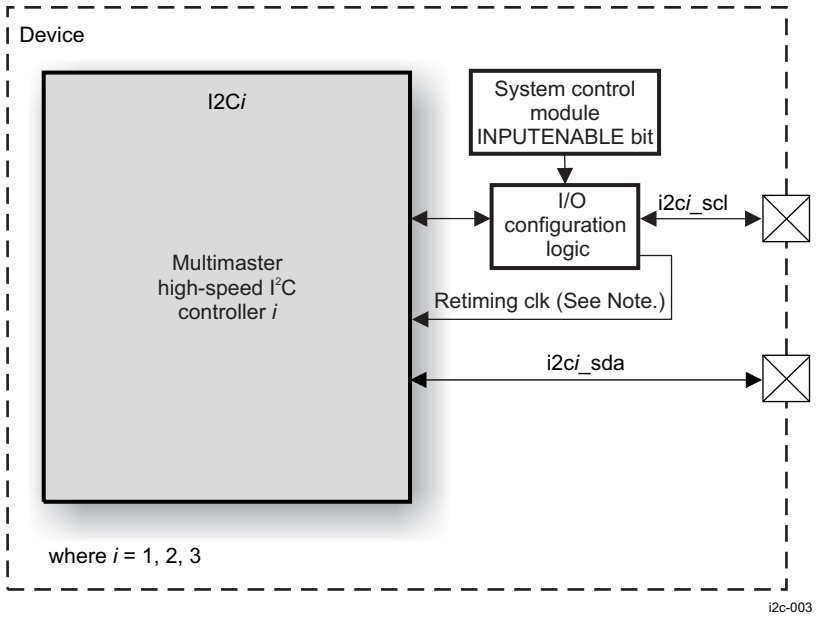
Figure 15-2. Multimaster HS I²C Controllers and Typical Connections to I²C Devices



15.2.1.1 Multimaster HS I²C Controller Pins for Typical Connections in I²C Mode

Figure 15-3 shows the multimaster HS I²C controller pins used for typical connections with I²C devices.

Figure 15-3. Multimaster HS I²C Controller Interface Signals in I²C Mode



NOTE: In master mode, the clock signal (IP clk configured as output) is also used as retiming input (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).

15.2.1.2 I²C Interface Typical Connections

Table 15-1 lists the pins associated with the I²C interface.

Table 15-1. Input/Output

Signal	I/O[1]	Description	Reset Value
i2ci_scl	I/O(OD)	I²C serial clock line[2]. Open-drain output buffer. Requires external pullup resistor (Rp).	Hi-Z
i2ci_sda	I/O(OD)	I²C serial data line. Open-drain output buffer. Requires external Rp.	Hi-Z

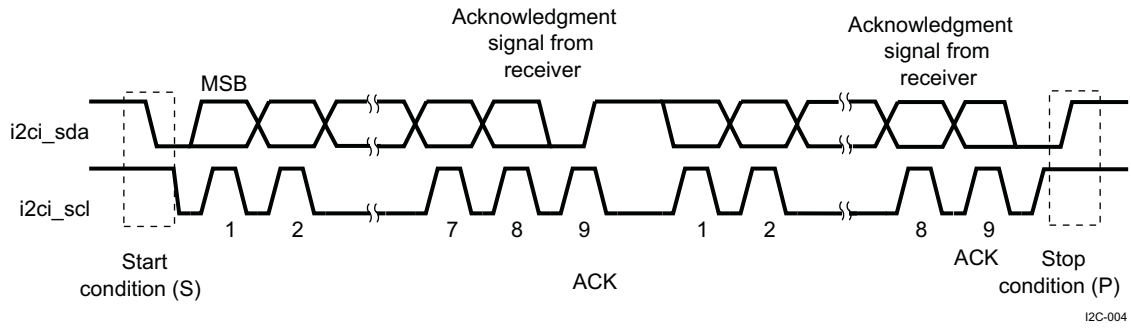
(1) I = Input; O = Output; OD = Open Drain; Hi-Z = High Impedance
(2) This signal is also used as retiming input (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).

15.2.1.3 I²C Typical Connection Protocol and Data Format

15.2.1.3.1 Serial Data Format

The I²C controller operates in 8-bit word data format (byte write access supported for the last access). Each byte transmitted or received on the serial data line (i2ci_sda) is 8 bits long. The number of bytes that can be transmitted or received is not restricted. The data is transferred with the most-significant bit (MSB) first. In receiver mode, each byte is followed by an acknowledge bit from the I²C. Figure 15-4 shows a typical I²C communication format.

Figure 15-4. I²C Data Transfer

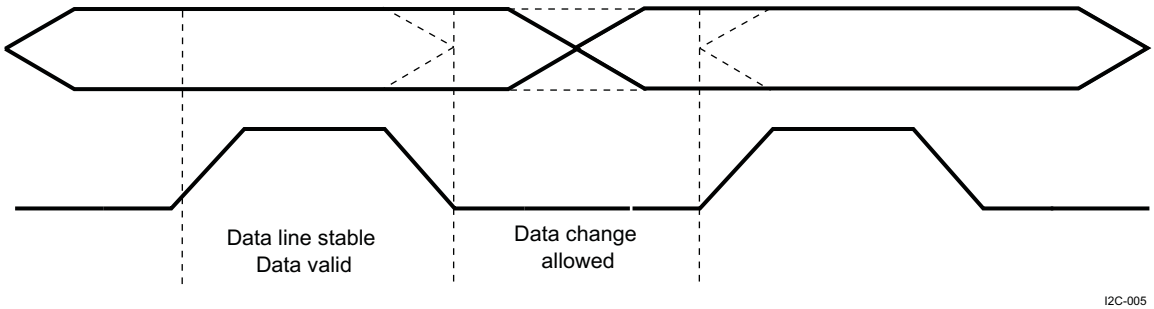


15.2.1.3.2 Data Validity

The data on the serial data line must be stable during the high period of the clock i2ci_scl. The high and low states of the data line can change only when the clock signal on the serial clock line is low.

Figure 15-5 is an example of data validity requirements.

Figure 15-5. Bit Transfer on the I²C Bus



15.2.1.3.3 Start and Stop Conditions

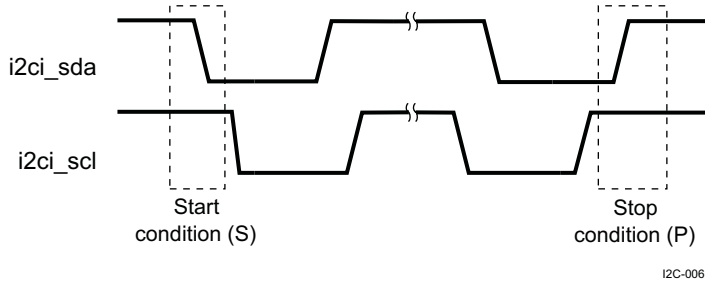
The I²C module generates start (S) and stop (P) conditions when it is configured as a master.

- An S condition is a high-to-low transition on the i2ci_sda line while i2ci_scl is high.
- A P condition is a low-to-high transition on the i2ci_sda line while i2ci_scl is high.

The bus is considered busy after the S condition (the I2Ci.I2C_STAT[12] BB bit is 1 to indicate that the bus is busy) and free after the P condition (the I2Ci.I2C_STAT[12] BB bit is 0 to indicate that the bus is free).

Figure 15-6 shows the waveforms that occur during an S and a P condition.

Figure 15-6. S and P Condition Events



15.2.1.3.4 Addressing

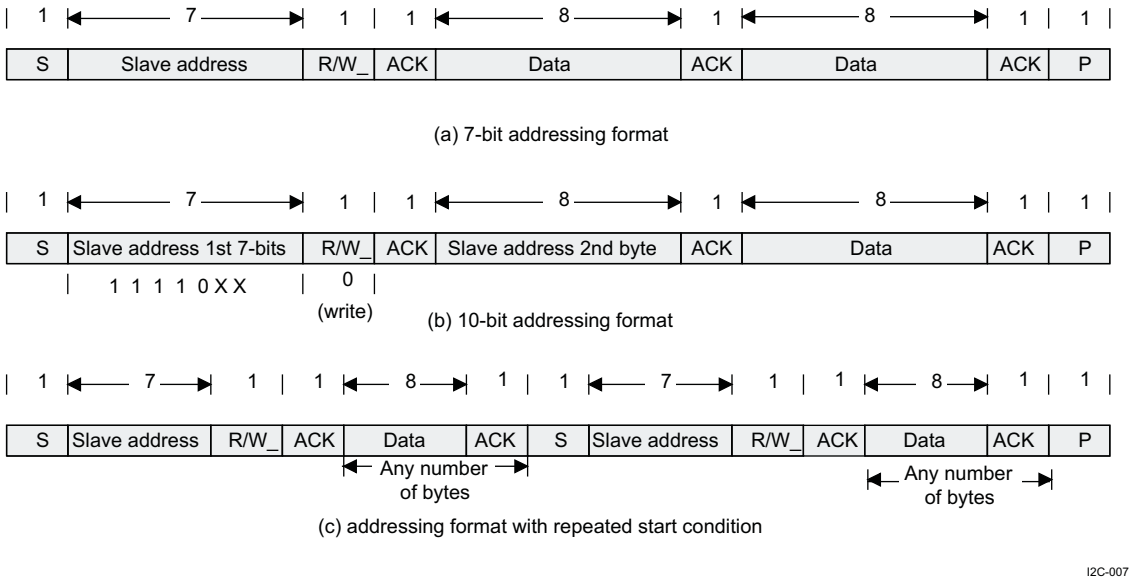
The I²C module supports two data formats in fast/standard (F/S) and HS modes:

- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start (Sr) condition

15.2.1.3.4.1 Data Transfer Format in F/S Mode

Figure 15-7 shows the I²C data transfer format in F/S mode.

Figure 15-7. I²C Data Transfer Formats in F/S Mode



The first word after a S condition consists of 8 bits. In acknowledge mode, an extra dedicated acknowledgment bit is inserted after each byte.

In addressing formats with 7-bit addresses, the first byte is composed of 7 MSB slave address bits and 1 least-significant bit (LSB) R/W_ bit.

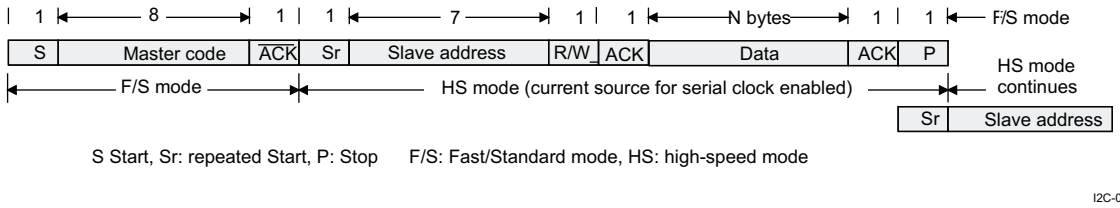
The LSB R/W_ bit of the address byte indicates the transmission direction of the data bytes that follow it. If R/W_ is 0, the master writes data to the selected slave; if it is 1, the master reads data from the slave.

In addressing formats with 10-bit addresses, the structure of the first byte is 11110XXY, where XX is the two MSBs of the 10-bit addresses, and Y is the R/W_ bit. If the R/W_ bit is 0, the next byte contains the last 8 bits of the slave address. If the R/W_ bit is 1, the next byte contains data transmitted from the slave to the master.

15.2.1.3.4.2 Data Transfer Format in HS Mode

Figure 15-8 shows the I²C data transfer format in HS mode.

Figure 15-8. I²C Data Transfers in HS Mode



Each multimaster HS I²C controller module can also operate in HS mode. In this case, after the S condition, the module, which is in F/S mode, writes the master code address (000001XXX, where XXX is the variable portion of the master code) on the bus. No device connected on the same bus acknowledges this address. The module switches the clock to the HS clock and after an Sr condition, and sends the slave address and the data, as shown in Figure 15-8.

NOTE: For more information, see *I²C Bus Specification v2.1*, January 2000.

15.2.1.3.5 Master Transmitter

In master transmitter mode, data assembled in one of the previously described data formats is shifted out on serial data line i2ci_sda in synchronization with the self-generated clock pulses on serial clock line i2ci_scl. When the intervention of the processor is required (I2Ci.I2C_STAT[10] XUDF bit) after a byte is transmitted, the clock pulses are inhibited and the serial clock line is held low.

15.2.1.3.6 Master Receiver

Master receiver mode can be entered only from master transmitter mode. With any of the address formats (a), (b), or (c) (see Figure 15-7), if R/W_ is high, the module enters master receiver mode after the slave address byte and bit R/W_ are transmitted. Serial data bits received on bus line i2ci_sda are shifted in synchronization with the self-generated clock pulses on i2ci_scl. When the intervention of the processor is required (I2Ci.I2C_STAT[11] ROVR bit) after a byte is transmitted, the clock pulses are inhibited and the serial clock line is held low. At the end of a transfer, a P condition is generated.

15.2.1.3.7 Slave Transmitter

Slave transmitter mode can be entered only from slave receiver mode. With any of the address formats (a), (b), or (c) (see Figure 15-7), slave transmitter mode is entered if the slave address byte is the same as its own address, and when the R/W_ bit transmitted by the master transmitter is high. The slave transmitter shifts the serial data out on data line i2ci_sda in synchronization with the clock pulses generated by the master device. It does not generate the clock, but it can hold clock line i2ci_scl low when intervention of the LH is required (the I2Ci.I2C_STAT[10] XUDF bit).

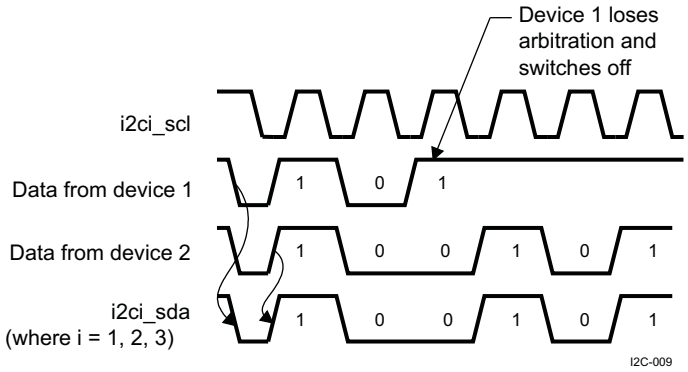
15.2.1.3.8 Slave Receiver

In slave receiver mode, serial data bits received on bus line i2ci_sda are shifted in in synchronization with the clock pulses on i2ci_scl generated by the master device. The slave receiver does not generate the clock, but it can hold the serial clock line low when intervention of the LH is required (the I2Ci.I2C_STAT[11] ROVR bit) after a byte is received.

15.2.1.3.9 Bus Arbitration

If two or more master transmitters start a transmission on the same bus simultaneously, an arbitration procedure is invoked. This arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to slave receiver mode, sets the arbitration lost flag (I2Ci.I2C_STAT[0] AL bit), and generates the arbitration lost interrupt. Figure 15-9 shows arbitration between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. If two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

Figure 15-9. Arbitration Between Master Transmitters



15.2.1.3.10 I²C Clock Generation and Synchronization

Under normal conditions, only one master device generates the clock signal i2ci_scl. During arbitration, however, there are two or more master devices, and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that the device that first generates a low period of the clock line overrules the other devices. At this high-low transition, the clock generators of the other devices are forced to start generating their own low periods. The clock line is then held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus 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, all clock generators must enter the wait state. In this way, a slave can slow down a fast master, and the slow device can create enough time to store a received byte or prepare a byte to be transmitted (Clock Stretching).

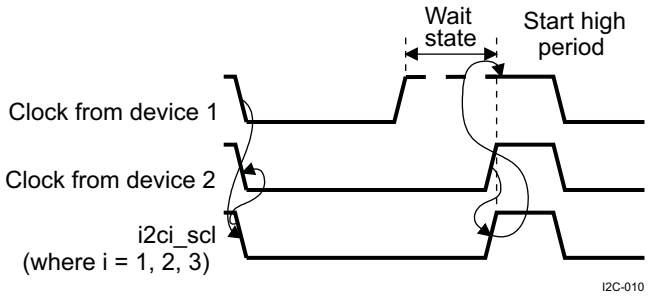
NOTE: In case the SCL or SDA lines are stuck low, the Bus Clear operation is supported:

If the clock line (SCL) is stuck low, the preferred procedure is to reset the bus using the hardware reset signal if your I²C devices have hardware reset inputs. If the I²C devices do not have hardware reset inputs, cycle power to the devices to activate the mandatory internal power-on reset (POR) circuit.

If the data line (SDA) is stuck low, the master should send nine clock pulses. The device that held the bus low should release it sometime within those nine clocks. If not, then use the hardware reset or cycle power to clear the bus.

Figure 15-10 shows clock synchronization.

Figure 15-10. Synchronization of I²C Clock Generators



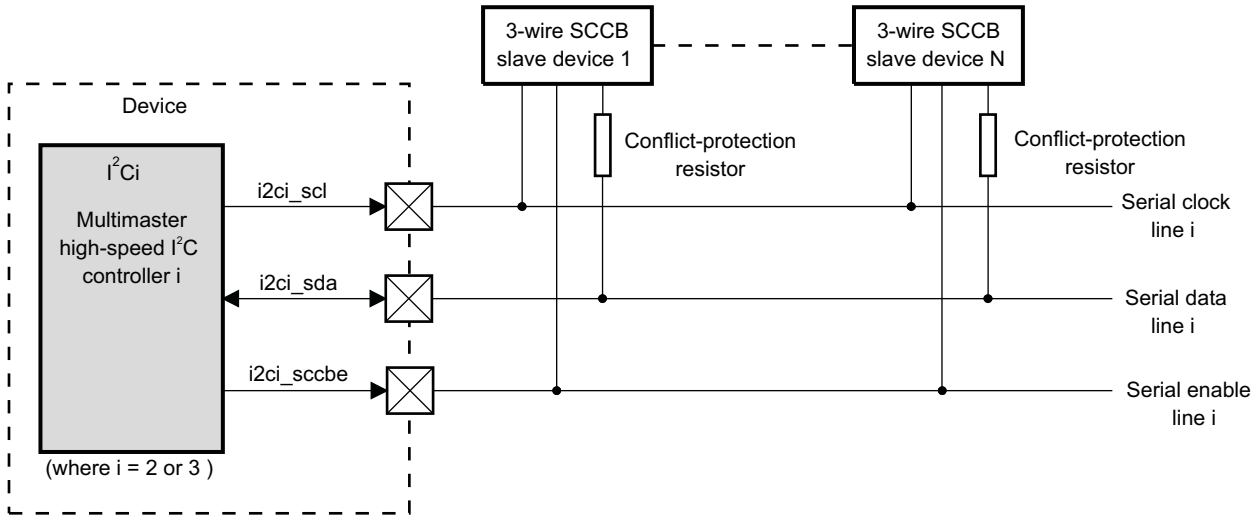
15.2.2 Multimaster High-Speed I²C Controllers in SCCB Mode

The SCCB is a 3-wire serial bus developed by Omnivision Technologies, Inc. The SCCB can also operate in a modified 2-wire serial mode. For details, see the SCCB specifications version 2.1 document at *http://www.ovt.com/pdfs/ds_note.pdf*.

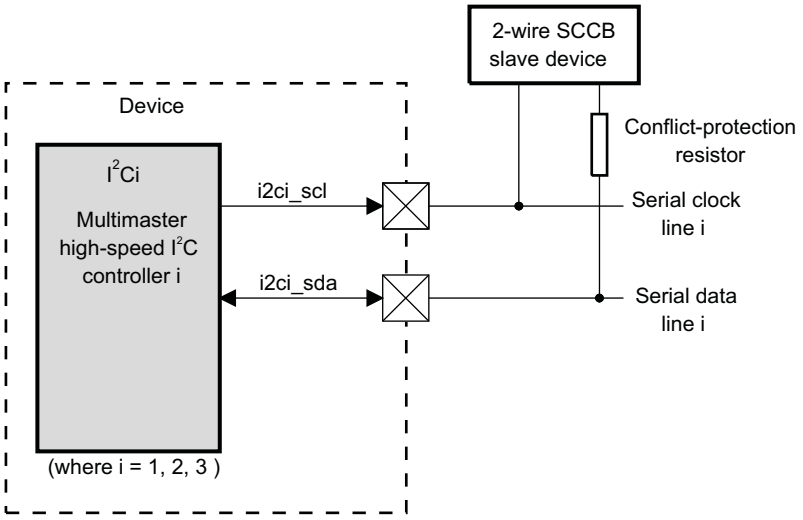
The multimaster HS I²C controllers support the 2-wire SCCB protocol in master mode. Only the I2C2 and I2C3 modules support the 3-wire SCCB protocol in master mode.

Figure 15-11 shows the multimaster HS I²C controllers and their related connections with 3-wire or 2-wire SCCB-compliant devices.

Figure 15-11. Multimaster HS I²C Controllers and Typical Connections to SCCB Devices



(a) Typical connection with 3-wire SCCB devices



(b) Typical connection with 2-wire SCCB devices

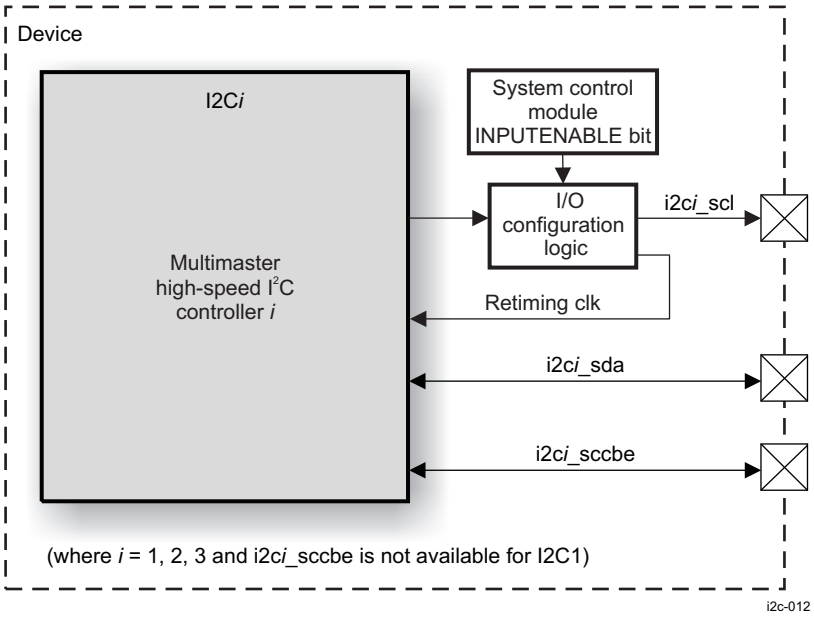
I2C-011

NOTE: Only one 2-wire SCCB slave device can be connected to the 2-wire SCCB bus.

15.2.2.1 Multimaster HS I²C Controller Pins for Typical Connections in SCCB Mode

Figure 15-12 shows the multimaster HS I²C controller pins used for typical connections with 3-wire or 2-wire SCCB devices.

Figure 15-12. Multimaster HS I²C Controller Interface Signals in SCCB Mode



15.2.2.2 SCCB Interface Typical Connections

Table 15-2 lists the pins associated with the SCCB interface.

Table 15-2. Input/Output

Signal	I/O[1]	Description	Reset Value	I2Ci.I2C_CON[15] I2C_EN bit = 0
i2ci_scl	O	SCCB serial clock line[2]. Standard CMOS output buffer.	Hi-Z	High
i2ci_sda	I/O(OD)	SCCB serial data line. Standard CMOS 3-state output buffer. Requires external conflict-protection resistor for each slave device connected to the bus.	Hi-Z	Hi-Z
i2ci_sccbe [3]	O	SCCB enable line. Standard CMOS output buffer.	High	High

(1) I = Input; O = Output; OD = Open Drain; Hi-Z = High Impedance
(2) This output signal is also used as retiming input (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).
(3) This signal is used for the 3-wire SCCB protocol only.

NOTE: Because they share the same ball, the i2c2_sccbe and i2c3_sccbe signals are not available at the same time. For detailed information about pin configuration, see Chapter 6, *System Control Module*.

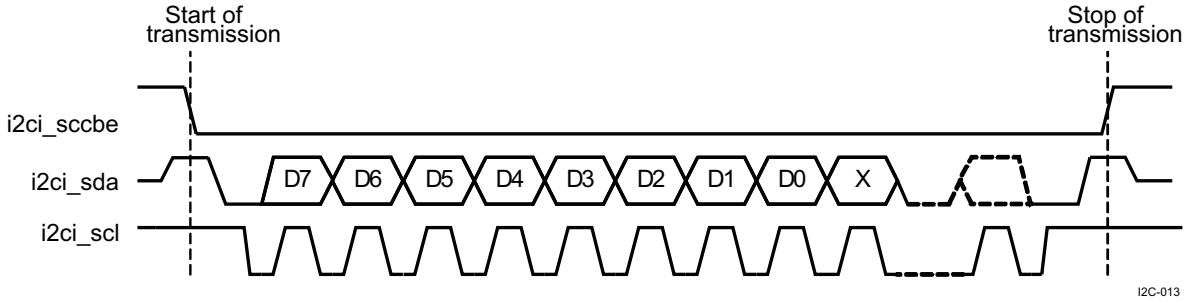
The I2C1 module does not provide any i2c1_sccbe signal at the chip boundary of the device; thus, this module does not support the 3-wire SCCB protocol

15.2.2.3 SCCB Typical Connection Protocol and Data Format

15.2.2.3.1 Serial Transmission Timing Diagram

Figure 15-13 is the timing diagram of the 3-wire SCCB data transmission.

Figure 15-13. 3-wire SCCB Transmission Timing Diagram

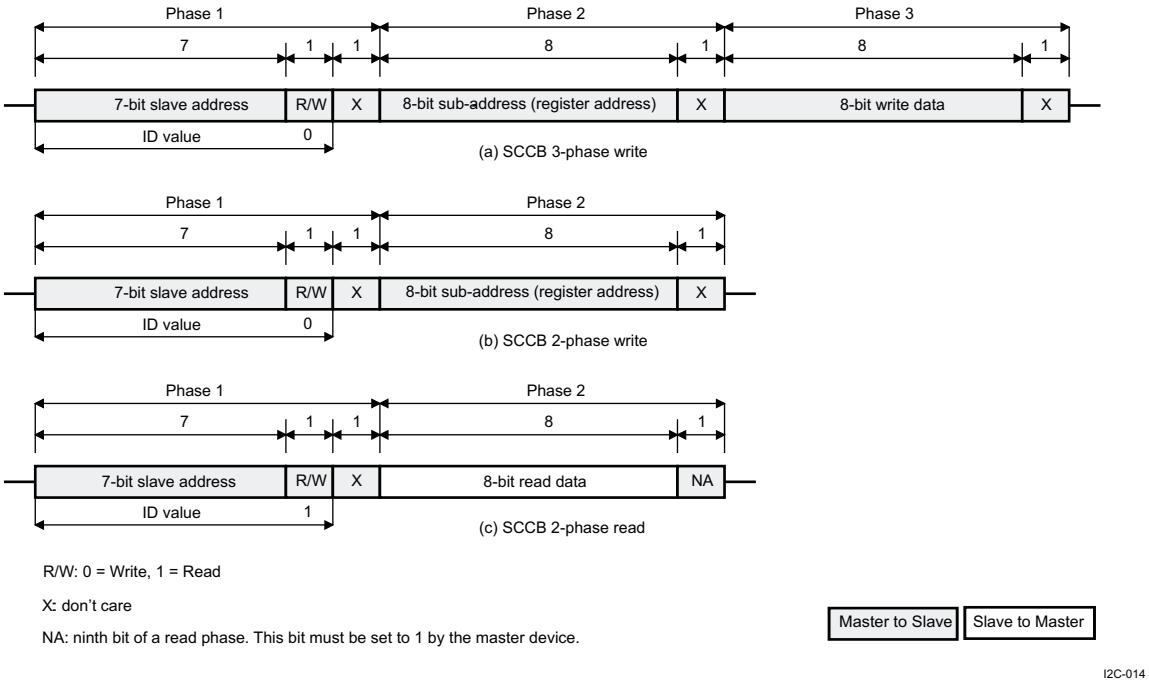


NOTE: When operating in 2-wire SCCB mode, the i2ci_sccbe signal is not used by the 2-wire SCCB-compliant slave device attached to the 2-wire SCCB bus.

15.2.2.3.2 SCCB Transmission Data Formats

Figure 15-14 describes the data format of the three kinds of transmission.

Figure 15-14. SCCB Transmission Data Formats



The basic element of a data transmission is a phase. A phase contains 9 bits, which consist of an 8-bit sequential data transmission followed by a ninth bit. The ninth bit is a don't-care bit (X) or an NA bit, depending on whether the data transmission is a write or a read. The maximum number of phases that can be included in a transmission is three. The MSB is always asserted first for each phase.

A data transmission is one of three types:

- **3-phase write transmission**: The 3-phase write transmission cycle (see (a) of Figure 15-14) is a full write operation in which the master can write 1 byte of data to a specific slave(s). The 7-bit slave

address in the ID value identifies the specific slave that the master intends to access. The subaddress identifies the register location of the specified slave. The write data contains 8-bit data that the master intends to write over the content of this specific address. The ninth bit of each of the three phases is a don't-care bit (X bit).

- **2-phase write transmission**: The 2-phase write transmission cycle is followed by a 2-phase read transmission cycle (see below). The purpose of issuing a 2-phase write transmission cycle (see (b) of Figure 15-14) is to identify the subaddress of some specific slave from which the master intends to read data for the following 2-phase read transmission cycle. The ninth bit of each phase is a don't-care bit (X bit).
- **2-phase read transmission**: Either a 3-phase or a 2-phase write transmission cycle must be asserted ahead of a 2-phase read transmission cycle. The 2-phase read transmission cycle (see (c) of Figure 15-14) has no ability to identify the subaddress. The 2-phase write transmission cycle contains read data of 8 bits and a ninth don't-care bit or NA bit. The master must drive the NA bit at logical 1.

In each transmission type, phase 1 (7-bit slave address of the ID value) is asserted by the master to identify the selected slave to which the data is read or written. Each slave has a unique 7-bit slave address. The 7-bit slave address of the ID value comprises 7 bits, from bit 7 to bit 1, that can identify up to 128 slaves. The eighth bit, bit 0, is the read/write selector bit that specifies the transmission direction of the current cycle. A logical 0 represents a write cycle and a logical 1 represents a read cycle. The ninth bit of phase 1 is a don't-care bit (X bit).

Phase 2 (subaddress/read data) is asserted by the master (subaddress) or the slave(s) (read data). A phase 2 transmission asserted by the master identifies the subaddress of the slave(s) the master intends to access. A phase 2 transmission asserted by the slave(s) indicates the read data that the master will receive. The slave(s) recognize the subaddress of this read data according to the previous 3-phase or 2-phase write transmission cycle. The ninth bit is defined as a don't-care bit (X bit) when the master asserts phase 2. The ninth bit is defined as an NA bit when the slave(s) asserts the phase 2 transmission. The master is responsible for a logical 1 during the period of the NA bit.

Phase 3 (write data) is asserted only by the master. This phase contains the data the master intends to write to the slave(s). Because the master asserts the transmission, the ninth bit of the phase 3 transmission is defined as a don't-care bit (X bit).

NOTE: A multimaster HS I²C controller configured in SCCB mode can perform two operations:

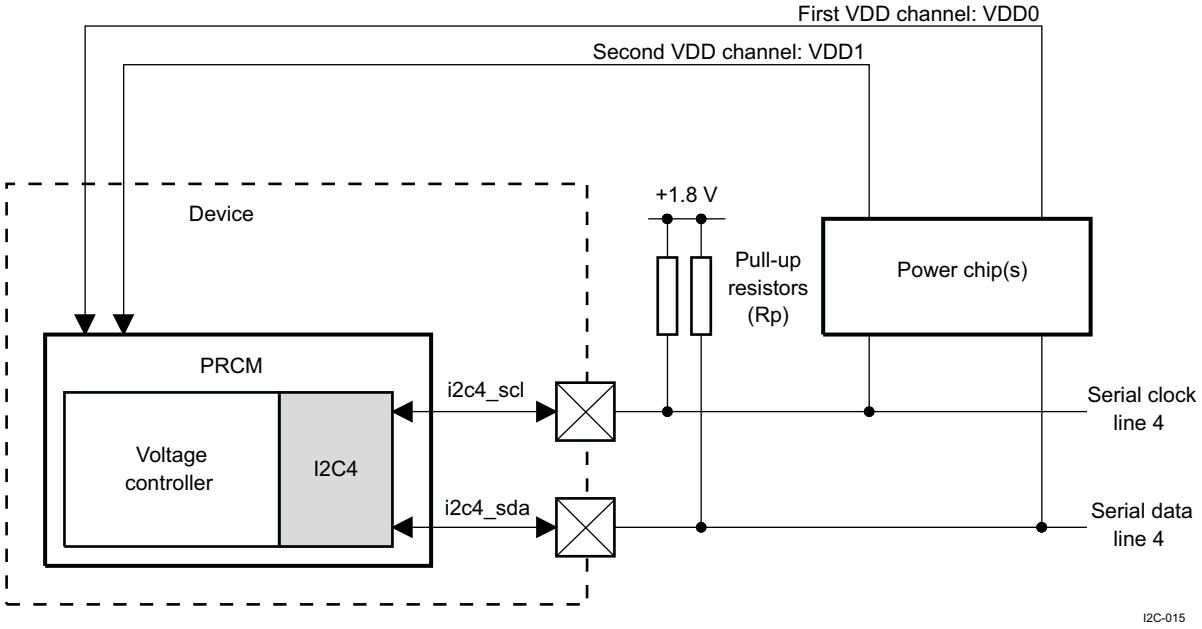
- Writing a single byte to an SCCB slave device by using the 3-phase write transmission cycle
- Reading a single byte from an SCCB slave device by using the 2-phase write transmission cycle followed by the 2-phase read transmission cycle

15.2.3 High-Speed I²C Controller for Communication With Power Chip(s)

The master transmitter HS I²C controller I2C4 interfaces between the and external power chip(s) for voltage control. This module is always configured as an I²C master transmitter; it does not support the SCCB protocol. For a definition of master transmitter mode, see Section 15.2.1.3.5, *Master Transmitter*.

Figure 15-15 shows a typical connection between the master transmitter HS I²C controller I2C4 of the device and external power chip(s).

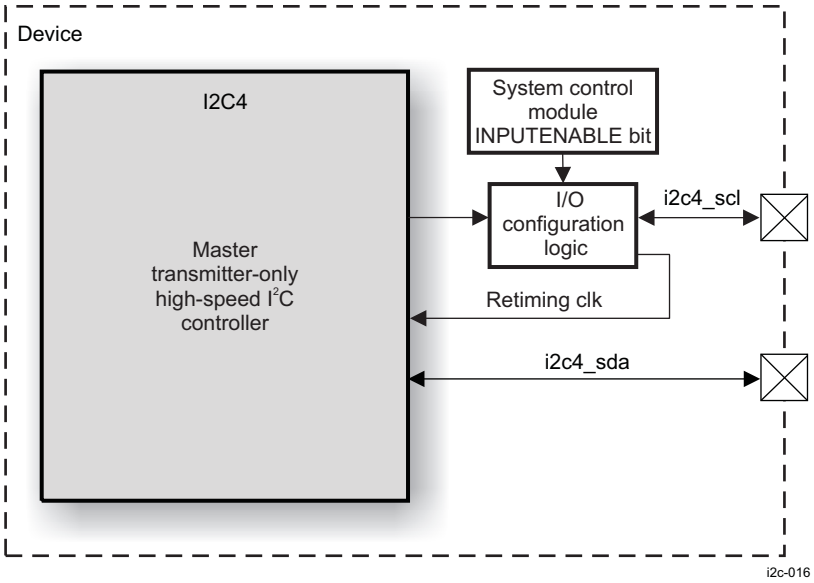
Figure 15-15. Typical Connection Between the HS I²C Controller and Power Chip(s)



15.2.3.1 HS I²C Controller I2C4 Pins for Typical Connections

Figure 15-16 shows the HS I²C controller I2C4 pins used for typical connections with power chips.

Figure 15-16. HS I²C Controller I2C4 Interface Signals



15.2.3.2 HS I²C Controller I2C4 Interface Typical Connections

Table 15-3 lists the pins associated with the I²C interface of the HS I²C controller I2C4 of the device.

Table 15-3. Input/Output Description

Signal	I/O[1]	Description	Reset Value
i2c4_scl	I/O(OD)	I²C serial clock line[2]. Open-drain output buffer. Requires external Rp.	Hi-Z
i2c4_sda	I/O(OD)	I²C serial data line. Open-drain output buffer. Requires external Rp.	Hi-Z

(1) I = Input; O = Output; OD = Open Drain; Hi-Z = High Impedance

(2) This signal is also used as retiming input (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).

15.2.3.3 I²C Typical Connections Protocol and Data Format for I2C4

15.2.3.3.1 Serial Data Format

The serial data format is the same as described in Section 15.2.1.3.1, *Serial Data Format*.

15.2.3.3.2 Data Validity

The data validity is the same as described in Section 15.2.1.3.2, *Data Validity*.

15.2.3.3.3 S and P Conditions

The S and stop P conditions are the same as described in Section 15.2.1.3.3, *Start and Stop Conditions*.

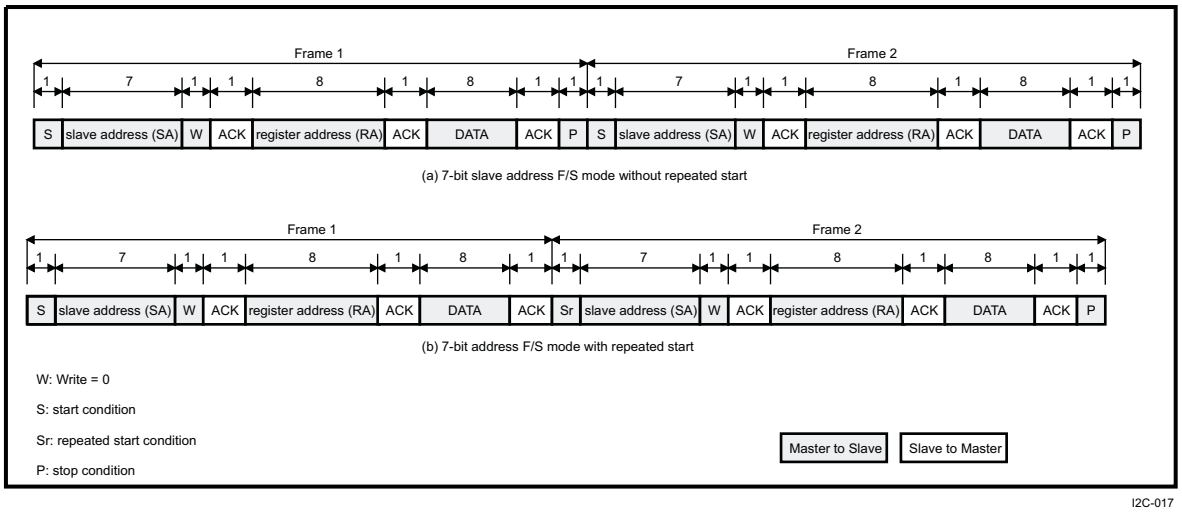
15.2.3.3.4 Addressing

The master transmitter HS I²C controller I2C4 supports only the 7-bit addressing mode. For each frame, the master writes the 8-bit value (DATA) in the register specified by the 8-bit register address (RA) of the slave addressed by the slave address (SA).

15.2.3.3.4.1 Data Transfer Format in F/S Mode

Figure 15-17 shows the I²C data transfer format in F/S mode for the I2C4 module.

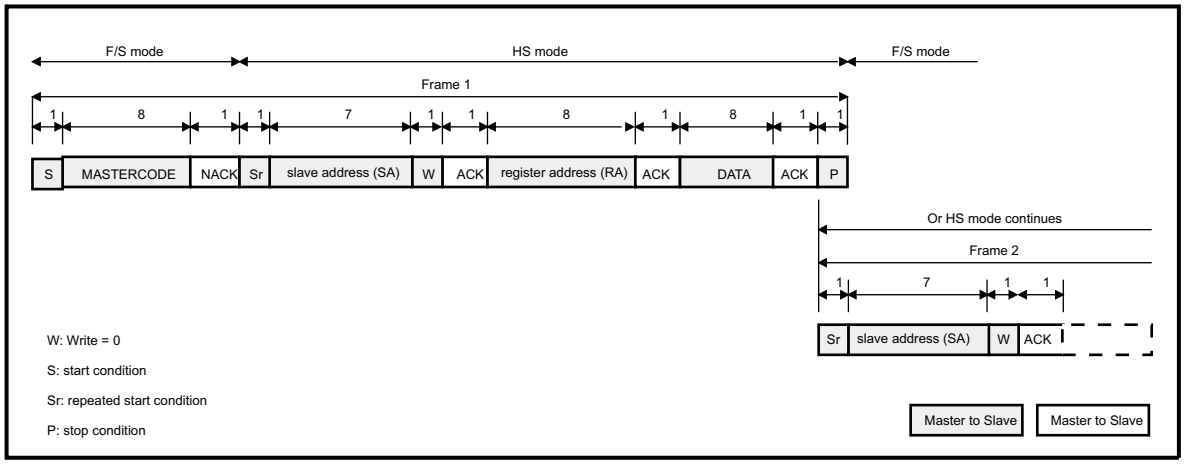
Figure 15-17. I²C Data Transfer Format in F/S Mode for the I2C4 Module



15.2.3.3.4.2 Data Transfer Format in HS Mode

Figure 15-18 shows the I²C data transfer format in HS mode for the I2C4 module.

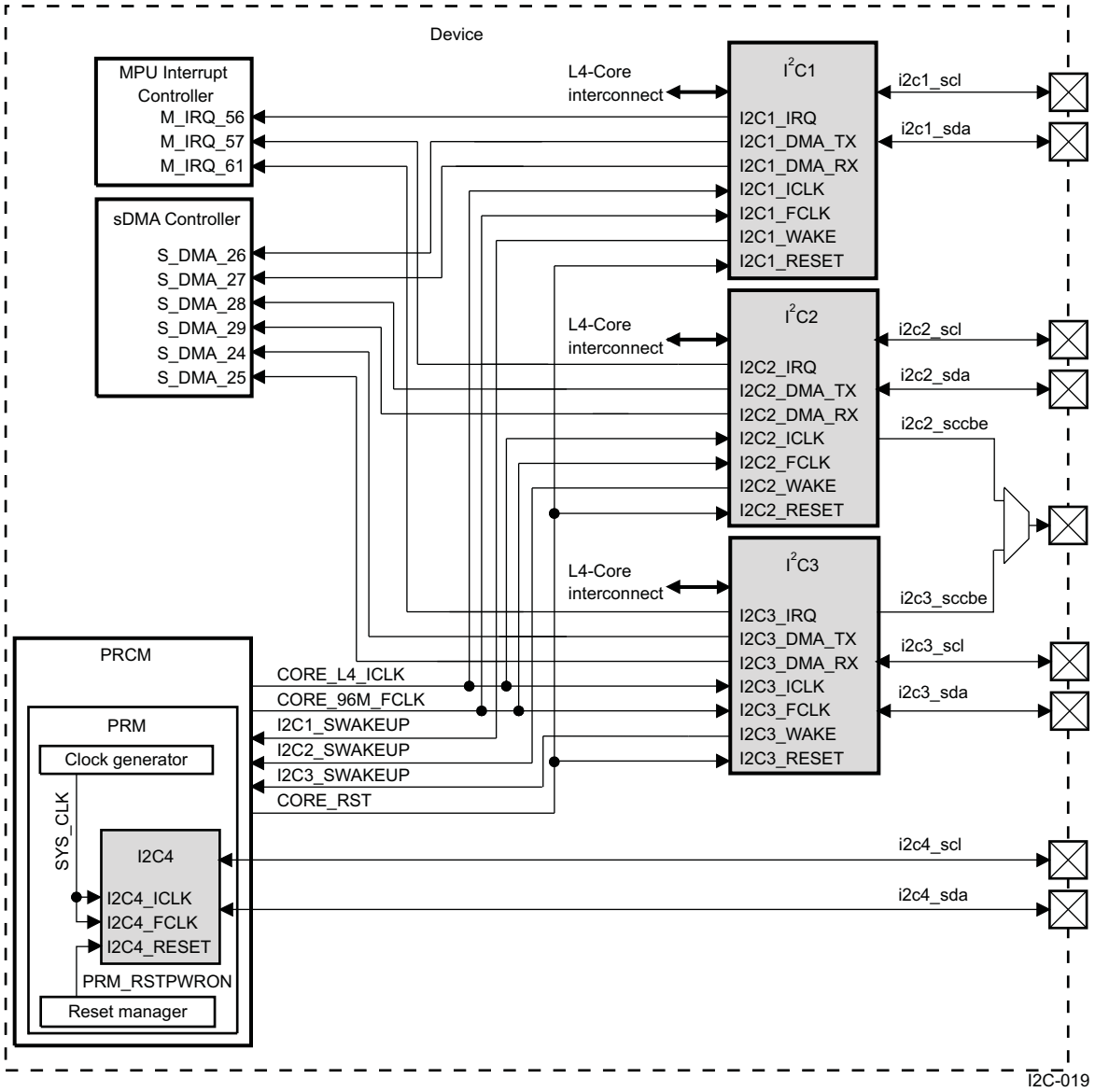
Figure 15-18. I²C Data Transfer Format in HS Mode for the I2C4 Module



15.3 High-Speed I²C Controller Integration

Figure 15-19 shows the integration of the four HS I²C controllers in the device.

Figure 15-19. HS I²C Controller Integration



15.3.1 Clocking, Reset, and Power-Management Scheme

15.3.1.1 Clocks

15.3.1.1.1 Module Clocks

Each multimaster HS I²C controller is clocked with an independent functional clock of 96 MHz (I2Ci_FCLK) and an interface clock (I2Ci_ICLK) for interfacing with the L4-Core interconnect. These clocks are provided by the PRCM module.

The SYS_CLK clock provided by the clock generator of the PRCM module is connected to the functional and interface clocks of the HS I²C controller I2C4. For detailed information about the module clocking, see , *Power, Reset, and Clock Management*.

The interface clock can be enabled or disabled for each multimaster HS I²C controller by setting the following bits in the PRCM module:

- PRCM.CM_ICLKEN1_CORE[15] EN_I2C1 bit for the I2C1 module

- PRCM.CM_ICLKEN1_CORE[16] EN_I2C2 bit for the I2C2 module
- PRCM.CM_ICLKEN1_CORE[17] EN_I2C3 bit for the I2C3 module

The functional clock can be enabled or disabled for each multimaster HS I²C controller by setting the following bits in the PRCM module:

- PRCM.CM_FCLKEN1_CORE[15] EN_I2C1 bit for the I2C1 module
- PRCM.CM_FCLKEN1_CORE[16] EN_I2C2 bit for the I2C2 module
- PRCM.CM_FCLKEN1_CORE[17] EN_I2C3 bit for the I2C3 module

The functional clock is processed by a prescaler block to produce the internal sampling clock. This clock is generated by the I²C prescaler block. The prescaler block consists of the I2Ci.I2C_PSC[7:0] PSC bit field (where i = 1, 2, 3) that is used to divide down the functional clock to obtain an internal sampling clock with a frequency value of I2Ci_FCLK/(I2Ci.I2C_PSC[7:0] PSC bit field value + 1, where i = 1, 2, 3).

NOTE: The I2C4.I2C_PSC[7:0] PSC bit field of the I2C4 module is not accessible by software. For details about the bit rates available for the I2C4 module, see Section 15.4.7.

15.3.1.2 Power Management

15.3.1.2.1 Module Power Saving

This section describes power-saving techniques for the multimaster HS I²C controllers. To conserve power, when no activity is detected on the L4-Core interconnect interface of the module, each of these modules supports an automatic idle mode that is enabled or disabled by setting the I2Ci.I2C_SYSC[0] AUTOIDLE bit.

When this bit is asserted (set to 1), if no activity is detected on the L4-Core interface, the automatic idle mode is enabled and the interface clock I2Ci_ICLK is disabled internally to the module, thus reducing power consumption.

When new activity is detected on the L4-Core interconnect interface of the module, the clock restarts with no latency penalty. After reset, the automatic idle mode is disabled; thus, this mode must be enabled by software for reduced power consumption.

15.3.1.2.2 System Power Management

As part of the system-wide power-management scheme, each multimaster HS I²C controller supports a communication protocol with the PRCM module to request the module to enter a low-power mode. When a module acknowledges a low-power mode request from the PRCM module, the interface and/or the functional clocks are gated off at the PRCM clock generator. Because the clocks are disabled at the source, the low-power mode offers lower power consumption than the internal clock autogating method used by local power management.

The PRCM.CM_ICLKEN1_CORE[15] EN_I2C1, PRCM.CM_ICLKEN1_CORE[16] EN_I2C2, and PRCM.CM_ICLKEN1_CORE[17] EN_I2C3 bits in the PRCM module control the interface clocks of the I2C1, I2C2, and I2C3 modules, respectively.

The PRCM.CM_FCLKEN1_CORE[15] EN_I2C1, PRCM.CM_FCLKEN1_CORE[16] EN_I2C2, and PRCM.CM_FCLKEN1_CORE[17] EN_I2C3 bits in the PRCM module control the functional clocks of the I2C1, I2C2, and I2C3 modules, respectively.

For details about clock enabling/disabling in the PRCM module, see , *Power, Reset, and Clock Management*.

Each multimaster HS I²C controller can be configured through the I2Ci.I2C_SYSC[4:3] IDLEMODE bit field as one of the following acknowledgment modes:

- Force-idle mode (I2Ci.I2C_SYSC[4:3] IDLEMODE bit field = b00): The module immediately enters idle mode on receiving a low-power-mode request from the PRCM module. In this mode, the software must ensure that there are no asserted output interrupts before requesting this mode to go to idle state.
- No-idle mode (I2Ci.I2C_SYSC[4:3] IDLEMODE bit field = b01): The module never enters idle mode.
- Smart-idle mode (I2Ci.I2C_SYSC[4:3] IDLEMODE bit field = b10): After receiving a low-power-mode request from the PRCM module, the module enters idle mode only after all asserted output interrupts are acknowledged and there is no pending internal event.

NOTE: The value I2Ci.I2C_SYSC[4:3] IDLEMODE bit field = b11 must not be used.

Table 15-4 describes the multimaster HS I²C controller power management modes.

Table 15-4. Multimaster HS I²C Controller Power Management Modes

Power management mode requested by the PRCM module	I2Ci.I2C_SYSC[4:3] IDLEMODE bit field value (where i = 1, 2, 3)
Force-idle	b00
No-idle	b01
Smart-idle	b10
Reserved (not used)	b11

The PRCM module gates the interface and/or the functional clocks after receiving the acknowledgment of the I2Ci module (where i = 1, 2, 3). The I2Ci.I2C_SYSC[9:8] CLOCKACTIVITY bit field indicates the state of the interface and functional clocks of the module when in idle mode. Table 15-5 lists the value of the I2Ci.I2C_SYSC[9:8] CLOCKACTIVITY bit field and indicates the state of the interface and functional clocks at the PRCM clock generator output in idle mode.

Table 15-5. State of the Interface and Functional Clocks When the Module is in Idle Mode

I2Ci.I2C_SYSC[9:8] CLOCKACTIVITY bit field value (where i= 1, 2, 3)	Functional Clock	Interface Clock
b00	OFF	OFF
b01	OFF	ON
b10	ON	OFF
b11	ON	ON

NOTE: The PRCM.CM_AUTOIDLE1_CORE[15] AUTO_I2C1, PRCM.CM_AUTOIDLE1_CORE[16] AUTO_I2C2, and PRCM.CM_AUTOIDLE1[17] AUTO_I2C3 bits control, for each multimaster HS I²C controller, whether the L4-Core interconnect interface clock is enabled or disabled in synchronization with the CORE power domain state transition (see , *Power, Reset, and Clock Management*).

NOTE: The voltage controller, in which the HS I²C controller I2C4 is implemented, has no idle request/acknowledge mechanism. The idle modes for the voltage controller are directly managed by the PRM module. For details, see , *Power, Reset, and Clock Management*.

15.3.1.2.3 Wake-Up Capability

Each multimaster HS I²C controller I2Ci can wake up the system by generating a wake-up request through the I2Ci_WAKE signal connected to the PRCM module.

The wake-up request is composed of the merge of all wake-up events. Each wake-up event can be separately enabled or disabled by setting the corresponding bit in the I2Ci.I2C_WE register.

The global wake-up capability of the module can be enabled or disabled by setting the I2Ci.I2C_SYSC[2] ENAWAKEUP bit (1: enabled, 0: disabled).

Figure 15-20 shows the wake-up generation flow.

Figure 15-20. Wake-up Generation Flow

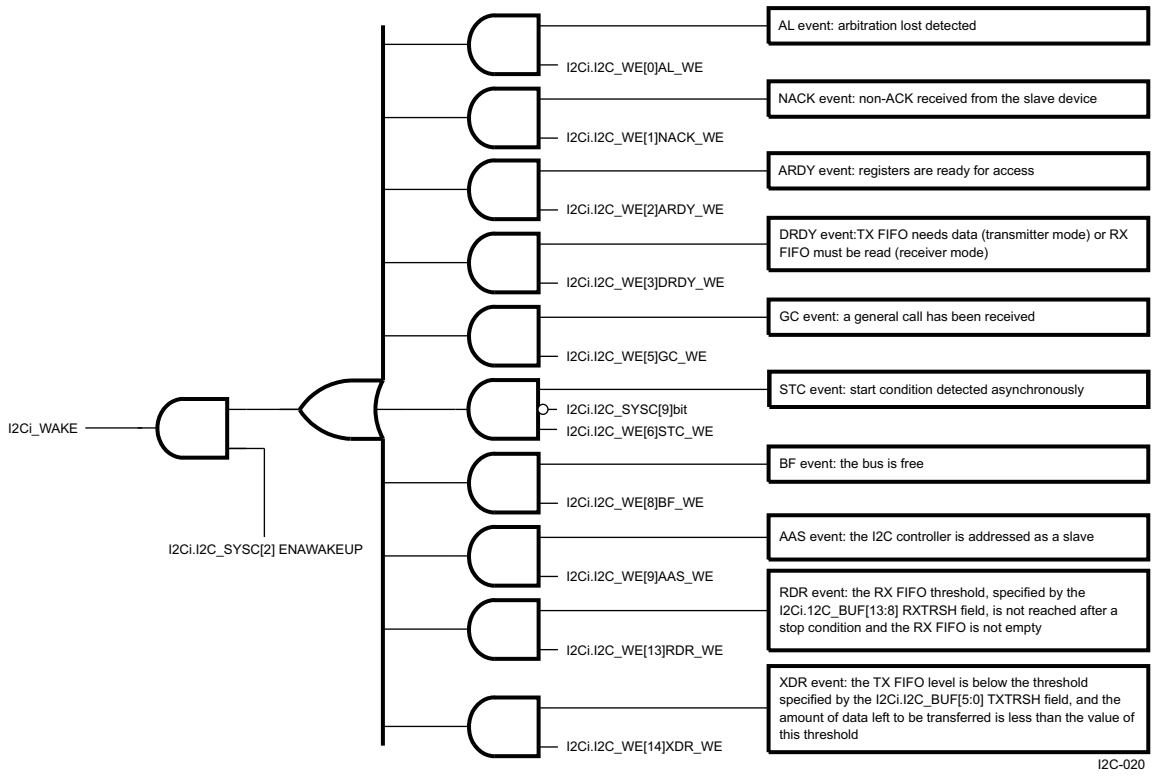


Table 15-6 lists all wake-up events with the corresponding enable/disable bit in the I2Ci.I2C_WE register.

Table 15-6. Wake-up Events

Wake-up Event Name	Supported Configuration Mode	Enable/Disable bit[1]	Event Generated When:
AL event	I²C mode only	I2Ci.I2C_WE[0] AL_WE	The I2Ci module in the device loses the arbitration of the I²C bus in master transmitter mode.
NACK event	I²C mode only	I2Ci.I2C_WE[1] NACK_WE	A nonacknowledgment has been generated on the I²C bus, indicating a transmission error.
ARDY event	I²C receive mode and SCCB read mode	I2Ci.I2C_WE[2] ARDY_WE	The current transaction is finished and the module registers can be accessed.
DRDY event	I²C and SCCB modes	I2Ci.I2C_WE[3] DRDY_WE	The TX FIFO needs some data to be transferred or the RX FIFO must be read.
GC event	I²C mode only	I2Ci.I2C_WE[5] GC_WE	A general call is received on the I²C bus.
STC event [2]	I²C mode only	I2Ci.I2C_WE[6] STC_WE	A start (S) condition is detected on the I²C bus.[3]
BF event	I²C and SCCB modes	I2Ci.I2C_WE[8] BF_WE	The bus is free and the LH can initiate its own transfer.

(1) To enable the corresponding wake-up event generation, set the bit to 1. To disable the corresponding wake-up event generation, set the bit to 0.
(2) Wake-up event asynchronously detected.
(3) This event must not be enabled if the functional clock cannot be disabled.

Table 15-6. Wake-up Events (continued)

Wake-up Event Name	Supported Configuration Mode	Enable/Disable bit[1]	Event Generated When:
AAS event	I²C mode only	I2Ci.I2C_WE[9] AAS_WE	An external master I²C device addresses the module of the device to inform the LH that it can check which of its own addresses was used by the external master I²C device to access the module of the device.
RDR event	I²C receive mode only	I2Ci.I2C_WE[13] RDR_WE	The module, configured as a receiver, has detected a stop (P) condition on the I²C bus and the RX FIFO threshold (I2Ci.I2C_BUF[13:8] RTRSH field value + 1) is not reached and the RX FIFO is not empty. This allows the module to inform the LH that it can check the amount of data to be transferred from the RX FIFO.
XDR event	I²C master transmit mode only	I2Ci.I2C_WE[14] XDR_WE	The TXFIFO level is below the threshold (I2Ci.I2C_BUF[5:0] XTRSH field value + 1) and the amount of data left to be transferred is less than this threshold. This allows the module to inform the LH that it can check the amount of data to be written to the TX FIFO.

NOTE: With the exception of the STC event, the functional clock must be active for wake-up event detection to occur. The HS I²C controller I2C4 has no wake-up capability.

15.3.1.3 Resets

15.3.1.3.1 Hardware Reset

The three multimaster HS I²C controllers receive their reset signal CORE_RST (the reset signal of the CORE power domain) from the PRCM module.

The master transmitter HS I²C controller I2C4 gets its reset signal PRM_RSTPWRON from the reset manager in the PRCM module.

15.3.1.3.2 Software Reset

Each multimaster HS I²C controller supports the software reset by accessing the I2Ci.I2C_SYSC[1] SRST bit (1: reset, 0: normal mode).

The software reset status can be checked by accessing the I2Ci.I2C_SYSS[0] RDONE bit (1: reset is done, 0: reset is ongoing).

To do a software reset, the following steps must be done:

1. Ensure that the module is disabled (clear the I2Ci.I2C_CON[15] I2C_EN bit to 0).
2. Set the I2Ci.I2C_SYSC[1] SRST bit to 1.
3. Enable the module by setting I2Ci.I2C_CON[15] I2C_EN bit to 1.
4. Check the I2Ci.I2C_SYSS[0] RDONE bit until it is set to 1 to indicate the software reset is complete.

NOTE: The I2Ci.I2C_CON[15] I2C_EN bit can hold the functional clock domain of the multimaster HS I²C controller in reset after the device reset has been released. When the system bus reset is removed, this bit remains cleared. The functional part of the I²C controller is held in reset state while this bit is 0, and all configuration registers can be accessed.

The I2Ci.I2C_CON[15] I2C_EN bit must be set to 1 to enable the functional part of the I²C controller.

The I2Ci.I2C_SYSS[0] RDONE bit is asserted only after the module is enabled by setting the I2Ci.I2C_CON[15] I2C_EN bit to 1.

15.3.1.4 Power Domain

The three multimaster HS I²C controllers are connected to the CORE power domain, whereas the HS I²C controller I2C4 belongs to the WKUP power domain.

15.3.2 Hardware Requests

15.3.2.1 DMA Requests

Each multimaster HS I²C controller can generate two DMA requests to the system DMA (sDMA) controller. Table 15-7 lists the DMA requests with mapping on the sDMA controller.

Table 15-7. Multimaster HS I²C Controller DMA Requests

Name	Source	Destination (sDMA controller)	Description
I2C1_DMA_TX	I2C1	S_DMA_26	I2C1 DMA write request to inform the sDMA to write new data in the I2C1.I2C_DATA[7:0] register
I2C1_DMA_RX	I2C1	S_DMA_27	I2C1 DMA read request to inform the sDMA to read the data in the I2C1.I2C_DATA[7:0] register
I2C2_DMA_TX	I2C2	S_DMA_28	I2C2 DMA write request to inform the sDMA to write new data in the I2C2.I2C_DATA[7:0] register
I2C2_DMA_RX	I2C2	S_DMA_29	I2C2 DMA read request to inform the sDMA to read the data in the I2C2.I2C_DATA[7:0] register
I2C3_DMA_TX	I2C3	S_DMA_24	I2C3 DMA write request to inform the sDMA to write new data in the I2C3.I2C_DATA[7:0] register
I2C3_DMA_RX	I2C3	S_DMA_25	I2C3 DMA read request to inform the sDMA to read the data in the I2C3.I2C_DATA[7:0] register

NOTE: The HS I²C controller I2C4 does not generate any DMA request.

15.3.2.2 Interrupt Requests

Each multimaster HS I²C controller can generate an interrupt I2Ci_IRQ to the MPU subsystem. Table 15-8 lists the interrupt requests with the mapping on the MPU interrupt controller.

Table 15-8. Multimaster HS I²C Controller Interrupt Requests

Name	Source	Destination (MPU interrupt controller)
I2C1_IRQ	I2C1	M_IRQ_56
I2C2_IRQ	I2C2	M_IRQ_57
I2C3_IRQ	I2C3	M_IRQ_61

An event can generate an interrupt request when the corresponding mask bit in the I2Ci.I2C_IE register is set to 1. Table 15-9 summarizes the events causing the generation of an interrupt request.

Table 15-9. Multimaster HS I²C Controller Interrupt Events

Event name	Supported configuration mode	Status bit	Mask bit	This event happens when:
AL event	I²C mode only	I2Ci.I2C_STAT [0] AL	I2Ci.I2C_IE[0] AL_IE	Two or more I²C master devices initiate a transfer and the I²C module I2Ci in the device loses arbitration of the I²C bus.
NACK event	I²C mode only	I2Ci.I2C_STAT [1] NACK	I2Ci.I2C_IE[1] NACK_IE	A nonacknowledgment has been received. In master mode, the transfer is automatically ended by generating a stop condition on the bus. The I2Ci.I2C_CON[1] STP, I2Ci.I2C_CON[10] MST and I2Ci.I2C_CON[9] TRX bits are automatically cleared to 0 (slave receiver mode). TX and RX FIFOs must be cleared (I2Ci.I2C_BUF[6] TXFIFO_CLR and I2Ci.I2C_BUF[14] RXFIFO_CLR bits set to 1).

Table 15-9. Multimaster HS I²C Controller Interrupt Events (continued)

Event name	Supported configuration mode	Status bit	Mask bit	This event happens when:
ARDY event	I²C and SCCB modes	I2Ci.I2C_STAT [2] ARDY	I2Ci.I2C_IE[2] ARDY_IE	One of the following cases occurs: • In I²C master receiver mode, the I2Ci.I2C_CON[1] STP bit is set to 1, the I2Ci.I2C_CNT[15:0] DCOUNT field value is 0, and the RX FIFO is empty. • In I²C master transmitter mode, the I2Ci.I2C_CON[1] STP bit is set to 1 and the I2Ci.I2C_CNT[15:0] DCOUNT field value is 0. • In I²C master transmitter mode, the I2Ci.I2C_CON[1] STP bit is cleared to 0 and the I2Ci.I2C_CNT[15:0] DCOUNT field value passed 0. • In I²C master receiver mode, the I2Ci.I2C_CON[1] STP bit is set to 1, the I2Ci.I2C_CNT[15:0] DCOUNT field value passed 0, and the RX FIFO is empty. • In I²C slave transmitter mode, a stop or start (S) condition is received from the external I²C master device. • In I²C slave receiver mode, a stop or start (S) condition is received from the external I²C master device and the RX FIFO is empty. • In SCCB master transmitter or receiver mode, a stop (P) condition is detected.
RRDY event	I²C receive mode and SCCB read mode	I2Ci.I2C_STAT [3] RRDY	I2Ci.I2C_IE[3] RRDY_IE	The RX FIFO level is above the threshold (I2Ci.I2C_BUF[13:8] RTRSH field value + 1).
XRDY event	I²C transmit mode and SCCB write mode	I2Ci.I2C_STAT [4] XRDY	I2Ci.I2C_IE[4] XRDY_IE	The module requires new data to be served. A master transmitter module requests new data when the TX FIFO level is below the threshold (I2Ci.I2C_BUF[5:0] XTRSH field value + 1) and the required amount of data to be transmitted specified by the I2Ci.I2C_BUFSTAT[5:0] TXSTAT field is greater than the threshold. A slave transmitter requests new data when the TX FIFO is below the threshold (if the I2Ci.I2C_BUF[5:0] XTRSH field value is greater than 1), or when there is a read request from the external master device (for each acknowledge received from the master) when the I2Ci.I2C_BUF[5:0] XTRSH field value is 1.
GC event	I²C mode only	I2Ci.I2C_STAT [5] GC	I2Ci.I2C_IE[5] GC_IE	The module detects a general call on the I²C bus (all bits of the address cleared to 0).
STC event	I²C mode only	I2Ci.I2C_STAT [6] STC	I2Ci.I2C_IE[6] STC_IE	The module is in idle mode and a start (S) condition is asynchronously detected on the I²C bus and signaled with a wakeup. When the active mode is restored and the interrupt generated, this bit indicates the reason for the wakeup. See (1).
AERR event	I²C and SCCB modes	I2Ci.I2C_STAT [7] AERR	I2Ci.I2C_IE[7] AERR_IE	A write access to the I2Ci.I2C_DATA register by the LH through the L4-Core interconnect is performed while the TX FIFO is full or a read access to the I2Ci.I2C_DATA register by the LH through the L4-Core interconnect is performed while the RX FIFO is empty. When the RX FIFO is empty, the read of the RX FIFO returns the previous read data value. When the TX FIFO is full, a write in the TX FIFO is ignored.
BF event	I²C and SCCB modes	I2Ci.I2C_STAT [8] BF	I2Ci.I2C_IE[8] BF_IE	The I²C bus becomes free (after a transfer is ended on the bus and a stop (P) condition is detected).
AAS event	I²C mode only	I2Ci.I2C_STAT [9] AAS	I2Ci.I2C_IE[9] AAS_IE	The module has recognized its own slave address or an alternate Own Address, or a general call (all address bits cleared to 0).

(1) The interrupt request generation on an STC event must be enabled only if the module is configured to allow the possibility of switching off the functional clock while in idle state (the I2Ci.I2C_SYSC[9:8] CLOCKACTIVITY field set to b00 or b01). The first transfer (corresponding to the detected start (S) condition) is lost, and is used only to restore the active mode of the module. On the I²C bus, the external master that generated the transfer detects this behavior as a nonacknowledge to the address phase and may possibly restart the transfer.

Table 15-9. Multimaster HS I²C Controller Interrupt Events (continued)

Event name	Supported configuration mode	Status bit	Mask bit	This event happens when:
RDR event	I²C receive mode only	I2Ci.I2C_STAT [13] RDR	I2Ci.I2C_IE[13] RDR_IE	The module is configured as a receiver, a stop (P) condition was received on the I²C bus, and the RX FIFO level is below the threshold (I2Ci.I2C_BUF[13:8] RTRSH field value + 1).
XDR event	I²C master transmit mode only	I2Ci.I2C_STAT [14] XDR	I2Ci.I2C_IE[14] XDR_IE	The module is configured as a master transmitter, the TX FIFO level is below the threshold (I2Ci.I2C_BUF[5:0] XTRSH field value + 1), and the amount of data still to be transferred is less than this threshold.

When an interrupt request is generated, the software must read the I2Ci.I2C_STAT register to check which event caused the interrupt request generation, process accordingly, and acknowledge each processed event by writing 1 to the corresponding bit in the I2Ci.I2C_STAT register.

NOTE: The I2Ci.I2C_STAT[9] AAS status bit, corresponding to the AAS event, can be cleared in two ways:

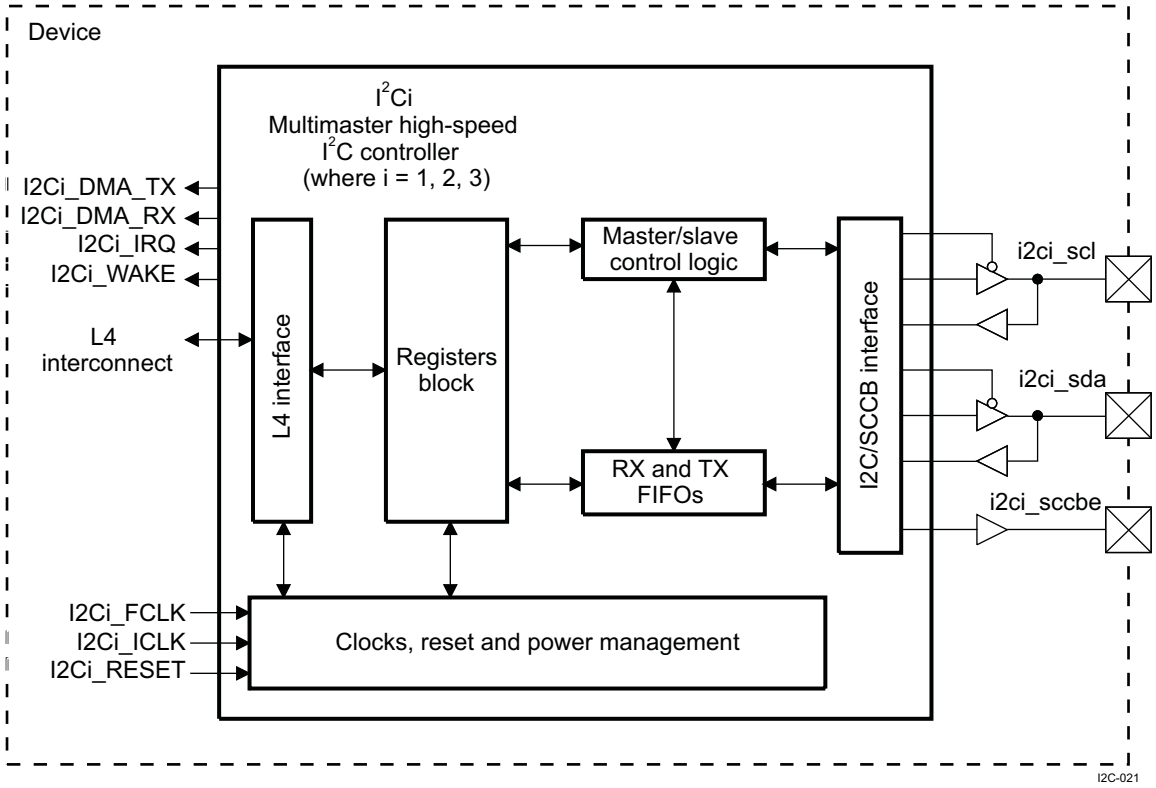
- If the I2Ci.I2C_IE[9] AAS_IE bit is set to 1 (interrupt generation enabled), the status bit is cleared by writing 1 to the I2Ci.I2C_STAT[9] AAS status bit
- If the I2Ci.I2C_IE[9] AAS_IE bit is cleared to 0 (interrupt generation disabled), the status bit is cleared when a new start or stop (P) condition is detected on the I²C bus.

15.4 High-Speed I²C Controller Functional Description

15.4.1 Block Diagram

Figure 15-21 is a functional block diagram of the multimaster I²C HS controller.

Figure 15-21. Multimaster HS I²C Controller Block Diagram



NOTE: The i2c1_sccbe signal is not available.

The three multimaster HS I²C controllers can be configured in F/S I²C mode, in HS I²C mode, or in SCCB mode. The operation mode is selected by configuring the I2Ci.I2C_CON[13:12] OPMODE field. Table 15-10 lists the available operation modes.

Table 15-10. Operation Mode Selection

Operation Mode	I2Ci.I2C_CON[13:12] OPMODE field value
Fast/standard (F/S) I²C	0x0
High-speed (HS) I²C	0x1
SCCB	0x2
Reserved (not used)	0x3

15.4.2 Transmit Mode in I²C Mode

This mode is available for master or slave. The master and slave modes are configurable with the I2Ci.I2C_CON[10] MST bit (0: slave mode, 1: master mode).

In master mode, the transmit mode is configured by setting the I2Ci.I2C_CON[9] TRX bit to 1. The MPU subsystem puts the data to transmit in the TX FIFO by writing to the I2Ci.I2C_DATA[7:0] DATA bit field.

The transmitter can write new data to this register when the I2Ci.I2C_STAT[4] XRDY bit is set to 1, or when the I2Ci.I2C_STAT[14] XDR bit is set to 1 according to the draining mechanism description.

A master transmitter requests new data if the I2Ci.I2C_CNT[15:0] DCOUNT bit field value is not 0. A slave transmitter requests new data if a read is performed by the external master.

NOTE: The HS I2C controller I2C4 is configured in master transmitter mode, and its configuration cannot be changed.

15.4.3 Receive Mode in I²C Mode

This mode is available for master or slave. In master mode, it is configured by clearing the I2Ci.I2C_CON[9] TRX bit to 0.

The MPU subsystem can read new data from the I2Ci.I2C_DATA[15:0] DCOUNT bit field when the I2Ci.I2C_STAT[3] RRDY bit is set to 1, or when the I2Ci.I2C_STAT[13] RDR bit is set according to the draining mechanism description.

Each time the I2Ci.I2C_STAT[3] RRDY bit is set to 1, if the interrupt is enabled (the I2Ci.I2C_IE[3] RRDY_IE bit must be set to 1), the I²C controller generates an interrupt to the MPU subsystem.

NOTE: In interrupt mode, the MPU subsystem must read this bit after each read in the I2Ci.I2C_DATA register to ensure that no other data is waiting for the FIFO to be read. For a new interrupt to be received, the I2Ci.I2C_STAT[3] RRDY bit must be cleared in the interrupt routine.

If the DMA receive mode is enabled (the I2Ci.I2C_BUF[15] RDMA_EN bit is set), this bit is forced to 0 and no interrupt is generated; instead, a DMA RX request to the sDMA controller is generated.

15.4.4 FIFO Management

Each multimaster HS I²C controller implements two internal 8-bit FIFOs, the RX and TX FIFOs.

Table 15-11 lists the depth of these FIFOs, depending on the instance of the I²C controller.

Table 15-11. RX and TX FIFO Depths

I²C Controller Instance	RX and TX FIFO Depth (in bytes)
I2C1	8
I2C2	8
I2C3	64

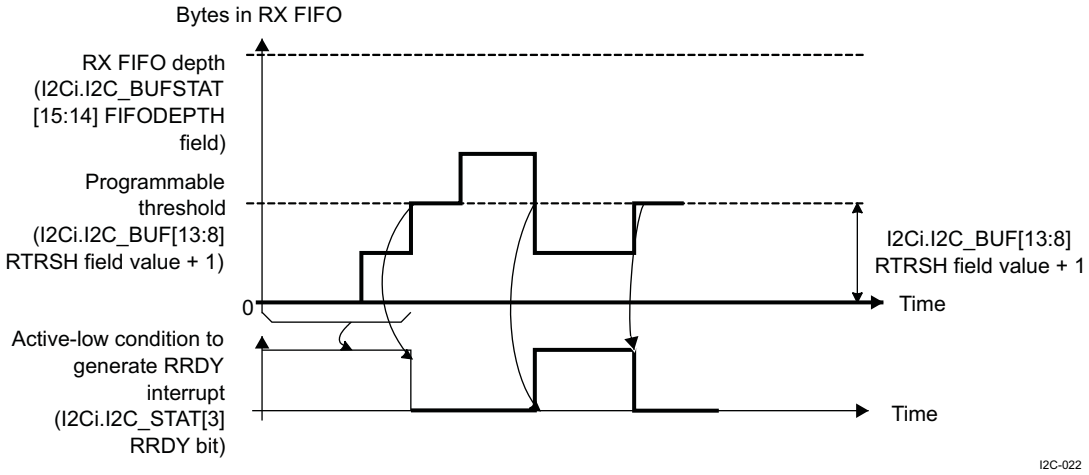
The depth of the RX and TX FIFOs can be checked by reading the I2Ci.I2C_BUFSTAT[15:14] FIFODEPTH bit field (0x0: 8 bytes, 0x1: 16 bytes, 0x2: 32 bytes, and 0x3: 64 bytes).

15.4.4.1 FIFO Interrupt Mode Operation

In FIFO interrupt mode (relevant interrupts enabled by the I2Ci.I2C_IE register), the processor is informed of the receiver and transmitter status by an interrupt signal. These interrupts are raised when the RX/TX FIFO thresholds (defined by the I2Ci.I2C_BUF[13:8] RTRSH field value + 1 for the RX FIFO or the I2Ci.I2C_BUF[5:0] XTRSH field value + 1 for the TX FIFO) are reached; the interrupt signals instruct the LH to transfer data to the destination (from the I²C controller module in receive mode and/or from any source to the I²C controller FIFO in transmit mode).

Figure 15-22 and Figure 15-23 show receive and transmit operations, respectively, from a FIFO management point of view.

Figure 15-22. Receive FIFO Interrupt Request Generation

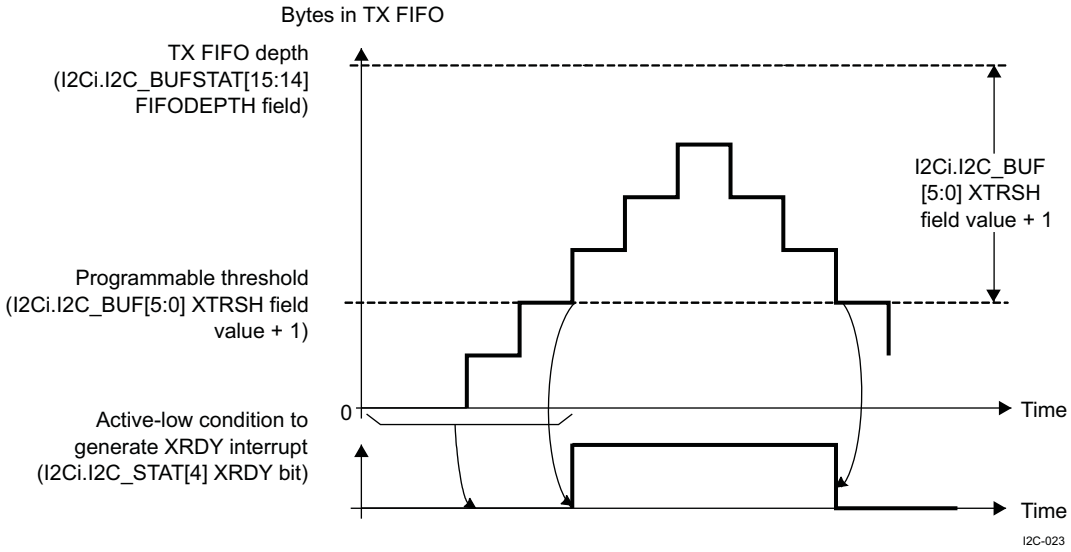


In Figure 15-22, the RRDY interrupt condition shows that the condition for generating an RRDY interrupt is achieved. The interrupt request is generated when this signal is active, and it can be cleared only by the LH by writing 1 in the I2Ci.I2C_STAT[3] RRDY bit. If the condition is still present after clearing the previous interrupt, another interrupt request is generated.

In receive mode, an RRDY interrupt is generated as soon as the FIFO reaches its receive threshold (I2Ci.I2C_BUF[13:8] RTRSH field value + 1). The interrupt can be deasserted only when the LH has handled enough bytes to make the number of bytes in the RX FIFO lower than the programmed threshold. For each interrupt, the LH can be configured to read a number of bytes equal to the value of the RX FIFO threshold.

NOTE: In SCCB mode, the RX and TX threshold values must be set to 1 by setting the I2Ci.I2C_BUF[13:8]RTRSH and I2Ci.I2C_BUF[5:0]XTRSH fields to 0x0.

Figure 15-23. Transmit FIFO Interrupt Request Generation



In Figure 15-23, the XRDY interrupt condition shows that the condition for generating an XRDY interrupt is achieved. The interrupt request is generated when TX FIFO is empty or when the TX FIFO threshold is not reached, and the LH can clear the XRDY status bit by writing 1 in the I2Ci.I2C_STAT[4] XRDY bit after transmitting the configured number of bytes. If the condition is still present after clearing the previous interrupt, another interrupt request is generated.

In interrupt mode, the module offers two options for the LH application to handle the interrupts:

- When detecting an interrupt request (XRDY/RRDY type), the LH can write/read 1 data byte to/from the TX/RX FIFO and then clear the interrupt. The module reasserts the interrupt until the interrupt condition is no longer met.
- When detecting an interrupt request (XRDY/RRDY type), the LH can be programmed to write/read the number of data bytes specified by the corresponding FIFO threshold (I2Ci.I2C_BUF[5:0] XTRSH field value + 1 for the TX threshold or I2Ci.I2C_BUF[13:8] RTRSH field value + 1 for the RX threshold). In this case, the interrupt condition is cleared and the next interrupt is asserted again when the XRDY/RRDY condition is met again.

If the second-interrupt-serving approach is used, an additional mechanism (draining feature) is implemented for cases where the transfer length is not a multiple of the FIFO threshold value (see Section 15.4.4.4).

NOTE: In slave transmit mode (the I2Ci.I2C_CON[10] MST bit is cleared and the I2Ci.I2C_CON[9] TRX bit is set to 1), the draining feature must not be used, because the transfer length is not known at configuration time, and the external master can end the transfer at any point by not acknowledging 1 data byte. If the draining feature is used in slave transmit mode, data can remain in the TX FIFO without being transmitted over the I²C bus. In this case, the TX FIFO must be cleared by setting the I2Ci.I2C_BUF[6] TXFIFO_CLR bit.

15.4.4.2 FIFO Polling Mode Operation

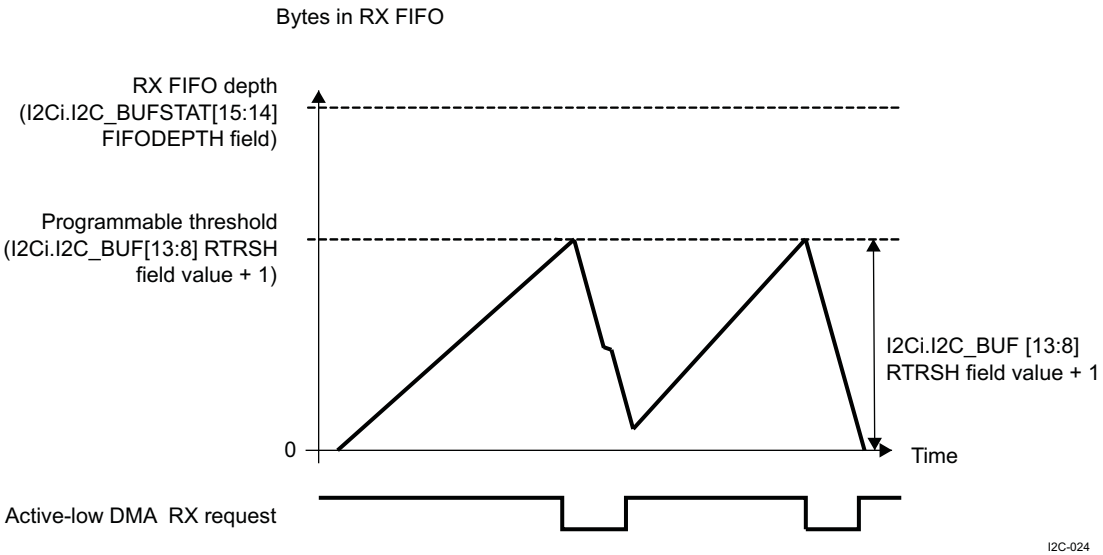
In FIFO polled mode (the I2Ci.I2C_IE[4] XRDY_IE and I2Ci.I2C_IE[3] RRDY_IE bits are disabled, and the I2Ci.I2C_BUF[15] RDMA_EN and I2Ci.I2C_BUF[7] XDMA_EN bits are disabled), the status of the module (receiver or transmitter) can be checked by polling the I2Ci.I2C_STAT[4] XRDY and the I2Ci.I2C_STAT[3] RRDY bits (the I2Ci.I2C_STAT[13] RDR and I2Ci.I2C_STAT[14] XDR bits can also be polled if the draining feature is enabled). The I2Ci.I2C_STAT[4] XRDY and I2Ci.I2C_STAT[3] RRDY bits accurately reflect the interrupt conditions described in the discussion of the FIFO interrupt mode operation.

15.4.4.3 FIFO DMA Mode Operation (I²C Mode Only)

In receive mode, a DMA request is generated by the I2Ci_DMA_RX signal as soon as the RX FIFO exceeds its threshold level (I2Ci.I2C_BUF[13:8] RTRSH field value + 1). This request is deasserted when the number of bytes defined by the threshold level is read by the sDMA controller.

Figure 15-24 shows the DMA request generation in receive mode.

Figure 15-24. Receive FIFO DMA Request Generation



In transmit mode, a DMA request is automatically asserted by the I2Ci_DMA_TX signal when the TX FIFO is empty. This request is deasserted when the number of bytes (I2Ci.I2C_BUF[5:0] XTRSH field value + 1) is written in the FIFO by the sDMA controller. If an insufficient number of bytes is written, the DMA request remains active. Figure 15-25 and Figure 15-26 show the DMA TX transfers with different values for the I2Ci.I2C_BUF[5:0] XTRSH bit field.

Figure 15-25. Transmit FIFO Request Generation (High Threshold)

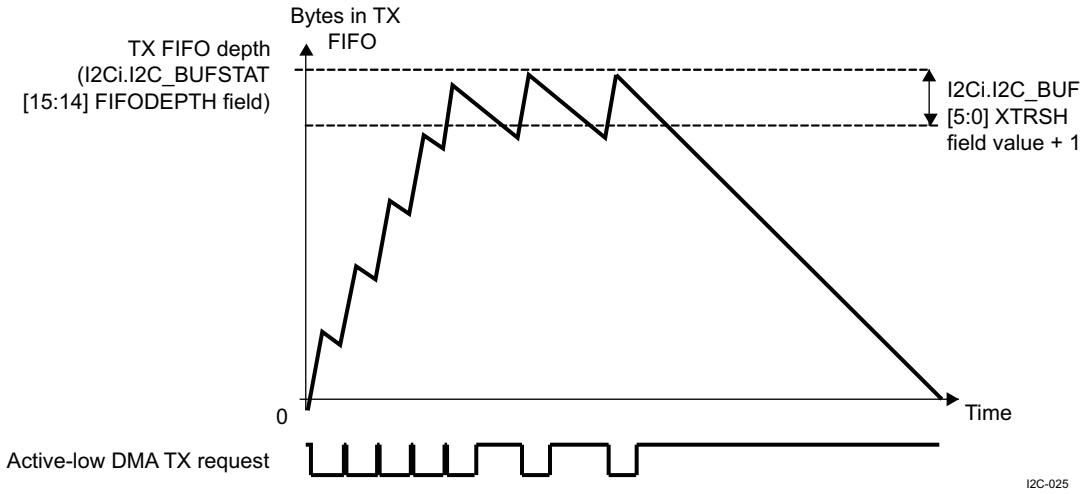
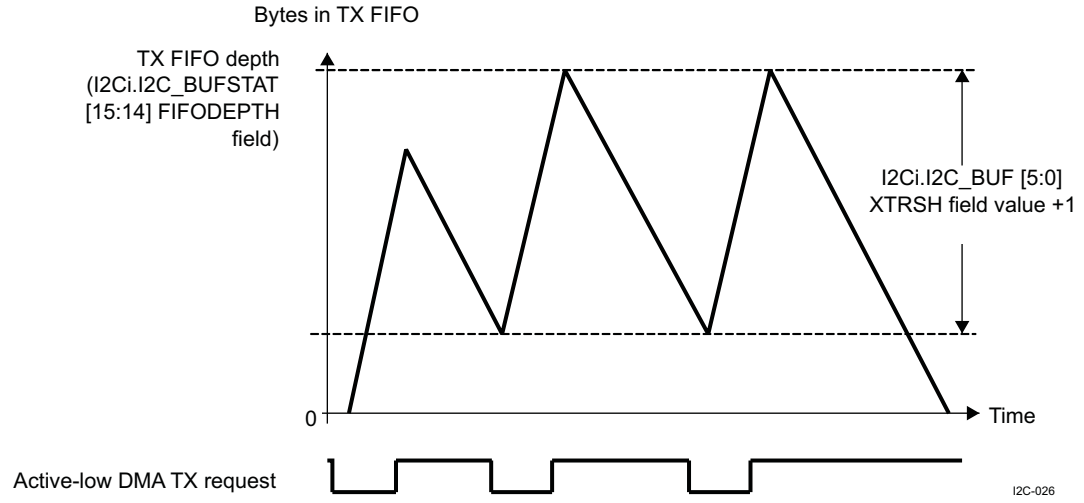


Figure 15-26. Transmit FIFO Request Generation (Low Threshold)



NOTE: In SCCB mode, the RX and TX threshold values must be set to 1 by setting the I2Ci.I2C_BUF[13:8]RTRSH and I2Ci.I2C_BUF[5:0]XTRSH fields to 0x0.

The I²C module allows the user to clear the RX or TX FIFO by setting the I2Ci.I2C_BUF[14] RXFIFO_CLR and I2Ci.I2C_BUF[6] TXFIFO_CLR bits, which act like software reset for the FIFOs. In DMA mode, these bits also reset the DMA state-machines. The FIFO clearing feature can be used when the following conditions are met:

- The module is configured as a transmitter.
- The external receiver responds with a NACK in the middle of the transfer.
- Data in the TX FIFO are still waiting to be transferred.

15.4.4.4 Draining Feature (I²C Mode Only)

The draining feature is implemented to handle the end of a transfer whose length is not a multiple of the FIFO threshold values (I2Ci.I2C_BUF[13:8] RTRSH field value + 1 for the RX threshold and I2Ci.I2C_BUF[5:0] XTRSH field value + 1 for the TX threshold). It also offers the possibility of transferring the remaining number of bytes (because the threshold is not reached).

This feature prevents the LH or the sDMA controller from trying more FIFO accesses than necessary (for example, to generate at the end of a transfer a DMA RX request having fewer bytes in the FIFO than the configured DMA transfer length). Otherwise, an AERR interrupt is generated by the I2Ci.I2C_STAT[7] AERR status bit.

The draining mechanism generates an interrupt using the I2Ci.I2C_STAT[13] RDR or the I2Ci.I2C_STAT[14] XDR status bit at the end of the transfer, informing the LH that it must check the amount of data left to be transferred (the I2Ci.I2C_BUFSTAT[13:8] RXSTAT or I2Ci.I2C_BUFSTAT[5:0] TXSTAT fields) and enable the draining feature of the DMA controller by reconfiguring the DMA transfer length according to this value (when the DMA mode is enabled) or perform only the required number of data accesses (when the DMA mode is disabled).

In receive mode (master or slave), if the RX FIFO threshold (I2Ci.I2C_BUF[13:8] RTRSH field value + 1) is not reached, but the transfer ends on the I²C bus and there is still data in the RX FIFO (less than the threshold), the receive draining interrupt (the I2Ci.I2C_STAT[13] RDR status bit) is asserted to inform the LH that it can read the amount of data in the FIFO (the I2Ci.I2C_BUFSTAT[13:8] RXSTAT field). The LH performs a number of data read accesses equal to the I2Ci.I2C_BUFSTAT[13:8] RXSTAT field value (interrupt or polling mode), or reconfigures the sDMA controller with the required value to drain the FIFO.

In master transmit mode, if the TX FIFO threshold (I2Ci.I2C_BUF[5:0] XTRSH field value + 1) is not reached, but the amount of data remaining to be written in the TX FIFO is less than the threshold, the transmit draining interrupt (the I2Ci.I2C_STAT[14] XDR status bit) is asserted to inform the LH that it can read the amount of data remaining to be written in the TX FIFO (the I2Ci.I2C_BUFSTAT[5:0] TXSTAT field). The LH must write the required number of data bytes specified by the I2Ci.I2C_BUFSTAT[5:0] TXSTAT field value or reconfigure the sDMA controller with the value required to transfer the last bytes to the FIFO.

In master mode, the LH can alternately skip the checking of the I2Ci.I2C_BUFSTAT[5:0] TXSTAT and I2Ci.I2C_BUFSTAT[13:8] RXSTAT field values, because it can obtain this information internally (by computing the the I2Ci.I2C_CNT[15:0] DATACOUNT field value modulo I2Ci.I2C_BUF[13:8] RTRSH or I2Ci.I2C_BUF[5:0] XTRSH).

By default, the draining feature is disabled; it can be enabled using the I2Ci.I2C_IE[14] XDR_IE or I2Ci.I2C_IE[13] RDR_IE bits (default disabled) only for transfers with lengths not equal to the threshold values (I2Ci.I2C_BUF[5:0] XTRSH field value + 1 for the TX threshold or I2Ci.I2C_BUF[13:8] RTRSH field value + 1 for the RX threshold).

15.4.5 Programmable Multislave Channel Feature (I²C Mode Only)

This feature allows each multimaster HS I²C controller to be addressed using four separate Own Addresses configured in the I2Ci.I2C_OAx registers (with x = 0, 1, 2, 3). An additional register (I2Ci.I2C_ACTOA) is used to indicate to the LH which address is used by the external master to communicate with the I²C controller.

Each Own Address can be independently configured in 7-bit or 10-bit mode by setting the corresponding bit (I2Ci.I2C_CON[7] XOA0, I2Ci.I2C_CON[6] XOA1, I2Ci.I2C_CON[5] XOA2, or I2Ci.I2C_CON[4] XOA3).

15.4.6 Automatic Blocking of the I²C Clock Feature (I²C Mode Only)

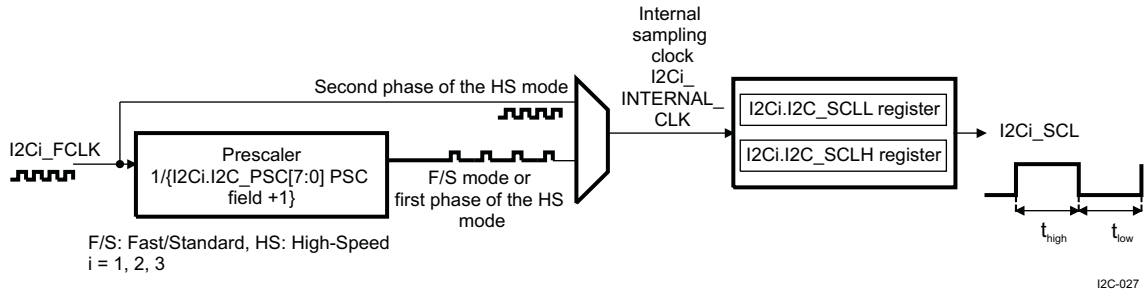
This feature offers the possibility for the LH to command the blocking of the I²C clock after the slave addressing phase, when the I²C controller is addressed by an external master device using a certain Own Address.

The release of the I²C clock (i2ci_scl signal, with i = 1, 2, 3) can be performed independently for each Own Address (I2Ci.I2C_OAi registers, with i = 0, 1, 2, 3) by deasserting the corresponding bit in the I2Ci.I2C_SBLOCK register.

15.4.7 Clocking

Figure 15-27 shows the I²C clock generation of the HS I²C controllers.

Figure 15-27. I²C Clock Generation



Each multimaster HS I²C controller uses the I2Ci_FCLK functional clock in the PRCM module. The internal sampling clock I2Ci_INTERNAL_CLK is generated by dividing the functional clock by (I2Ci.I2C_PSC[7:0] PSC bit field value + 1) in F/S mode, in SCCB mode, or in the first phase of HS mode; or by directly using the functional clock in the second phase of HS mode (prescaler is bypassed).

The low time of the I2Ci_SCL signal is determined by the I2Ci.I2C_SCLL[7:0] SCLL field in F/S mode, in SCCB mode, or in the first phase of HS mode; or by the I2Ci.I2C_SCLL[15:8] HSSCLL bit field in the second phase of HS mode.

The high time of the I2Ci_SCL signal is determined by the I2Ci.I2C_SCLH[7:0] SCLH bit field in F/S mode, in SCCB mode, or in the first phase of HS mode; or by the I2Ci.I2C_SCLH[15:8] HSSCLH bit field in the second phase of HS mode.

Table 15-12 lists the t_{LOW} and t_{HIGH} values in master mode only (in slave mode, the I²C controller does not generate the I²C clock).

Table 15-12. HS I²C t_{LOW} and t_{HIGH} Values of the I²C Clock

Mode	I2Ci_INTERNAL_CLK	t_{LOW}	t_{HIGH}
F/S, SCCB, or HS first phase	I2Ci_INTERNAL_CLK = I2Ci_FCLK / (I2Ci.I2C_PSC[7:0] PSC field + 1)	(I2Ci.I2C_SCLL[7:0] SCLL bit field value + 7) x I2Ci_INTERNAL_CLK period	(I2Ci.I2C_SCLH[7:0] SCLH bit field value + 5) x I2Ci_INTERNAL_CLK period
HS second phase	I2Ci_FCLK	(I2Ci.I2C_SCLL[15:8] HSSCLL field value + 7) x I2Ci_INTERNAL_CLK period	(I2Ci.I2C_SCLH[15:8] HSSCLH bit field value + 5) x I2Ci_INTERNAL_CLK period

NOTE: The equations in Table 15-12 give the SCL timing values for SCLL/SCLH/HSSCLL/HSSCLH at HS I²C controller outputs. Actual tlow and thigh periods may vary, depending on the board (the load capacitance on the SCL signal). If necessary, any adjustments to the SCLL/SCLH/HSSCLL/HSSCLH values must be determined by measurements of actual SCL signal on the board

NOTE: For HS mode, both the I2Ci.I2C_SCLL[15:8] HSSCLL and I2Ci.I2C_SCLL[7:0] SCLL fields must be programmed (the first phase of an HS transaction is performed at F/S speed).

For HS mode, both the I2Ci.I2C_SCLH[15:8] HSSCLH and I2Ci.I2C_SCLH[7:0] SCLH fields must be programmed (the first phase of an HS transaction is performed at F/S speed).

CAUTION

During active mode (I2Ci.I2C_CON[15] I2C_EN bit is set to 1), make no changes to the I2Ci.I2C_SCLL and I2Ci.I2C_SCLH registers. Changes may result in unpredictable behavior.

NOTE: Each multimaster HS I²C controller can be used with an internal secondary pullup. This pullup is mandatory when the I²C controller is configured in HS mode for a bit rate of 3.4M bps, and the bus line capacitance exceeds 45 pF. Pullups can be programmed through the CONTROL.CONTROL_DEVCONF1[12] I2C1HSMASTER bit for I2C1, the CONTROL.CONTROL_DEVCONF1[13] I2C2HSMASTER bit for I2C2, and the CONTROL.CONTROL_DEVCONF1[14] I2C3HSMASTER bit for I2C3.

The maximum bit rate specified by the SCCB specifications is 100K b/s.

15.4.8 Noise Filter

The noise filter is used to suppress any noise that is 50 ns or less in case of F/S and SCCB operation modes, and any noise that is 10 ns or less in case of HS mode operation. The noise filter is always one period of the I2Ci_INTERNAL_CLK clock. This way, for HS mode operation (prescaler bypassed), the filter suppresses spikes of less than 10.4 ns.

For SCCB modes (for example, I2Ci.I2C_PSC[7:0] PSC bit field = 4), the maximum width of suppressed spikes is 52 ns.

To ensure correct filtering, the prescaler must be programmed accordingly by the 2Ci.I2C_PSC[7:0] PSC bit field.

15.4.9 System Test Mode

A system test mode is available for multimaster HS I²C controller module testing. This mode is enabled by setting the I2Ci.I2C_SYSTEST[15] ST_EN to 1. When this bit is cleared to 0, the I²C controller is configured in normal operation mode.

In system test mode, the I2Ci_SYSTEST[13:12] TMODE bit field selects the type of test. Table 15-13 lists the tests available for the multimaster HS I²C controllers.

Table 15-13. List of tests for the Multimaster HS I²C Controllers

I2Ci.I2C_SYSTEST[13:12] TMODE field value	Test	Description
b00	Functional mode	Normal operation mode
b01	Reserved (not used)	
b10	Test of i2ci_scl serial clock line	The i2ci_scl line is driven with a permanent clock as if mastered with the parameters set in the I2Ci.I2C_PSC, I2Ci.I2C_SCLL, and I2Ci.I2C_SCLH registers.
b11	Loop-back mode + i2ci_scl/ i2ci_sda/ i2ci_sccbe input/output	In the master transmit mode only, data transmitted out of the I2Ci.I2C_DATA register (write action) is received in the same I2Ci.I2C_DATA register through an internal path through the FIFO buffers. The DMA and interrupt requests are normally generated if they are enabled. Moreover, the i2ci_scl, i2ci_sda, and i2ci_sccbe lines are controlled with the I2Ci.I2C_SYSTEST[4:0] bits.

NOTE: When the I2Ci.I2C_SYSTEST[13:12] TMODE field = b11, the I²C controller must be configured in I²C F/S (I2Ci.I2C_CON[13:12] OPMODE = b00) or I²C HS mode (I2Ci.I2C_CON[13:12] OPMODE = b01). The loop-back mode is not available in SCCB mode (I2Ci.I2C_CON[13:12] OPMODE = b10).

NOTE: In normal operation mode (I2Ci.I2C_SYSTEST[15] ST_EN clear to 0), the I2Ci.I2C_SYSTEST[4:0] bits that control the i2ci_scl, i2ci_sda, and i2ci_sccbe lines in system test mode are read-only bits.

In system test mode (I2Ci.I2C_SYSTEST[15] ST_EN set to 1), the I2Ci.I2C_STAT[5:0] status bits can be set to 1 when the I2Ci.I2C_SYSTEST[11] SSB bit is set to 1. Clearing the I2Ci.I2C_SYSTEST[11] SSB bit to 0 does not clear the I2Ci.I2C_STAT[5:0] status bits to 0. The I2Ci.I2C_STAT[5:0] status bits can be cleared to 0 only by writing 1 in the corresponding bits.

15.4.10 Write and Read Operations in SCCB Mode

In SCCB mode, the multimaster HS I²C controller can write or read a single byte to or from the external SCCB device.

To write a single byte to the external SCCB device, the multimaster HS I²C controller must be configured in multimaster transmitter mode by setting the I2Ci.I2C_CON[10] MST and I2Ci.I2C_CON[9] TRX bits to 1. The external device slave address (7-bit address of the ID value) is set in the I2Ci.I2C_SA register; the register address (8-bit subaddress in the external SCCB device) is set in the I2Ci.I2C_OA0 register. The 8-bit data to be transmitted is written by the LH in the I2Ci.I2C_DATA register.

To read a single byte from the external SCCB device, the multimaster HS I²C controller must be configured in multimaster receiver mode by setting the I2Ci.I2C_CON[10] MST to 1 and by clearing the I2Ci.I2C_CON[9] TRX bit to 0. The external device slave address (7-bit address of the ID value) is set in the I2Ci.I2C_SA register; the register address (8-bit subaddress in the external SCCB device) is set in the I2Ci.I2C_OA0 register. The 8-bit data received from the external SCCB device is read by the LH from the I2Ci.I2C_DATA register.

NOTE: In SCCB mode, the RX and TX thresholds must be set to 1 by configuring the I2Ci.I2C_BUF[13:8] RTRSH and I2Ci.I2C_BUF[5:0]XTRSH fields to 0x0.

15.4.11 Power Chip Communication Operations

The master transmitter HS I²C controller I2C4 inside the voltage controller of the PRM module is used to send configuration commands from the LH to external power chip(s).

For voltage control operations, the LH must set the slave address of the first power chip in the PRCM.PRM_VC_SMPS_SA[6:0] SA0 field, and the slave address of the second power chip (if necessary) in the PRCM.PRM_VC_SMPS_SA[22:16] SA1 field.

The LH must also configure the specific registers in the voltage controller of the PRCM module, for the voltage control and power-sequencing functions.

A high-priority bypass mode is available to allow the LH to configure the external power chip(s) through the I²C bus. To write an 8-bit data to the configuration registers of an external power chip, the LH must set the 7-bit external power chip slave address in the PRCM.PRM_VC_BYPASS_VAL[6:0] SLAVEADDR field, the configuration register address in the PRCM.PRM_VC_BYPASS_VAL[15:8] REGADDR field, and the 8-bit data in the PRCM.PRM_BYPASS_VAL[23:16] register.

For details about voltage control, see , *Power, Reset, and Clock Management*.

15.5 High-Speed I²C Controller Basic Programming Model

15.5.1 Multimaster HS I²C Controller Basic Programming Model in I²C Mode

This section describes the programming model of the multimaster HS I²C controllers configured in I²C mode.

15.5.1.1 Main Program

15.5.1.1.1 Configure the Module Before Enabling the I²C Controller

Before enabling the I²C controller, perform the following steps:

1. Enable the functional and interface clocks (see Section 15.3.1.1.1).
2. Program the prescaler to obtain an approximately 12-MHz internal sampling clock (I2Ci_INTERNAL_CLK) by programming the corresponding value in the I2Ci.I2C_PSC[3:0] PSC field. This value depends on the frequency of the functional clock (I2Ci_FCLK). Because this frequency is 96MHz, the I2Ci.I2C_PSC[7:0] PSC field value is 0x7.
3. Program the I2Ci.I2C_SCLL[7:0] SCLL and I2Ci.I2C_SCLH[7:0] SCLH fields to obtain a bit rate of 100K bps or 400K bps. These values depend on the internal sampling clock frequency (see Table 15-12).
4. (Optional) Program the I2Ci.I2C_SCLL[15:8] HSSCLL and I2Ci.I2C_SCLH[15:8] HSSCLH fields to obtain a bit rate of 400K bps or 3.4M bps (for the second phase of HS mode). These values depend on the internal sampling clock frequency (see Table 15-12).
5. (Optional) If a bit rate of 3.4M bps is used and the bus line capacitance exceeds 45 pF, program the CONTROL.CONTROL_DEVCONF1[12] I2C1HSMASTER bit for I2C1, the CONTROL.CONTROL_DEVCONF1[13] I2C2HSMASTER bit for I2C2, or the CONTROL.CONTROL_DEVCONF1[14] I2C3HSMASTER bit for I2C3.
6. Configure the Own Address of the I²C controller by storing it in the I2Ci.I2C_OA0 register. Up to four Own Addresses can be programmed in the I2Ci.I2C_OAi registers (with i = 0, 1, 2, 3) for each I²C controller.

NOTE: For a 10-bit address, set the corresponding expand Own Address bit in the I2Ci.I2C_CON register.

7. Set the TX threshold (in transmitter mode) and the RX threshold (in receiver mode) by setting the I2Ci.I2C_BUF[5:0]XTRSH field to (TX threshold - 1) and the I2Ci.I2C_BUF[13:8]RTRSH field to (RX threshold - 1), where the TX and RX thresholds are greater than or equal to 1.
8. Take the I²C controller out of reset by setting the I2Ci.I2C_CON[15] I2C_EN bit to 1.

15.5.1.1.2 Initialize the I²C Controller

To initialize the I²C controller, perform the following steps:

1. Configure the I2Ci.I2C_CON register:
   - For master or slave mode, set the I2Ci.I2C_CON[10] MST bit (0: slave, 1: master).
   - For transmitter or receiver mode, set the I2Ci.I2C_CON[9] TRX bit (0: receiver, 1: transmitter).
2. For a module that should work in master mode, set the CONTROL.CONTROL_PADCONF_x.INPUTENABLE bit for the corresponding I²C module to achieve the synchronous clock on the clock output.
3. If using an interrupt to transmit/receive data, set to 1 the corresponding bit in the I2Ci.I2C_IE register (the I2Ci.I2C_IE[4] XRDY_IE bit for the transmit interrupt, the I2Ci.I2C_IE[3] RRDY bit for the receive interrupt).
4. If using DMA to receive/transmit data, set to 1 the corresponding bit in the I2Ci.I2C_BUF register (the I2Ci.I2C_BUF[15] RDMA_EN bit for the receive DMA channel, the I2Ci.I2C_BUF[7] XDMA_EN bit for the transmit DMA channel).

15.5.1.1.3 Configure Slave Address and the Data Control Register

In master mode, configure the slave address register by programming the I2Ci.I2C_SA[9:0] SA field and the number of data bytes (I²C data payload) associated with the transfer by programming the I2Ci.I2C_CNT[15:0] DCOUNT field.

NOTE: For a 10-bit address, set the I2Ci.I2C_CON[8] XSA bit to 1.

15.5.1.1.4 Initiate a Transfer

Poll the I2Ci.I2C_STAT[12] BB bit. If it is cleared to 0 (bus not busy), configure the I2Ci.I2C_CON[0] STT and I2Ci.I2C_CON[1] STP bits. To initiate a transfer, the I2Ci.I2C_CON[0] STT bit must be set to 1, and it is not mandatory to set the I2Ci.I2C_CON[1] STP bit to 1.

15.5.1.1.5 Receive Data

Poll the I2Ci.I2C_STAT[3] RRDY bit, or use the RRDY interrupt (the I2Ci.I2C_IE[3] RRDY_IE bit must be set to 1) or the DMA RX channel (the I2Ci.I2C_BUF[15] RDMA_EN bit must be set to 1) to read the receive data in the I2Ci.I2C_DATA register.

If the transfer length does not equal the RX FIFO threshold (I2Ci.I2C_BUF[13:8]RTRSH field + 1), use the draining feature (enable the RDR interrupt by setting the I2Ci.I2C_IE[13] RDR_IE bit to 1).

NOTE: In receive mode only, the I2Ci.I2C_STAT[11] ROVR (receive overrun) bit indicates whether the receiver has experienced overrun. An overrun condition occurs when the shift register and the RX FIFO are full. An overrun condition does not result in data loss; the I²C controller simply holds the serial clock line i2ci_scl to low to prevent other bytes from being received.

The I2Ci.I2C_STAT[7] AERR bit is set to 1 when a read access is performed in the I2Ci.I2C_DATA register while the RX FIFO is empty. The corresponding interrupt can be enabled by setting the I2Ci.I2C_IE[7] AERR_IE bit to 1.

15.5.1.1.6 Transmit Data

Poll the I2Ci.I2C_STAT[4] XRDY bit, or use the XRDY interrupt (the I2Ci.I2C_IE[4] XRDY_IE bit must be set to 1) or the DMA TX channel (the I2Ci.I2C_BUF[7] XDMA_EN bit must be set to 1) to write data to the I2Ci.I2C_DATA register.

If the transfer length does not equal the TX FIFO threshold (I2Ci.I2C_BUF[5:0] XTRSH field + 1), use the draining feature (enable the XDR interrupt by setting the I2Ci.I2C_IE[14] XDR_IE bit to 1).

NOTE: In transmit mode only, the I2Ci.I2C_STAT[10] XUDF bit indicates whether the transmitter has experienced underflow.

In master transmit mode, underflow occurs when the shift register and the TX FIFO are empty and there are still some bytes to transmit (the I2Ci.I2C_CNT[15:0] DCOUNT field value is not 0).

In slave transmit mode, underflow occurs when the shift register and the TX FIFO are empty and the external I²C master device still requests data bytes to be read.

The I2Ci.I2C_STAT[7] AERR bit is set to 1 when a write access is performed in the I2Ci.I2C_DATA register while the TX FIFO is full. The corresponding interrupt can be enabled by setting the I2Ci.I2C_IE[7] AERR_IE bit to 1.

15.5.1.2 Interrupt Subroutine Sequence

1. Test for arbitration lost (I2Ci.I2C_STAT[0] AL status bit) and resolve accordingly.
2. Test for no acknowledgment (I2Ci.I2C_STAT[1] NACK status bit) and resolve accordingly.
3. Test for register access ready (I2Ci.I2C_STAT[2] ARDY status bit) and resolve accordingly.

4. Test for receive data ready (I2Ci.I2C_STAT[3] RRDY status bit) and resolve accordingly.
5. Test for transmit data ready (I2Ci.I2C_STAT[4] XRDY status bit) and resolve accordingly.
6. Test for general call (I2Ci.I2C_STAT[5] GC status bit) and resolve accordingly.
7. Test for start (S) condition (I2Ci.I2C_STAT[6] STC status bit) and resolve accordingly. For this test, the functional clock must be inactive.
8. Test for access error (I2Ci.I2C_STAT[7] AERR status bit) and resolve accordingly.
9. Test for bus free (I2Ci.I2C_STAT[8] BF status bit) and resolve accordingly.

15.5.1.3 Programming Flow Diagrams

Figure 15-28 through Figure 15-36 are procedure flowcharts for programming the F/S and HS I²C modes.

Figure 15-28. I²C Setup Procedure

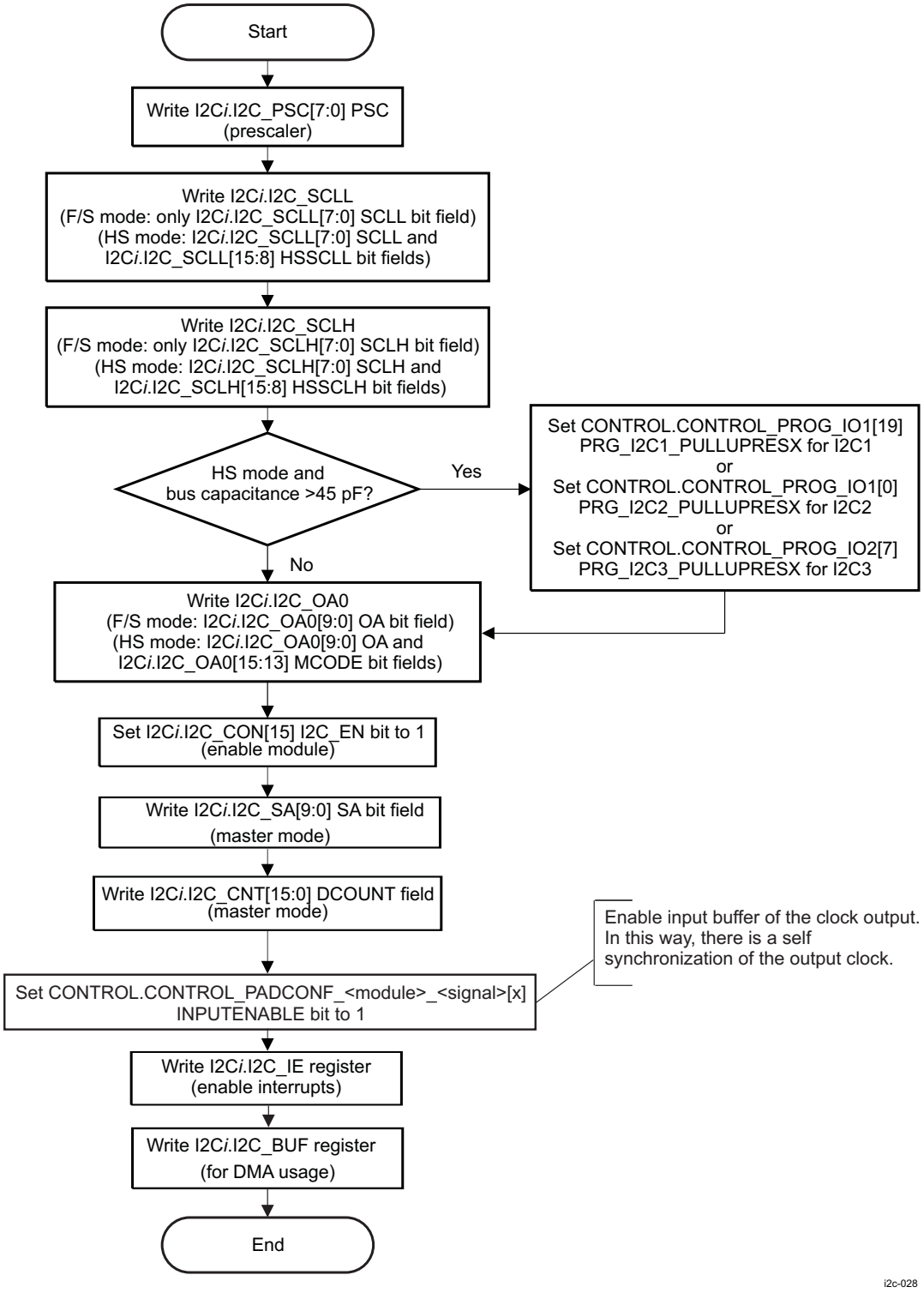
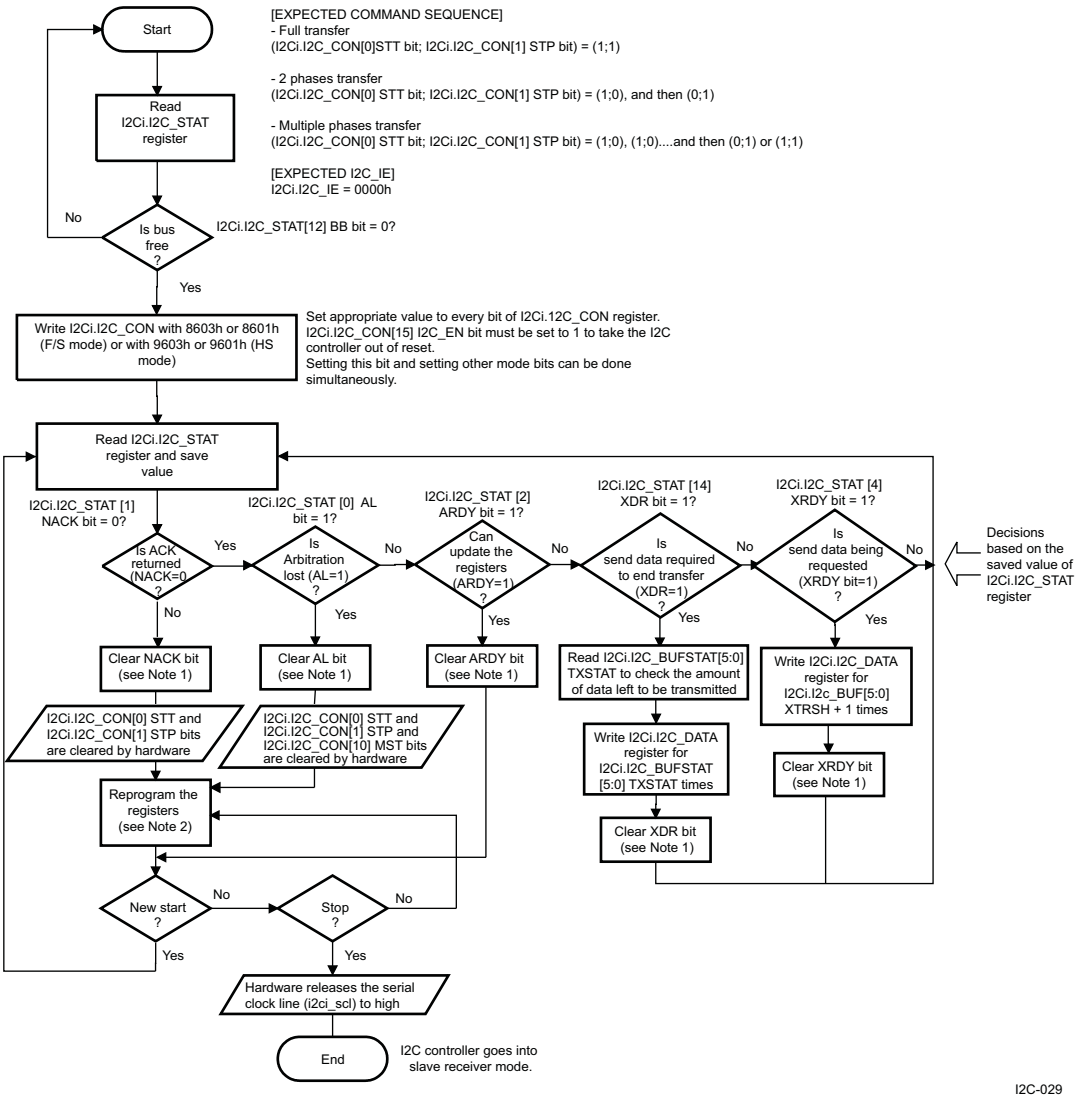


Figure 15-29. I²C Master Transmitter Mode, Polling Method, in F/S and HS Modes

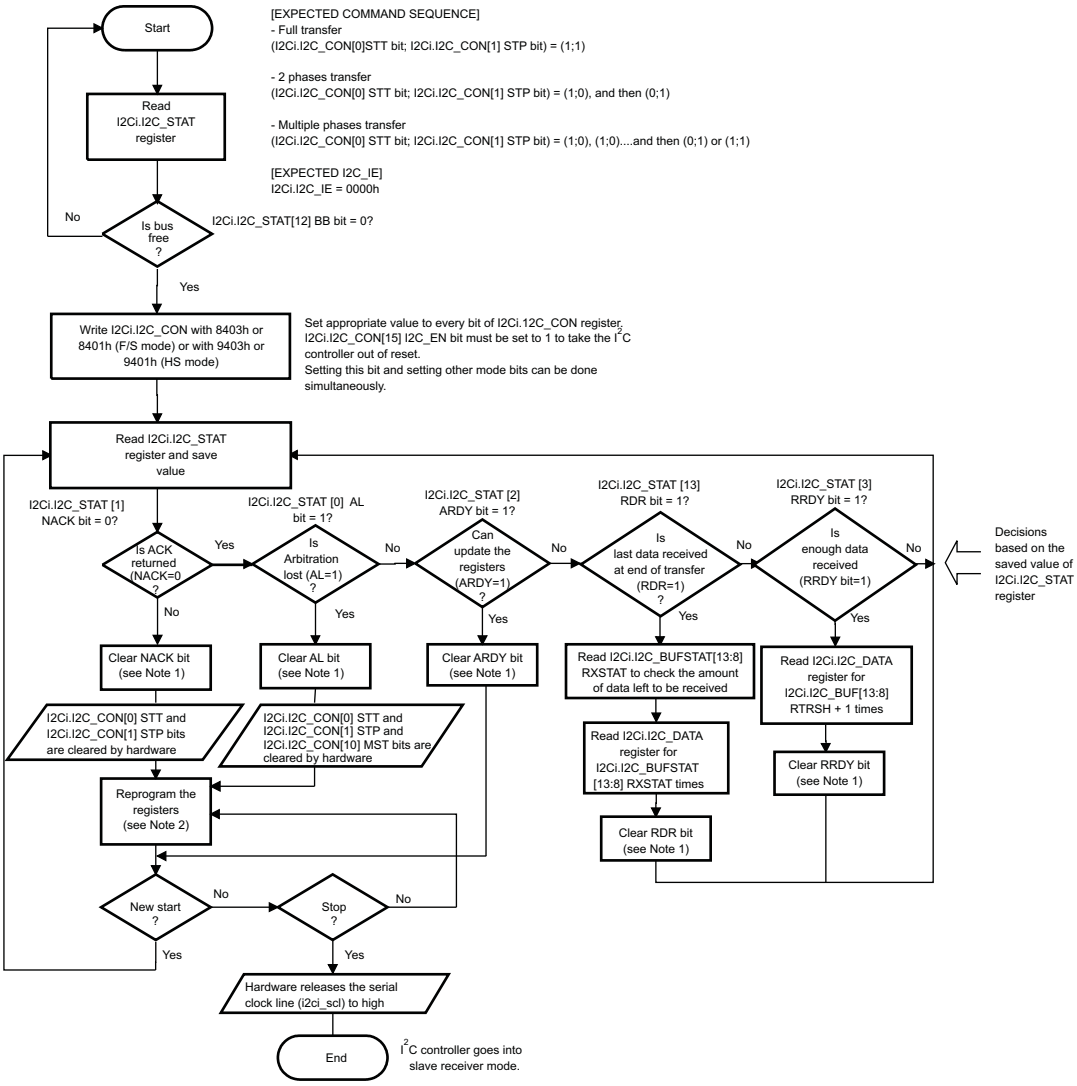


(1) The NAK, AL, ARDY, XDR, and XRDY bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) Reprogram the registers means: I2Ci.I2C_CON[11] STB and/or I2Ci.I2C_CON[10] MST bit and/or I2Ci.I2C_SA[9:0] SA register and/or I2Ci.I2C_CNT[15:0] DCOUNT register and/or I2Ci.I2C_CON[0] STT bit and/or I2Ci.I2C_CON[1] STP bit.

NOTE: In HS mode, the repeated start (Sr) condition and the clock frequency switching are automatically generated by the multimaster HS I²C controller.

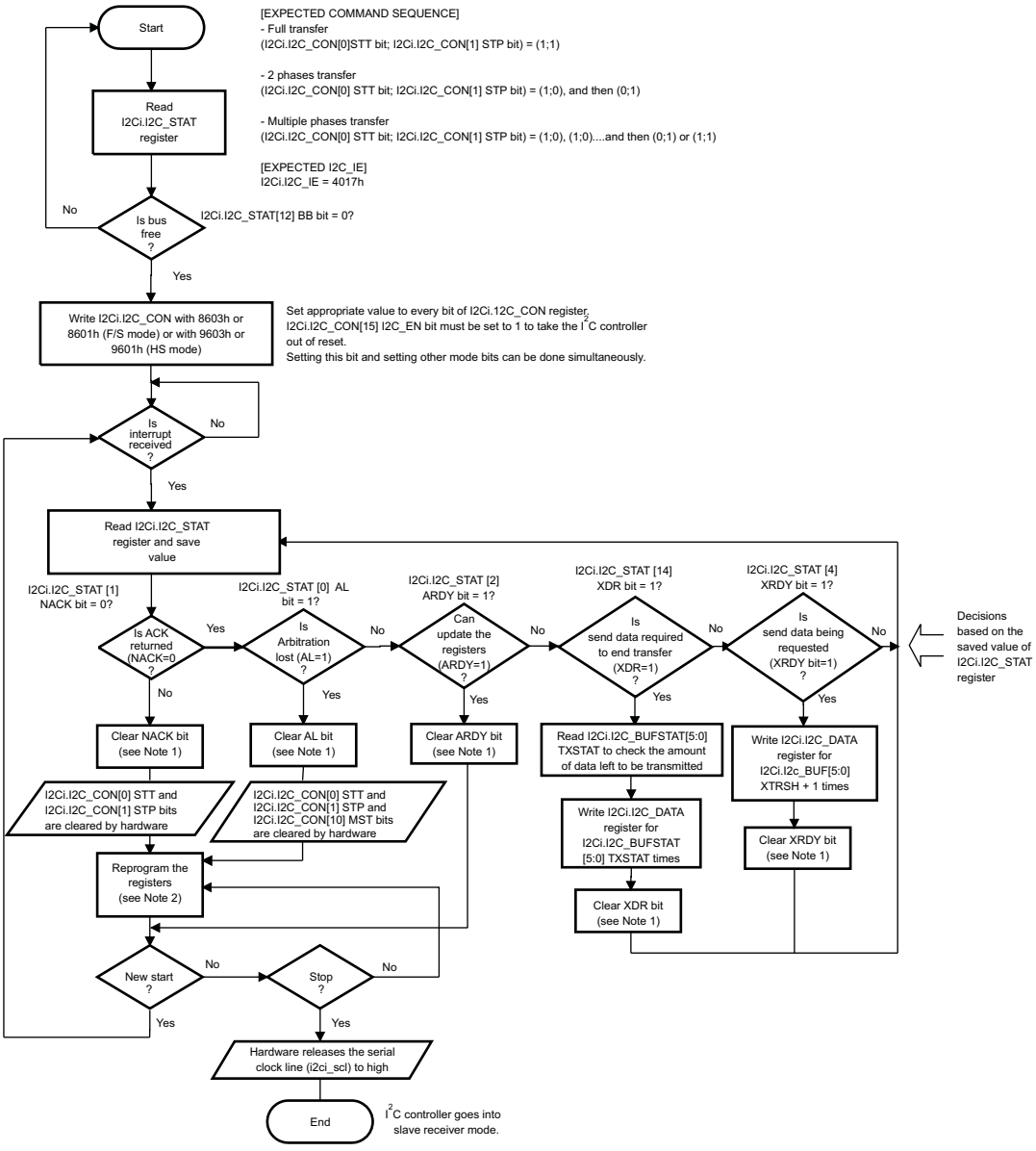
Figure 15-30. I²C Master Receiver Mode, Polling Method, in F/S and HS Modes



(1) The NAK, AL, ARDY, RDR, and RRDY bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) Reprogram registers means: I2Ci.I2C_CON[11] STB and/or I2Ci.I2C_CON[10] MST bit and/or I2Ci.I2C_SA[9:0] SA register and/or I2Ci.I2C_CNT[15:0] DCOUNT register and/or I2Ci.I2C_CON[0] STT bit and/or I2Ci.I2C_CON[1] STP bit.

Figure 15-31. I²C Master Transmitter Mode, Interrupt Method, in F/S and HS Modes

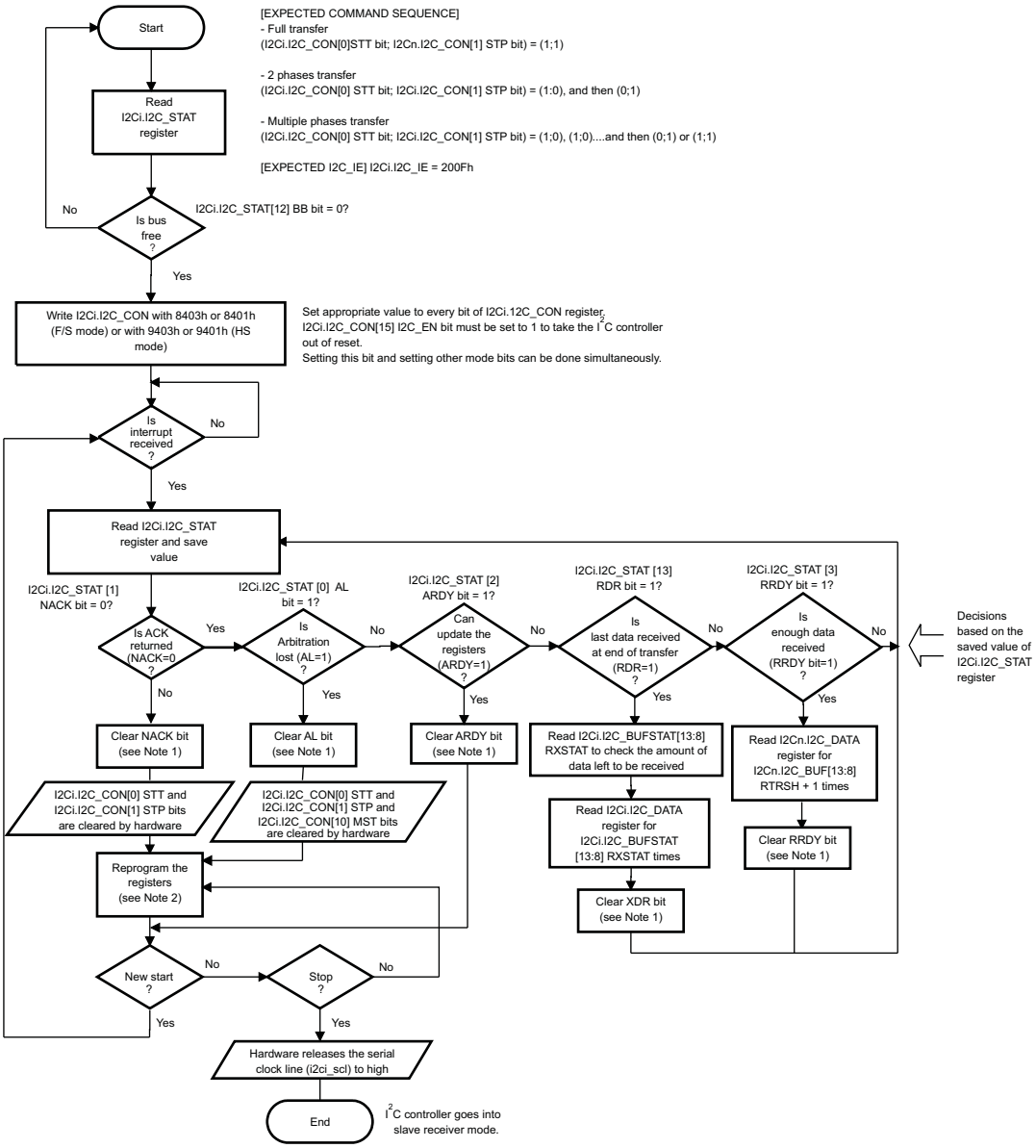


(1) The NAK, AL, ARDY, XDR, and XRDY bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) Reprogram registers means: I2Ci.I2C_CON[11] STB and/or I2Ci.I2C_CON[10] MST bit and/or I2Ci.I2C_SA[9:0] SA register and/or I2Ci.I2C_CNT[15:0] DCOUNT register and/or I2Ci.I2C_CON[0] STT bit and/or I2Ci.I2C_CON[1] STP bit.

NOTE: In HS mode, the repeated start (Sr) condition and the clock frequency switching are automatically generated by the multimaster HS I²C controller.

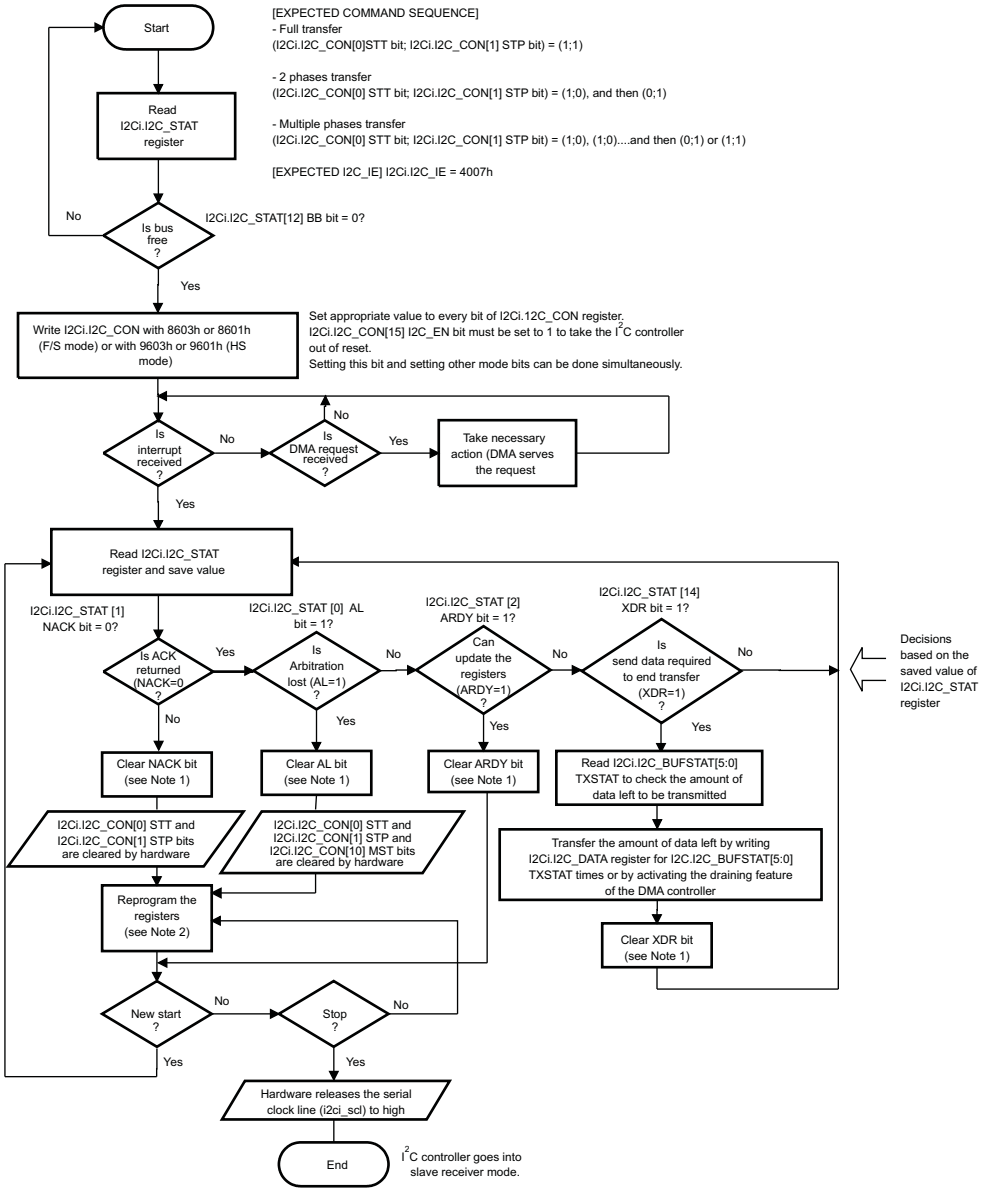
Figure 15-32. I²C Master Receiver Mode, Interrupt Method, in F/S and HS Modes



(1) The NAK, AL, ARDY, RDR, and RRDY bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) Reprogram registers means: I2Ci.I2C_CON[11] STB and/or I2Ci.I2C_CON[10] MST bit and/or I2Ci.I2C_SA[9:0] SA register and/or I2Ci.I2C_CNT[15:0] DCOUNT register and/or I2Ci.I2C_CON[0] STT bit and/or I2Ci.I2C_CON[1] STP bit.

Figure 15-33. I²C Master Transmitter Mode, DMA Method in F/S and HS Modes

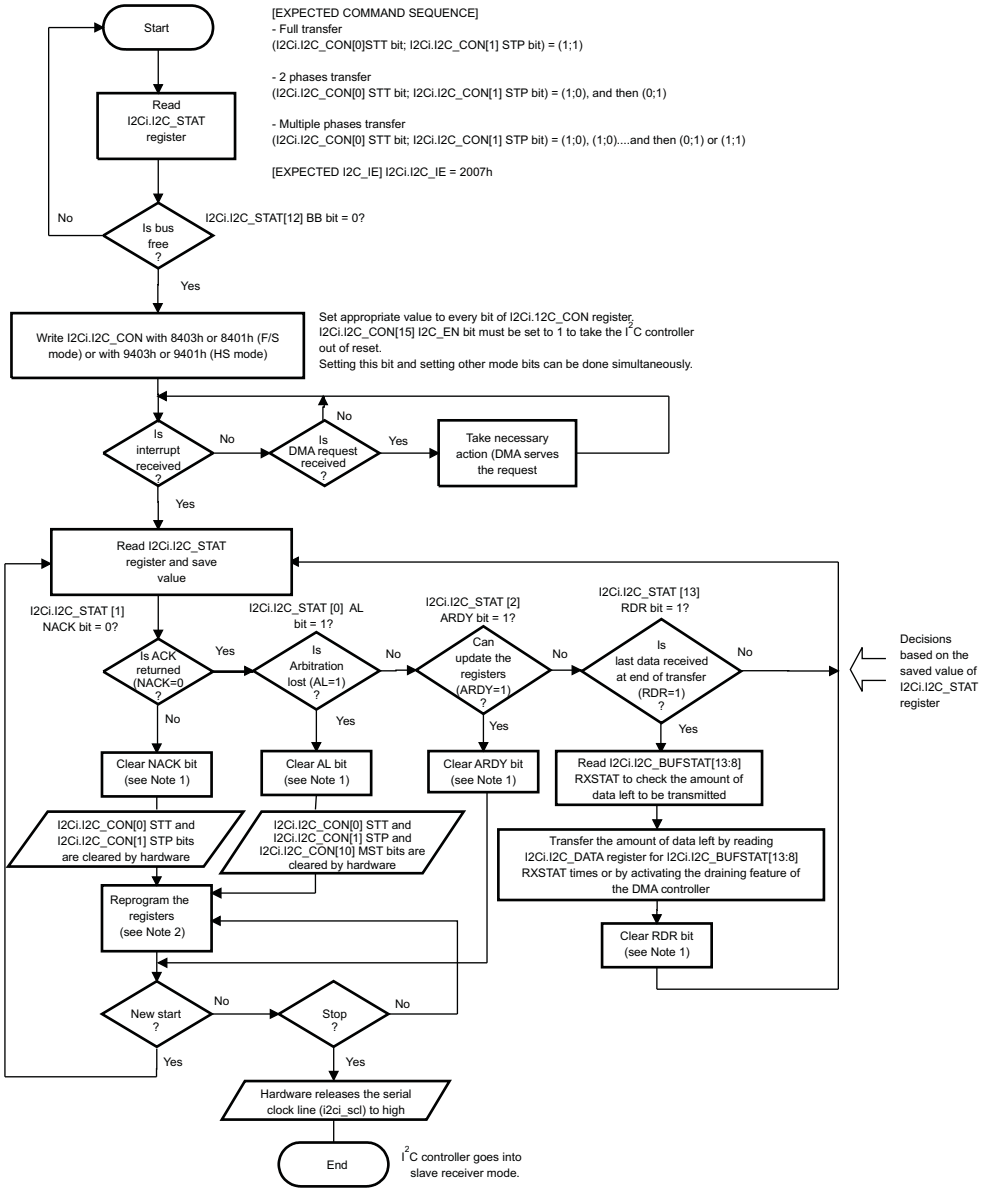


(1) The NAK, AL, ARDY, and XDR bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) Reprogram registers means: I2Ci.I2C_CON[11] STB and/or I2Ci.I2C_CON[10] MST bit and/or I2Ci.I2C_SA[9:0] SA register and/or I2Ci.I2C_CNT[15:0] DCOUNT register and/or I2Ci.I2C_CON[0] STT bit and/or I2Ci.I2C_CON[1] STP bit.

NOTE: In HS mode, the repeated start (Sr) condition and the clock frequency switching are automatically generated by the multimaster HS I²C controller.

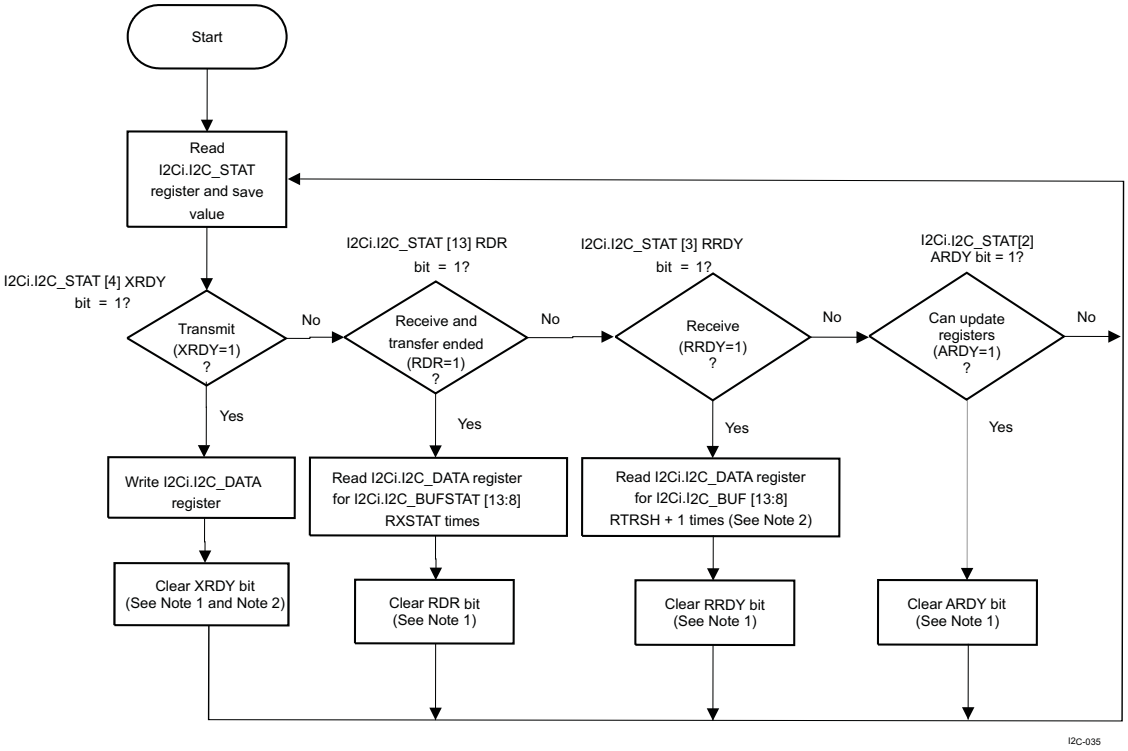
Figure 15-34. I²C Master Receiver Mode, DMA Method in F/S and HS Modes



(1) The NAK, AL, ARDY and RDR bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) Reprogram registers means: I2Ci.I2C_CON[11] STB and/or I2Ci.I2C_CON[10] MST bit and/or I2Ci.I2C_SA[9:0] SA register and/or I2Ci.I2C_CNT[15:0] DCOUNT register and/or I2Ci.I2C_CON[0] STT bit and/or I2Ci.I2C_CON[1] STP bit.

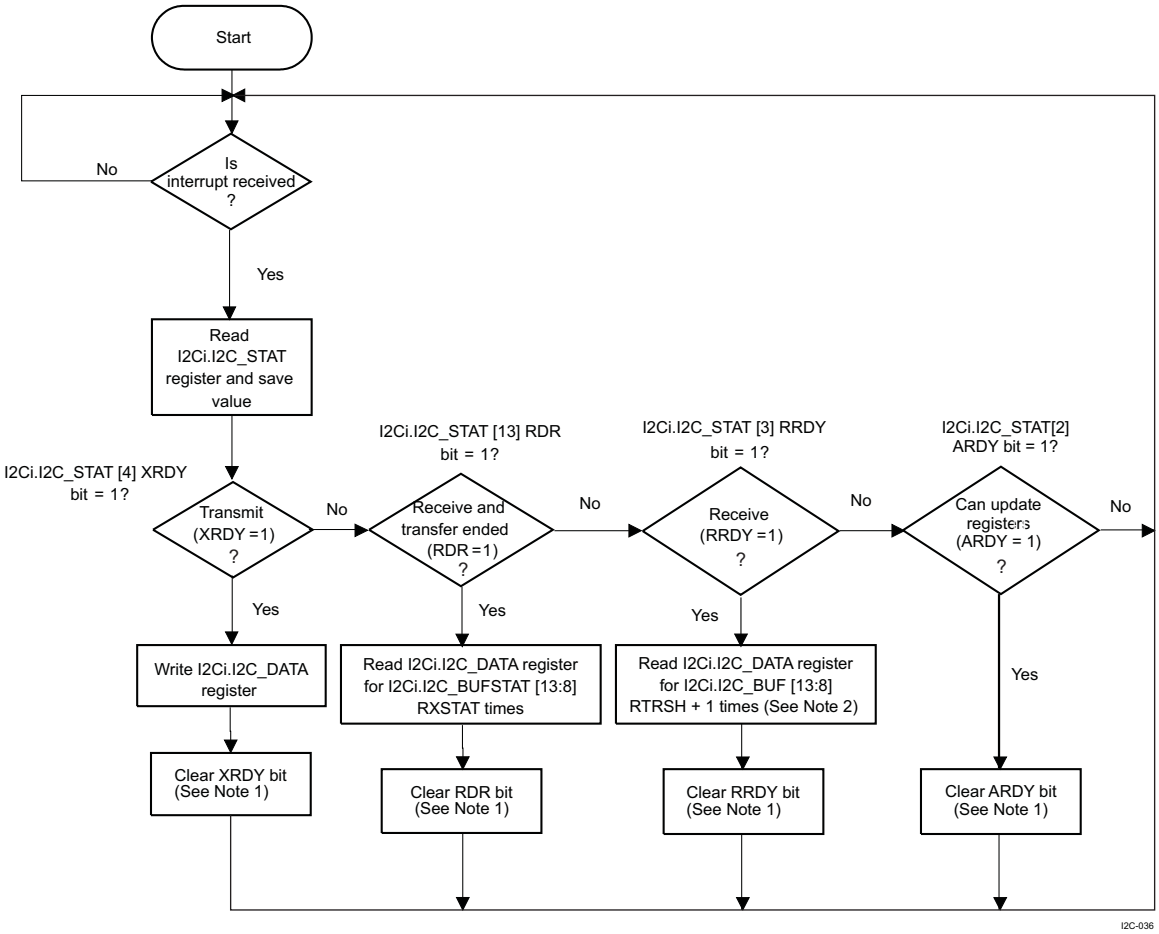
Figure 15-35. I²C Slave Transmitter/Receiver Mode, Polling



(1) The XRDY, RDR, RRDY and ARDY bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) In slave transmitter mode, the amount of data requested by the external master I²C device is unknown; thus, the I2Ci.I2C_BUF[5:0] XTRSH field must be configured to 0x0 (TX threshold = 1).

Figure 15-36. I²C Slave Transmitter/Receiver Mode, Interrupt



(1) The XRDY, RDR, RRDY and ARDY bits are cleared by writing 1 to each corresponding bit in the I2Ci.I2C_STAT register.

(2) In slave transmitter mode, the amount of data requested by the external master I²C device is unknown; thus, the I2Ci.I2C_BUF[5:0] XTRSH field must be configured to 0x0 (TX threshold = 1).

15.5.2 High-Speed I²C Controller Basic Programming Model in SCCB Mode

This section describes the programming model of the multimaster HS I²C controllers configured in SCCB mode.

15.5.2.1 Main Program

15.5.2.1.1 Configure the Module Before Enabling the I²C Controller

Before enabling the I²C controller, perform the following steps:

1. Enable the functional and interface clocks (see Section 15.3.1.1.1).
2. Program the prescaler to obtain an approximately 12-MHz internal sampling clock (I2Ci_INTERNAL_CLK) by programming the corresponding value in the I2Ci.I2C_PSC[7:0] PSC field. This value depends on the frequency of the functional clock (I2Ci_FCLK). Because this frequency is 96MHz, the I2Ci.I2C_PSC[7:0] PSC field value is 0x7.
3. Program the I2Ci.I2C_SCLL[7:0] SCLL and I2Ci.I2C_SCLH[7:0] SCLH fields to obtain a bit rate of 100K bps (maximum authorized bit rate in SCCB mode). This value depends on the internal sampling clock frequency (see Table 15-12).
4. Configure the 7-bit slave address (ID value) by storing it in the I2Ci.I2C_SA register.
5. Configure the 8-bit register address (subaddress) by storing it in the I2Ci.I2C_OA0 register.

6. Configure the I2Ci.I2C_BUF[13:8]RTRSH field to 0x0 (RX threshold to 1) and the I2Ci.I2C_BUF[5:0]XTRSH field to 0x0 (TX threshold to 1).
7. Take the I2C controller out of reset by setting the I2Ci.I2C_CON[15] I2C_EN bit to 1.

15.5.2.1.2 Initialize the I2C Controller

To initialize the I2C controller, perform the following steps:

1. Configure the I2Ci.I2C_CON register:
   - In SCCB mode, only the master mode is supported; set the I2Ci.I2C_CON[10] MST bit to 1.
   - For transmitter mode (write to the external SCCB device register) or receiver mode (read from the external SCCB device register), set the I2Ci.I2C_CON[9] TRX bit (0: receiver, 1: transmitter).
2. Set the CONTROL.CONTROL_PADCONF_x.INPUTENABLE bit for the corresponding I2C module to achieve synchronous clock on the clock output.
3. If using an interrupt to transmit/receive data, set to 1 the corresponding bit in the I2Ci.I2C_IE register (the I2Ci.I2C_IE[4] XRDY_IE bit for the transmit interrupt, the I2Ci.I2C_IE[3] RRDY bit for the receive interrupt).

15.5.2.1.3 Initiate a Transfer

Poll the I2Ci.I2C_STAT[12] BB bit. If it is cleared to 0 (bus not busy), set the I2Ci.I2C_CON[0] STT bit to 1. Because a transfer allows the LH to write or read only a single byte to or from the external SCCB device, the transmission automatically stops at the end of the transfer. When the transfer is complete, the I2Ci.I2C_STAT[2] ARDY bit is set to 1. In SCCB mode, the I2Ci.I2C_CON[1] STP bit is not used.

15.5.2.1.4 Receive Data

Poll the I2Ci.I2C_STAT[3] RRDY bit, or use the RRDY interrupt (the I2Ci.I2C_IE[3] RRDY_IE bit must be set to 1) to read the receive data in the I2Ci.I2C_DATA register.

NOTE: In SCCB mode, the I2Ci.I2C_BUF[13:8] RTRSH field (RX threshold) must be set to a value of 0x0 (RX threshold = 1).

15.5.2.1.5 Transmit Data

Poll the I2Ci.I2C_STAT[4] XRDY bit, or use the XRDY interrupt (the I2Ci.I2C_IE[4] XRDY_IE bit must be set to 1) to write the data to the I2Ci.I2C_DATA register.

NOTE: In SCCB mode, the I2Ci.I2C_BUF[5:0] XTRSH field (TX threshold) must be set to a value of 0x0 (TX threshold = 1).

15.5.2.2 Interrupt Subroutine Sequence

1. Test for register access ready (I2Ci.I2C_STAT[2] ARDY status bit) and resolve accordingly.
2. Test for receive data ready (I2Ci.I2C_STAT[3] RRDY status bit) and resolve accordingly.
3. Test for transmit data ready (I2Ci.I2C_STAT[4] XRDY status bit) and resolve accordingly.

15.5.2.3 Programming Flow Diagrams

Figure 15-37 through Figure 15-41 are procedure flowcharts for programming the SCCB mode.

Figure 15-37. SCCB Setup Procedure

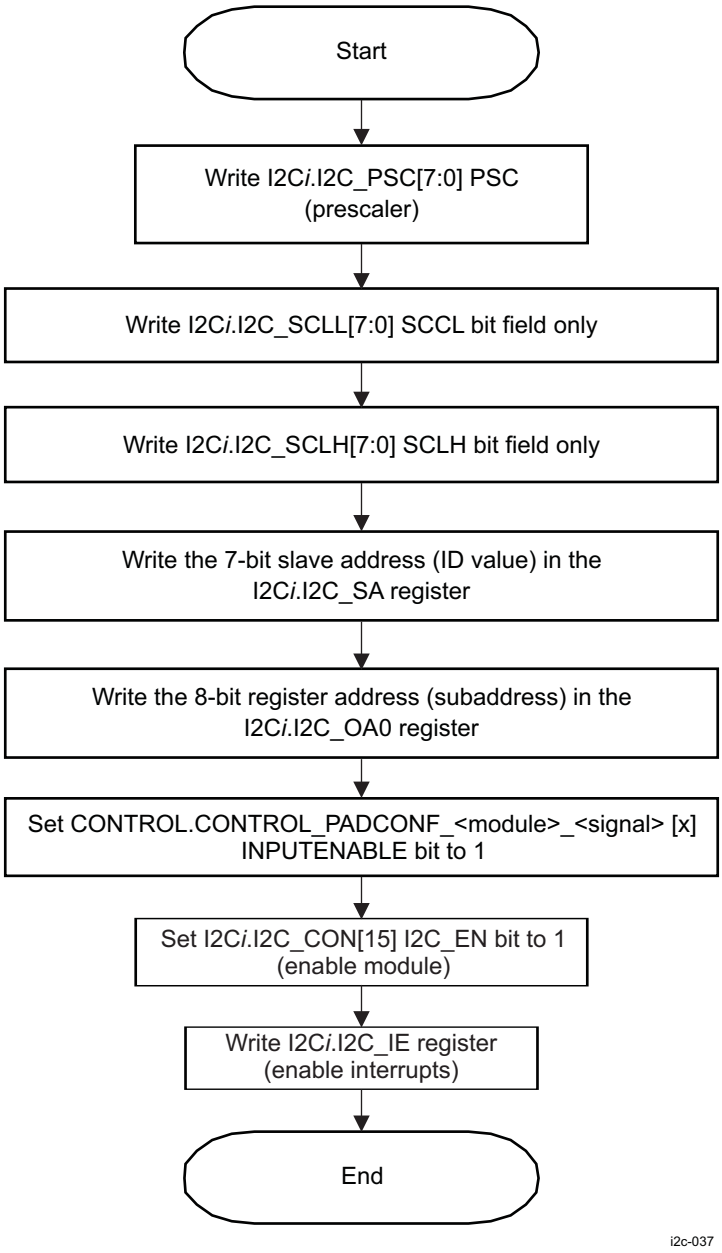
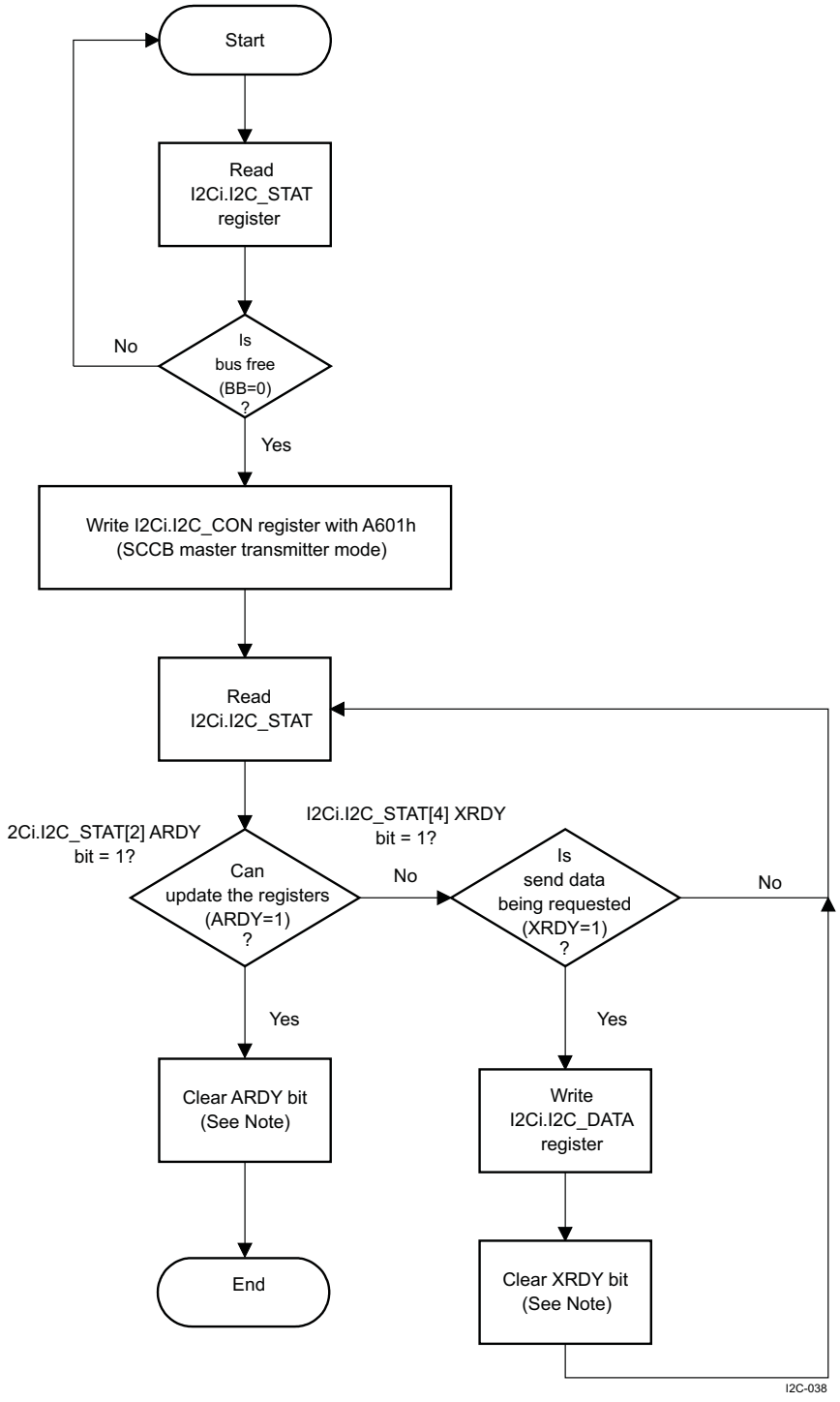
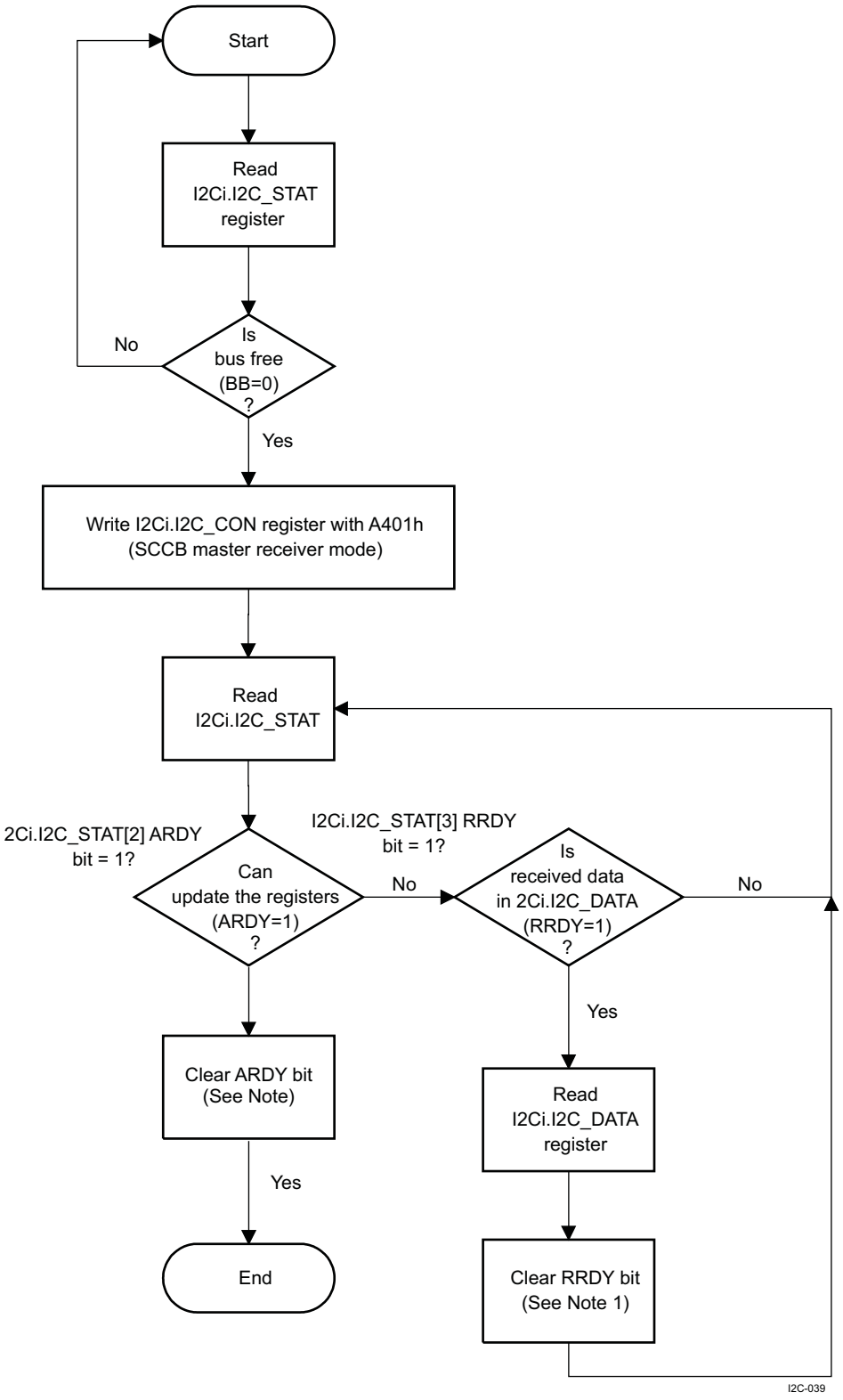


Figure 15-38. SCCB Master Transmitter Mode, Polling



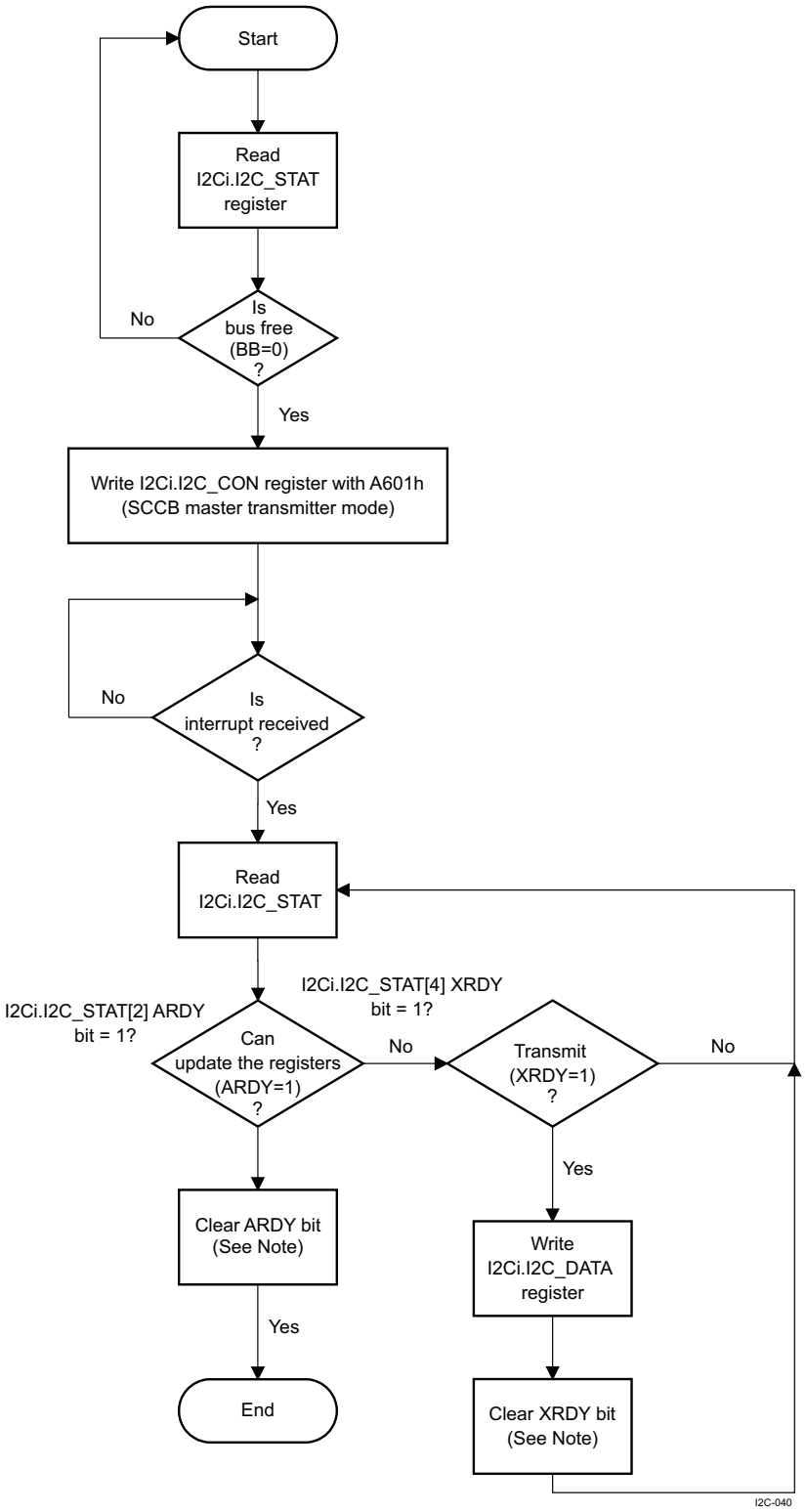
NOTE: The XRDY and ARDY bits are cleared by writing 1 to the corresponding bit in the I2Ci.I2C_STAT register.

Figure 15-39. SCCB Master Receiver Mode, Polling



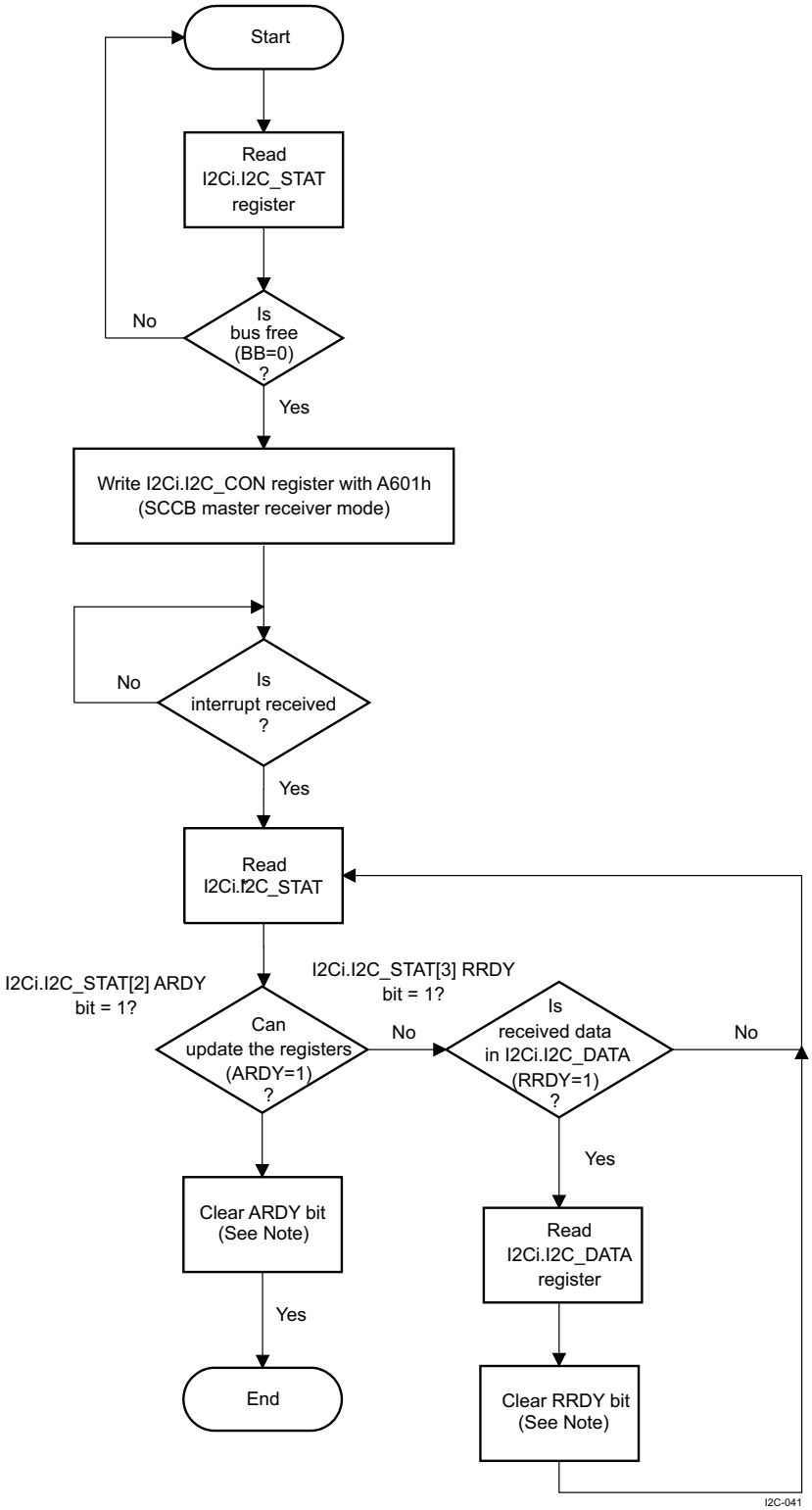
NOTE: The RRDY and ARDY bits are cleared by writing 1 in the corresponding bit in the I2Ci.I2C_STAT register.

Figure 15-40. SCCB Master Transmitter Mode, Interrupt



NOTE: The XRDY and ARDY bits are cleared by writing 1 in the corresponding bit in the I2Ci.I2C_STAT register.

Figure 15-41. SCCB Master Receiver Mode, Interrupt



NOTE: The RRDY and ARDY bits are cleared by writing 1 in the corresponding bit in the I2Ci.I2C_STAT register.

15.5.3 Master Transmitter HS I²C Controller I2C4 Basic Programming Model for Communication With Power Chips

This section describes the programming model of the master transmitter HS I²C controller I2C4 for communication with one or more power chips. The HS I²C controller I2C4 allowsthe two voltage Finite State Machines (FSMs) in the PRCM module of the device to be interfaced to external power chips through the I²C bus for dynamic voltage control of two power supplies and power sequencing. The primary programming tasks are to set up the configuration registers according to the external power supply chip(s) being used.

15.5.3.1 Configure the Voltage Controller Registers

To use the voltage control function, the LH must configure the following registers in the voltage controller of the PRCM module:

- The voltage configuration register address for each VDD channel in the PRCM.PRM_VC_SMPS_RA register
- The ON/ON-low-power-Retention/OFF command configuration register address for each VDD channel in the PRCM.PRM_VC_SMPS_CMD_RA register
- The set of ON/ON-low-power/Retention/OFF voltage/mode values in the PRCM.PRM_VC_CMD_VAL_0 register for the first VDD channel and the PRCM.PRM_VC_CMD_VAL_1 register for the second VDD channel
- The VDD channels configuration selection in the PRCM.PRM_VC_CH_CONF register

For details about voltage controller register configuration, see , *Power, Reset, and Clock Management.*

15.5.3.2 Configure the Master Transmitter HS I²C Controller I2C4

At power-on reset, the I2C4 is in HS mode (PRCM.PRM_VC_I2C_CFG[3] HSEN bit). In HS mode, the LH must configure the master code value for the preamble I²C high-speed transmission by configuring the PRCM.PRM_VC_I2C_CFG[2:0] MCODE field. If the external power chips do not support the I²C HS mode, the HS mode can be disabled and F/S mode enabled by clearing the PRCM.PRM_VC_I2C_CFG[3] HSEN bit to 0.

By default, the repeated start operation is enabled (PRCM.PRM_VC_I2C_CFG[4] SREN bit set to 1 at power-on reset). In F/S mode, the repeated start operation can be disabled by clearing the PRCM.PRM_VC_I2C_CFG[4] SREN bit to 0.

15.5.3.3 Configure the External Power Chip(s)

The LH can send configuration commands from one bypass input to the external power chip(s), using the same I²C interface. To send a configuration command, the LH must process as follows:

- Check whether the transmission of a configuration command is ongoing by polling the PRCM.PRM_VC_BYPASS_VAL[24]VALID bit (1: a transmission is ongoing, 0: a new transmission can start).
- Set the external power chip 7-bit slave address in the PRCM.PRM_VC_BYPASS_VAL[6:0] SLAVEADDR field, set the configuration register address of the external power chip in the PRCM.PRM_VC_BYPASS_VAL[15:8] REGADDR field, and set the data to be written to the external power chip configuration register in the PRCM.PRM_VC_BYPASS_VAL[23:16] DATA field.
- Set the PRCM.PRM_VC_BYPASS_VAL[24]VALID bit to 1 to send the configuration command to the external power chip.

15.6 High-Speed I²C Controllers Register Manual

CAUTION

The I2Ci registers are limited to 16 bit and 8 bit data accesses, 32 bit data access is not allowed and can corrupt register content.

NOTE: For details about the master transmitter HS I²C controller I2C4, see , *Power, Reset, and Clock Management*.

Table 15-14 provides the instance summary.

Table 15-14. Instance Summary

Module Name	Base Address	Size
I2C1	0x4807 0000	128 bytes
I2C2	0x4807 2000	128 bytes
I2C3	0x4806 0000	128 bytes

15.6.1 Multimaster HS I²C Controller Register Mapping Summary

Table 15-15 lists the I2Ci registers (where i = 1, 2 and 3).

Table 15-15. Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address for I2C1	Physical Address for I2C2	Physical Address for I2C3
I2C_REV	R	32	0x00	0x4807 0000	0x4807 2000	0x4806 0000
I2C_IE	RW	32	0x04	0x4807 0004	0x4807 2004	0x4806 0004
I2C_STAT	RW	32	0x08	0x4807 0008	0x4807 2008	0x4806 0008
I2C_WE	RW	32	0x0C	0x4807 000C	0x4807 200C	0x4806 000C
I2C_SYSS	R	32	0x10	0x4807 0010	0x4807 2010	0x4806 0010
I2C_BUF	RW	32	0x14	0x4807 0014	0x4807 2014	0x4806 0014
I2C_CNT	RW	32	0x18	0x4807 0018	0x4807 2018	0x4806 0018
I2C_DATA	RW	32	0x1C	0x4807 001C	0x4807 201C	0x4806 001C
I2C_SYSC	RW	32	0x20	0x4807 0020	0x4807 2020	0x4806 0020
I2C_CON	RW	32	0x24	0x4807 0024	0x4807 2024	0x4806 0024
I2C_OA0	RW	32	0x28	0x4807 0028	0x4807 2028	0x4806 0028
I2C_SA	RW	32	0x2C	0x4807 002C	0x4807 202C	0x4806 002C
I2C_PSC	RW	32	0x30	0x4807 0030	0x4807 2030	0x4806 0030
I2C_SCLL	RW	32	0x34	0x4807 0034	0x4807 2034	0x4806 0034
I2C_SCLH	RW	32	0x38	0x4807 0038	0x4807 2038	0x4806 0038
I2C_SYSTEST	RW	32	0x3C	0x4807 003C	0x4807 203C	0x4806 003C
I2C_BUFSTAT	R	32	0x40	0x4807 0040	0x4807 2040	0x4806 0040
I2C_OA1	RW	32	0x44	0x4807 0044	0x4807 2044	0x4806 0044
I2C_OA2	RW	32	0x48	0x4807 0048	0x4807 2048	0x4806 0048
I2C_OA3	RW	32	0x4C	0x4807 004C	0x4807 204C	0x4806 004C
I2C_ACTOA	R	32	0x50	0x4807 0050	0x4807 2050	0x4806 0050
I2C_SBLOCK	RW	32	0x54	0x4807 0054	0x4807 2054	0x4806 0054

15.6.2 Register Description

Table 15-16. I2C_REV

Address Offset	0x00		
Physical Address	0x4806 0000	**Instance**	I2C3
	0x4807 0000		I2C1
	0x4807 2000		I2C2
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								REV							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:8	Reserved	Read returns 0.	R	0x00
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x30 for 3.0, 0x31 for 3.1	R	See (1)

(1) TI internal data

Table 15-17. Register Call Summary for Register I2C_REV

High-Speed I2C Controllers Register Manual

- Multimaster HS I2C Controller Register Mapping Summary: [0]

Table 15-18. I2C_IE

Address Offset	0x04		
Physical Address	0x4806 0004	**Instance**	I2C3
	0x4807 0004		I2C1
	0x4807 2004		I2C2
Description	This register contains the interrupt enable bits.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved	XDR_IE	RDR_IE	Reserved			AAS_IE	BF_IE	AERR_IE	STC_IE	GC_IE	XRDY_IE	RRDY_IE	ARDY_IE	NACK_IE	AL_IE

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
14	XDR_IE	Transmit Draining interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[XDR]. 0x0: Transmit Draining interrupt disabled 0x1: Transmit Draining interrupt enabled	RW	0
13	RDR_IE	Receive Draining interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[RDR]. 0x0: Receive Draining interrupt disabled	RW	0

Bits	Field Name	Description	Type	Reset
		0x1: Receive Draining interrupt enabled		
12:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
9	AAS_IE	Addressed as Slave interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[AAS]. 0x0: Addressed as Slave interrupt disabled 0x1: Addressed as Slave interrupt enabled	RW	0
8	BF_IE	Bus Free interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[BF]. 0x0: Bus Free interrupt disabled 0x1: Bus Free interrupt enabled	RW	0
7	AERR_IE	Access Error interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[AERR]. 0x0: Access Error interrupt disabled 0x1: Access Error interrupt enabled	RW	0
6	STC_IE	Start Condition interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[STC]. 0x0: Start Condition interrupt disabled 0x1: Start Condition interrupt enabled	RW	0
5	GC_IE	General Call interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[GC]. 0x0: General Call interrupt disabled 0x1: General Call interrupt enabled	RW	0
4	XRDY_IE	Transmit Data Ready interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[XRDY]. 0x0: Transmit Data Ready interrupt disabled 0x1: Transmit Data Ready interrupt enabled	RW	0
3	RRDY_IE	Receive Data Ready interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[RRDY]. 0x0: Receive Data Ready interrupt disabled 0x1: Receive Data Ready interrupt enabled	RW	0
2	ARDY_IE	Register Access Ready interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[ARDY]. 0x0: Register Access Ready interrupt disabled 0x1: Register Access Ready interrupt enabled	RW	0
1	NACK_IE	No Acknowledgment interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[NACK]. 0x0: No Acknowledge interrupt disabled 0x1: No Acknowledge Ready interrupt enabled	RW	0
0	AL_IE	Arbitration Lost interrupt enable. Mask or unmask the interrupt signaled by the bit in I2C_STAT[AL]. 0x0: Arbitration Lost interrupt disabled 0x1: Arbitration Lost interrupt enabled	RW	0

Table 15-19. Register Call Summary for Register I2C_IE

High-Speed I2C Controller Integration • Interrupt Requests: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]
High-Speed I2C Controller Functional Description • Receive Mode in I2C Mode: [15] • FIFO Interrupt Mode Operation: [16] • FIFO Polling Mode Operation: [17] [18] • Draining Feature (I2C Mode Only): [19] [20]

Table 15-19. Register Call Summary for Register I2C_IE (continued)

High-Speed I2C Controller Basic Programming Model • Initialize the I2C Controller: [21] [22] [23] • Receive Data: [24] [25] [26] • Transmit Data: [27] [28] [29] • Initialize the I2C Controller: [30] [31] [32] • Receive Data: [33] • Transmit Data: [34]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [35]

Table 15-20. I2C_STAT

Address Offset	0x08		
Physical Address	0x4806 0008	**Instance**	I2C3
	0x4807 0008		I2C1
	0x4807 2008		I2C2
Description	This register provides specific status information about the module, including interrupt status information.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved	XDR	RDR	BB	ROVR	XUDF	AAS	BF	AERR	STC	GC	XRDY	RRDY	ARDY	NACK	AL

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
14	XDR	Transmit Draining IRQ status Read 0x0: Transmit draining inactive Read 0x1: Transmit draining active Write 0x0: No effect Write 0x1: Clear this bit to 0	RW	0
13	RDR	Receive Draining IRQ status Read 0x0: Receive draining inactive Read 0x1: Receive draining active Write 0x0: No effect Write 0x1: Clear this bit to 0	RW	0
12	BB	Bus busy status. Read-only bit. Writing this bit does not affect the read value. Read 0x0: Bus is free. Read 0x1: Bus is occupied.	R	0
11	ROVR	Receive overrun status. Read-only bit. Writing this bit does not affect the read value. Read 0x0: Normal operation Read 0x1: Receiver overrun	R	0
10	XUDF	Transmit underflow status. Read-only bit. Writing this bit does not affect the read value. Read 0x0: Normal operation Read 0x1: Transmit underflow	R	0
9	AAS	Address recognized as slave IRQ status Read 0x0: No action Read 0x1: Address recognized	RW	0

Bits	Field Name	Description	Type	Reset
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
8	BF	Bus Free IRQ status	RW	0
		Read 0x0: No action		
		Read 0x1: Bus free		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
7	AERR	Access Error IRQ status	RW	0
		Read 0x0: No action		
		Read 0x1: Access error		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
6	STC	Start Condition IRQ status	RW	0
		Read 0x0: No action		
		Read 0x1: Start condition detected		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
5	GC	General Call IRQ status. Set to 1 when general call address detected. Interrupt signaled to MPU subsystem	RW	0
		Read 0x0: No general call detected		
		Read 0x1: General call address detected		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
4	XRDY	Transmit Data Ready IRQ status. Set to 1 in transmit mode when new data is requested for transmission. When this bit is set to 1, an interrupt is signaled to MPU subsystem	RW	0
		Read 0x0: No transmit data is requested for transmission		
		Read 0x1: Transmit data is requested for transmission		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
3	RRDY	Receive Data Ready IRQ status. Set to 1 when in receive mode, causing new data to be able to be read. When this bit is set to 1, an interrupt is signaled to MPU subsystem.	RW	0
		Read 0x0: No data available		
		Read 0x1: Receive data available		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
2	ARDY	Register Access Ready IRQ status. Setting this bit to 1 indicates that previous access has been performed and registers are ready to be accessed again. An interrupt is signaled to MPU subsystem.	RW	0
		Read 0x0: Module busy		
		Read 0x1: Access ready		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		
1	NACK	No Acknowledgment IRQ status. This bit is set when No Acknowledgment has been received. An interrupt is signaled to MPU subsystem. In master mode, the transfer is automatically ended by generating a stop condition on the bus. The I2Ci.I2C_CON[1] STP, I2Ci.I2C_CON[10] MST and I2Ci.I2C_CON[9] TRX bits are automatically cleared to 0 (slave receiver mode). TX and RX FIFOs must be cleared (I2Ci.I2C_BUF[6] TXFIFO_CLR and I2Ci.I2C_BUF[14] RXFIFO_CLR bits set to 1).	RW	0
		Read 0x0: Normal operation		
		Read 0x1: No Acknowledge detected		
		Write 0x0: No effect		
		Write 0x1: Clear this bit to 0.		

Bits	Field Name	Description	Type	Reset
0	AL	Arbitration Lost IRQ status. This bit is set automatically by the hardware when the I2C controller inside the device loses arbitration in master transmit mode. An interrupt is signaled to MPU subsystem. Read 0x0: Normal operation Read 0x1: Arbitration loss detected Write 0x0: No effect Write 0x1: Clear this bit to 0.	RW	0

Table 15-21. Register Call Summary for Register I2C_STAT

High-Speed I2C Controller Environment
- Start and Stop Conditions: [0] [1]
- Master Transmitter: [2]
- Master Receiver: [3]
- Slave Transmitter: [4]
- Slave Receiver: [5]
- Bus Arbitration: [6]

High-Speed I2C Controller Integration
- Interrupt Requests: [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22]

High-Speed I2C Controller Functional Description
- Transmit Mode in I2C Mode: [23] [24]
- Receive Mode in I2C Mode: [25] [26] [27] [28]
- FIFO Interrupt Mode Operation: [29] [30]
- FIFO Polling Mode Operation: [31] [32] [33] [34] [35] [36]
- Draining Feature (I2C Mode Only): [37] [38] [39] [40] [41]
- System Test Mode: [42] [43] [44]

High-Speed I2C Controller Basic Programming Model
- Initiate a Transfer: [45]
- Receive Data: [46] [47] [48]
- Transmit Data: [49] [50] [51]
- Interrupt Subroutine Sequence: [52] [53] [54] [55] [56] [57] [58] [59] [60]
- Programming Flow Diagrams: [61] [62] [63] [64] [65] [66] [67] [68]
- Initiate a Transfer: [69] [70]
- Receive Data: [71]
- Transmit Data: [72]
- Interrupt Subroutine Sequence: [73] [74] [75]
- Programming Flow Diagrams: [76] [77] [78] [79]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [91]
- Register Description: [92] [93] [94] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106]

Table 15-22. I2C_WE

Address Offset	0x0C		
Physical Address	0x4806 000C	**Instance**	I2C3
	0x4807 000C		I2C1
	0x4807 200C		I2C2
Description	This register contains the wakeup enable bits.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved	XDR_WE	RDR_WE	Reserved			AAS_WE	BF_WE	Reserved	STC_WE	GC_WE	Reserved	DRDY_WE	ARDY_WE	NACK_WE	AL_WE

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
14	XDR_WE	Transmit draining wakeup enable	RW	0
		0x0: Transmit draining wakeup disabled		
		0x1: Transmit draining wakeup enabled		
13	RDR_WE	Receive draining wakeup enable	RW	0
		0x0: Receive draining wakeup disabled		
		0x1: Receive draining wakeup enabled		
12:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
9	AAS_WE	Address as slave wakeup enabled	RW	0
		0x0: Address as slave wakeup disabled		
		0x1: Address as slave wakeup enabled		
8	BF_WE	Bus Free wakeup enable	RW	0
		0x0: Bus Free wakeup disabled		
		0x1: Bus Free wakeup enabled		
7	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
6	STC_WE	Start Condition wakeup enable	RW	0
		0x0: Start condition wakeup disabled		
		0x1: Start condition wakeup enabled		
5	GC_WE	General call wakeup enable	RW	0
		0x0: General call wakeup disabled		
		0x1: General call wakeup enabled		
4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
3	DRDY_WE	Transmit/receive data ready wakeup enable. Mask or unmask the interrupt signaled by bit in I2C_STAT[RRDY]	RW	0
		0x0: Transmit/receive data ready wakeup disabled		
		0x1: Transmit/receive data ready wakeup enabled		
2	ARDY_WE	Register access ready wakeup enable	RW	0
		0x0: Register access ready wakeup disabled		
		0x1: Register access ready wakeup enabled		
1	NACK_WE	No Acknowledgment wakeup enable	RW	0
		0x0: No Acknowledgment wakeup disabled		
		0x1: No Acknowledgment wakeup enabled		
0	AL_WE	Arbitration lost wakeup enable	RW	0
		0x0: Arbitration lost wakeup disabled		
		0x1: Arbitration lost wakeup enabled		

Table 15-23. Register Call Summary for Register I2C_WE

High-Speed I2C Controller Integration
• Wake-Up Capability: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]
High-Speed I2C Controllers Register Manual
• Multimaster HS I2C Controller Register Mapping Summary: [12]

Table 15-24. I2C_SYSS

Address Offset	0x10		
Physical Address	0x4806 0010	**Instance**	I2C3
	0x4807 0010		I2C1
	0x4807 2010		I2C2
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								Reserved							RDONE

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Read returns 0.	R	0x0000
15:8	Reserved	Read returns 0.	R	0x00
7:1	Reserved	Read returns 0.	R	0x00
0	RDONE	Internal reset monitoring	R	0
		Read 0x0: Internal module reset in ongoing		
		Read 0x1: Internal module reset complete		

Table 15-25. Register Call Summary for Register I2C_SYSS

High-Speed I2C Controller Integration
- Software Reset: [0] [1] [2]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [3]

Table 15-26. I2C_BUF

Address Offset	0x14		
Physical Address	0x4806 0014	**Instance**	I2C3
	0x4807 0014		I2C1
	0x4807 2014		I2C2
Description	Receive DMA channel disabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RDMA_EN	RXFIFO_CLR	RTRSH						XDMA_EN	TXFIFO_CLR	XTRSH					

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	RDMA_EN	Receive DMA channel enable	RW	0
		0x0: Receive DMA channel disabled		
		0x1: Receive DMA channel enabled		
14	RXFIFO_CLR	Receive FIFO clear	RW	0
		0x0: Normal mode		

Bits	Field Name	Description	Type	Reset
		0x1: Rx FIFO is reset		
13:8	RTRSH	Threshold value for FIFO buffer in RX mode is equal to RTRSH + 1.	RW	0x00
7	XDMA_EN	Transmit DMA channel enable	RW	0
		0x0: Transmit DMA channel disabled		
		0x1: Transmit DMA channel enabled		
6	TXFIFO_CLR	Transmit FIFO clear	RW	0
		0x0: Normal mode		
		0x1: Tx FIFO is reset		
5:0	XTRSH	Threshold value for FIFO buffer in TX mode is equal to XTRSH + 1.	RW	0x00

Table 15-27. Register Call Summary for Register I2C_BUF

High-Speed I2C Controller Integration
- Wake-Up Capability: [0] [1]
- Interrupt Requests: [2] [3] [4] [5] [6] [7] [8] [9]

High-Speed I2C Controller Functional Description
- Receive Mode in I2C Mode: [10]
- FIFO Interrupt Mode Operation: [11] [12] [13] [14] [15] [16] [17] [18]
- FIFO Polling Mode Operation: [19] [20]
- FIFO DMA Mode Operation (I2C Mode Only): [21] [22] [23] [24] [25] [26] [27]
- Draining Feature (I2C Mode Only): [28] [29] [30] [31] [32] [33] [34] [35]
- Write and Read Operations in SCCB Mode: [36] [37]

High-Speed I2C Controller Basic Programming Model
- Configure the Module Before Enabling the I2C Controller: [38] [39]
- Initialize the I2C Controller: [40] [41] [42]
- Receive Data: [43] [44]
- Transmit Data: [45] [46]
- Programming Flow Diagrams: [47] [48]
- Configure the Module Before Enabling the I2C Controller: [49] [50]
- Receive Data: [51]
- Transmit Data: [52]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [54]
- Register Description: [55] [56]

Table 15-28. I2C_CNT

Address Offset	0x18		
Physical Address	0x4806 0018	**Instance**	I2C3
	0x4807 0018		I2C1
	0x4807 2018		I2C2
Description	This read/write register is used to control the numbers of bytes in the I2C data payload.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																DCOUNT															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:0	DCOUNT	Data count **Note:** Because the transfer length for DCOUNT=0x0000 is 65536, the module does not allow the initiation of zero-data-byte transfers.	RW	0x0000

Table 15-29. Register Call Summary for Register I2C_CNT

High-Speed I2C Controller Integration • Interrupt Requests: [0] [1] [2] [3]
High-Speed I2C Controller Functional Description • Transmit Mode in I2C Mode: [4] • Draining Feature (I2C Mode Only): [5]
High-Speed I2C Controller Basic Programming Model • Configure Slave Address and the Data Control Register: [6] • Transmit Data: [7] • Programming Flow Diagrams: [8] [9] [10] [11] [12] [13]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [18]

Table 15-30. I2C_DATA

Address Offset	0x1C		
Physical Address	0x4806 001C	**Instance**	I2C3
	0x4807 001C		I2C1
	0x4807 201C		I2C2
Description	This register is the end point/entry point for the LH to read data from or write data to the FIFO buffer.		
	Read accesses from an empty FIFO (i.e. at reset) or write accesses to a full FIFO will return error.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								DATA							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
7:0	DATA	Transmit/Receive FIFO data	RW	0x–

Table 15-31. Register Call Summary for Register I2C_DATA

High-Speed I2C Controller Integration • DMA Requests: [0] [1] [2] [3] [4] [5] • Interrupt Requests: [6] [7]
High-Speed I2C Controller Functional Description • Transmit Mode in I2C Mode: [8] • Receive Mode in I2C Mode: [9] [10] • System Test Mode: [11] [12] • Write and Read Operations in SCCB Mode: [13] [14]
High-Speed I2C Controller Basic Programming Model • Receive Data: [15] [16] • Transmit Data: [17] [18] • Receive Data: [19] • Transmit Data: [20]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [22]

Table 15-32. I2C_SYSC

Address Offset	0x20		
Physical Address	0x4806 0020	**Instance**	I2C3
	0x4807 0020		I2C1
	0x4807 2020		I2C2
Description	This register controls the various parameters of the L4-Core interconnect interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved						CLOCKACTIVITY		Reserved			IDLEMODE		ENAWAKEUP	SRST	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
9:8	CLOCKACTIVITY	Clock Activity selection bits	RW	0x0
		0x0: Both clocks can be cut off		
		0x1: Only interface clock must be kept active; functional clock can be cut off		
		0x2: Only functional clock must be kept active; interface clock can be cut off		
		0x3: Both clocks must be kept active		
7:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
4:3	IDLEMODE	Idle Mode selection bits	RW	0x0
		0x0: Force Idle mode		
		0x1: No Idle mode		
		0x2: Smart Idle mode		
		0x3: Reserved		
2	ENAWAKEUP	Enable wakeup control bit	RW	0
		0x0: Wakeup mechanism is disabled		
		0x1: Wakeup mechanism is enabled		
1	SRST	Software reset. This bit is automatically reset by the hardware. During reads, it always returns 0.	RWI	0
		0x0: Normal mode		
		0x1: The module is reset		
0	AUTOIDLE	Auto Idle enable control bit	RW	0
		0x0: Auto Idle mechanism is disabled		
		0x1: Auto Idle mechanism is enabled		

Table 15-33. Register Call Summary for Register I2C_SYSC

High-Speed I2C Controller Integration

- Module Power Saving: [0]
- System Power Management: [1] [2] [3] [4] [5] [6] [7] [8] [9]
- Wake-Up Capability: [10]
- Software Reset: [11] [12]

High-Speed I2C Controllers Register Manual

- Multimaster HS I2C Controller Register Mapping Summary: [13]

Table 15-34. I2C_CON

Address Offset	0x24		
Physical Address	0x4806 0024	**Instance**	I2C3
	0x4807 0024		I2C1
	0x4807 2024		I2C2
Description	This register controls the functional features. Caution: during an active transfer phase (STT has been set to 1), no modification must be done in this register. Changing it may result in unpredictable behavior.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																I2C_EN	Reserved	OPMODE		STB	MST	TRX	XSA	XOA0	XOA1	XOA2	XOA3	Reserved		STP	STT

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	I2C_EN	Module enable bit	RW	0
		0x0: Controller in reset. FIFO are cleared and status bits are set to their default value.		
		0x1: Module enabled		
14	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0
13:12	OPMODE	Operation mode selection	RW	0x0
		0x0: I2C Fast/Standard mode		
		0x1: I2C High Speed mode		
		0x2: SCCB mode		
		0x3: Reserved		
11	STB	Start byte mode (master mode only)	RW	0
		0x0: Normal mode		
		0x1: Start byte mode		
10	MST	Master/slave mode selection	RWl	0
		0x0: Slave mode		
		0x1: Master mode		
9	TRX	Transmitter/Receiver mode (master mode only)	RW	0
		0x0: Receiver mode		
		0x1: Transmitter mode		
8	XSA	Expand slave address enable bit	RW	0
		0x0: 7-bit address mode		
		0x1: 10-bit address mode		
7	XOA0	Expand Own Address 0 enable bit (default)	RW	0
		0x0: 7-bit address mode		
		0x1: 10-bit address mode		
6	XOA1	Expand Own Address 1 enable bit	RW	0
		0x0: 7-bit address mode		
		0x1: 10-bit address mode		
5	XOA2	Expand Own Address 2 enable bit	RW	0
		0x0: 7-bit address mode		
		0x1: 10-bit address mode		
4	XOA3	Expand Own Address 3 enable bit	RW	0
		0x0: 7-bit address mode		
		0x1: 10-bit address mode		

Bits	Field Name	Description	Type	Reset
3:2	Reserved	Write 0s for future compatibility. Read returns 0	RW	0x0
1	STP	Stop condition (master mode only). The STP bit is cleared by the module itself once it has generated and detected the programmed stop condition on the bus. 0x0: No action or generated stop (P) condition detected on the bus (by the module) 0x1: Stop condition queried	RW	0
0	STT	Start condition (master mode only). The STT bit is cleared by the module itself once it has generated and detected the programmed start condition on the bus. 0x0: No action or generated start (S) condition detected (by the module) 0x1: Start condition queried	RW	0

Table 15-35. Register Call Summary for Register I2C_CON

High-Speed I2C Controller Environment
- SCCB Interface Typical Connections: [0]

High-Speed I2C Controller Integration
- Software Reset: [1] [2] [3] [4] [5]
- Interrupt Requests: [6] [7] [8] [9] [10] [11] [12]

High-Speed I2C Controller Functional Description
- Block Diagram: [13] [14]
- Transmit Mode in I2C Mode: [15] [16]
- Receive Mode in I2C Mode: [17]
- FIFO Interrupt Mode Operation: [18] [19]
- Programmable Multislave Channel Feature (I2C Mode Only): [20] [21] [22] [23]
- Clocking: [24]
- System Test Mode: [25] [26] [27]
- Write and Read Operations in SCCB Mode: [28] [29] [30] [31]

High-Speed I2C Controller Basic Programming Model
- Configure the Module Before Enabling the I2C Controller: [32] [33]
- Initialize the I2C Controller: [34] [35] [36]
- Configure Slave Address and the Data Control Register: [37]
- Initiate a Transfer: [38] [39] [40] [41]
- Programming Flow Diagrams: [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65]
- Configure the Module Before Enabling the I2C Controller: [66]
- Initialize the I2C Controller: [67] [68] [69]
- Initiate a Transfer: [70] [71]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [95]
- Register Description: [96] [97] [98]

Table 15-36. I2C_OA0

Address Offset	0x28		
Physical Address	0x4806 0028	**Instance**	I2C3
	0x4807 0028		I2C1
	0x4807 2028		I2C2
Description	This register is used to specify the module I2C 7-bit or 10-bit address for the I2C operations or the 8-bit subaddress of the SCCB module register for the SCCB operations.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MCODE			Reserved						OA							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:13	MCODE	Master C ode value	RW	0x0
12:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0
9:0	OA	Own address 0 value	RW	0x000

Table 15-37. Register Call Summary for Register I2C_OA0

High-Speed I2C Controller Functional Description • Write and Read Operations in SCCB Mode: [0] [1]
High-Speed I2C Controller Basic Programming Model • Configure the Module Before Enabling the I2C Controller: [2] • Configure the Module Before Enabling the I2C Controller: [3]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [4]

Table 15-38. I2C_SA

Address Offset	0x2C		
Physical Address	0x4806 002C	**Instance**	I2C3
	0x4807 002C		I2C1
	0x4807 202C		I2C2
Description	This register is used to specify the addressed I2C module 7-bit or 10-bit address for the I2C operations or the 7-bit address of the external SCCB module for the SCCB operations.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								SA							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
9:0	SA	Slave address value.	RW	0x3FF

Table 15-39. Register Call Summary for Register I2C_SA

High-Speed I2C Controller Functional Description • Write and Read Operations in SCCB Mode: [0] [1]
High-Speed I2C Controller Basic Programming Model • Configure Slave Address and the Data Control Register: [2] • Programming Flow Diagrams: [3] [4] [5] [6] [7] [8] • Configure the Module Before Enabling the I2C Controller: [9]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [16]

Table 15-40. I2C_PSC

Address Offset	0x30		
Physical Address	0x4806 0030	**Instance**	I2C3
	0x4807 0030		I2C1
	0x4807 2030		I2C2
Description	This register is used to specify the internal clocking of the I2C peripheral core. The core logic is sampled at the clock rate of the functional clock for the module divided by (PSC+1).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								PSC							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
7:0	PSC	Fast/Standard and SCCB modes prescale sampling clock divider value	RW	0x00

Table 15-41. Register Call Summary for Register I2C_PSC

High-Speed I2C Controller Integration
- Module Clocks: [0] [1] [2]

High-Speed I2C Controller Functional Description
- Clocking: [3] [4]
- Noise Filter: [5] [6]
- System Test Mode: [7]

High-Speed I2C Controller Basic Programming Model
- Configure the Module Before Enabling the I2C Controller: [8] [9]
- Configure the Module Before Enabling the I2C Controller: [10] [11]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [15]

Table 15-42. I2C_SCLL

Address Offset	0x34		
Physical Address	0x4806 0034	**Instance**	I2C3
	0x4807 0034		I2C1
	0x4807 2034		I2C2
Description	This register is used to determine the SCL low time value when master.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HSSCLL								SCLL							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	HSSCLL	I²C High Speed mode SCL low time value.	RW	0x00
7:0	SCLL	I²C Fast/Standard or SCCB modes SCL low time value	RW	0x00

Table 15-43. Register Call Summary for Register I2C_SCLL

High-Speed I2C Controller Functional Description • Clocking: [0] [1] [2] [3] [4] [5] [6] • System Test Mode: [7]
High-Speed I2C Controller Basic Programming Model • Configure the Module Before Enabling the I2C Controller: [8] [9] • Configure the Module Before Enabling the I2C Controller: [10]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [14]

Table 15-44. I2C_SCLH

Address Offset	0x38		
Physical Address	0x4806 0038	**Instance**	I2C3
	0x4807 0038		I2C1
	0x4807 2038		I2C2
Description	This register is used to determine the SCL low time value when master.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HSSCLH								SCLH							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:8	HSSCLH	I²C high-speed mode SCL high time value	RW	0x00
7:0	SCLH	I²C Fast/Standard or SCCB modes SCL high time value	RW	0x00

Table 15-45. Register Call Summary for Register I2C_SCLH

High-Speed I2C Controller Functional Description • Clocking: [0] [1] [2] [3] [4] [5] [6] • System Test Mode: [7]
High-Speed I2C Controller Basic Programming Model • Configure the Module Before Enabling the I2C Controller: [8] [9] • Configure the Module Before Enabling the I2C Controller: [10]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [14]

Table 15-46. I2C_SYSTEST

Address Offset	0x3C		
Physical Address	0x4806 003C	**Instance**	I2C3
	0x4807 003C		I2C1
	0x4807 203C		I2C2
Description	This register is used to facilitate system-level tests by overriding some of the standard functional features of the peripheral.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																ST_EN	FREE	TMODE		SSB	Reserved						SCCBE_O	SCL_I	SCL_O	SDA_I	SDA_O

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15	ST_EN	System test enable	RW	0
		0x0: Normal mode		
		0x1: System test enabled. Permit other system test registers bits to be set		
14	FREE	Free-running mode	RW	0
		0x0: Stop mode (on breakpoint condition). If Master mode, it stops after completion of the ongoing bit transfer. In slave mode, it stops during the phase transfer when 1 byte is completely transmitted/received.		
		0x1: Free-running mode		
13:12	TMODE	Test mode select	R	0
		0x0: Functional mode (default)		
		0x1: Reserved		
		0x2: Test of SCL counters (SCLL, SCLH, PSC). SCL provides a permanent clock with master mode.		
		0x3: Loop back mode select + SDA/SCL IO mode select		
11	SSB	Set status bits	R	0
		0x0: No action		
		0x1: Set all interrupt status bits to 1		
10:5	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
4	SCCBE_O	SCCBE line sense output value. Writing is possible only if ST_EN bit is set to 1	RW	0
		0x0: Write 0 to SCCBE line		
		0x1: Write 1 to SCCBE line		
3	SCL_I	SCL line sense input value	R	0
		0x0: Read 0 from SCL line		
		0x1: Read 1 from SCL line		
2	SCL_O	SCL line drive output value. Writing is possible only if ST_EN bit is set to 1	RW	0
		0x0: Write 0 to SCL line		
		0x1: Write 1 to SCL line		
1	SDA_I	SDA line sense input value	R	0
		0x0: Read 0 from SDA line		
		0x1: Read 1 from SDA line		
0	SDA_O	SDA line drive output value. Writing is possible only if ST_EN bit is set to 1	RW	0
		0x0: Write 0 to SDA line		
		0x1: Write 1 to SDA line		

Table 15-47. Register Call Summary for Register I2C_SYSTEST

High-Speed I2C Controller Functional Description • System Test Mode: [0] [1] [2] [3] [4] [5] [6] [7] [8]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [9]

Table 15-48. I2C_BUFSTAT

Address Offset	0x40		
Physical Address	0x4806 0040	**Instance**	I2C3
	0x4807 0040		I2C1
	0x4807 2040		I2C2
Description	This register contains the FIFO status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FIFODEPTH		RXSTAT						Reserved						TXSTAT	

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0000
15:14	FIFODEPTH (1)	FIFO depth indication. Read 0x0: 8-bytes FIFO Read 0x1: 16-bytes FIFO Read 0x2: 32-bytes FIFO Read 0x3: 64-bytes FIFO	R	0x-
13:8	RXSTAT	RX Buffer Status. It indicates the number of bytes to be read in the RX FIFO when the I2C_STAT[RDR] is asserted (set to 1). This indication is useful only in receiver mode when the draining feature is enabled.	R	0x00
7:6	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0
5:0	TXSTAT	TX Buffer Status. It indicates the number of bytes to be written in the TX FIFO when the I2C_STAT[XDR] is asserted (set to 1). This indication is useful only in transmitter mode when the draining feature is enabled.	R	0x00

(1) See Table 15-11

Table 15-49. Register Call Summary for Register I2C_BUFSTAT

High-Speed I2C Controller Integration
- Interrupt Requests: [0]

High-Speed I2C Controller Functional Description
- FIFO Management: [1]
- Draining Feature (I2C Mode Only): [2] [3] [4] [5] [6] [7] [8] [9]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [10]

Table 15-50. I2C_OA1

Address Offset	0x44		
Physical Address	0x4806 0044	**Instance**	I2C3
	0x4807 0044		I2C1
	0x4807 2044		I2C2
Description	This register is used to specify the module I2C 7-bit or 10-bit address.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								OA1							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
9:0	OA1	Own address 1 value	RW	0x000

Table 15-51. Register Call Summary for Register I2C_OA1

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [0]

Table 15-52. I2C_OA2

Address Offset	0x48		
Physical Address	0x4806 0048	**Instance**	I2C3
	0x4807 0048		I2C1
	0x4807 2048		I2C2
Description	This register is used to specify the module I2C 7-bit or 10-bit address.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								OA2							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
9:0	OA2	Own address 2 value	RW	0x000

Table 15-53. Register Call Summary for Register I2C_OA2

High-Speed I2C Controllers Register Manual

- Multimaster HS I2C Controller Register Mapping Summary: [0]

Table 15-54. I2C_OA3

Address Offset	0x4C		
Physical Address	0x4806 004C	**Instance**	I2C3
	0x4807 004C		I2C1
	0x4807 204C		I2C2
Description	This register is used to specify the module I2C 7-bit or 10-bit address.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved								OA3							

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:10	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x00
9:0	OA3	Own address 3 value	RW	0x000

Table 15-55. Register Call Summary for Register I2C_OA3

High-Speed I2C Controllers Register Manual

- Multimaster HS I2C Controller Register Mapping Summary: [0]

Table 15-56. I2C_ACTOA

Address Offset	0x50		
Physical Address	0x4806 0050	**Instance**	I2C3
	0x4807 0050		I2C1
	0x4807 2050		I2C2
Description	This register contains the accessed slave Own Address indicators.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved												OA3_ACT	OA2_ACT	OA1_ACT	OA0_ACT

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x0000
15:4	Reserved	Write 0s for future compatibility. Read returns 0.	R	0x000
3	OA3_ACT	Own Address 3 active Read 0x0: Own Address inactive Read 0x1: Own Address active	R	0
2	OA2_ACT	Own Address 2 active Read 0x0: Own Address inactive Read 0x1: Own Address active	R	0
1	OA1_ACT	Own Address 1 active Read 0x0: Own Address inactive Read 0x1: Own Address active	R	0
0	OA0_ACT	Own Address 0 active Read 0x0: Own Address inactive Read 0x1: Own Address active	R	0

Table 15-57. Register Call Summary for Register I2C_ACTOA

High-Speed I2C Controller Functional Description
- Programmable Multislave Channel Feature (I2C Mode Only): [0]

High-Speed I2C Controllers Register Manual
- Multimaster HS I2C Controller Register Mapping Summary: [1]

Table 15-58. I2C_SBLOCK

Address Offset	0x54		
Physical Address	0x4806 0054	**Instance**	I2C3
	0x4807 0054		I2C1
	0x4807 2054		I2C2
Description	This register controls the slave mode i2c bus lock features.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Reserved												OA3_EN	OA2_EN	OA1_EN	OA0_EN

Bits	Field Name	Description	Type	Reset
31:16	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x0000
15:4	Reserved	Write 0s for future compatibility. Read returns 0.	RW	0x000
3	OA3_EN	Enable I2C Clock Blocking for Own Address 3 0x0: I2C Clock Released 0x1: I2C Clock Blocked	RW	0
2	OA2_EN	Enable I2C Clock Blocking for Own Address 2 0x0: I2C Clock Released 0x1: I2C Clock Blocked	RW	0

Bits	Field Name	Description	Type	Reset
1	OA1_EN	Enable I2C Clock Blocking for Own Address 1 0x0: I2C Clock Released 0x1: I2C Clock Blocked	RW	0
0	OA0_EN	Enable I2C Clock Blocking for Own Address 0 0x0: I2C Clock Released 0x1: I2C Clock Blocked	RW	0

Table 15-59. Register Call Summary for Register I2C_SBLOCK

High-Speed I2C Controller Functional Description • Automatic Blocking of the I2C Clock Feature (I2C Mode Only): [0]
High-Speed I2C Controllers Register Manual • Multimaster HS I2C Controller Register Mapping Summary: [1]

Multichannel SPI

This chapter describes the four multichannel serial port interface (McSPI) modules.

Topic **Page**

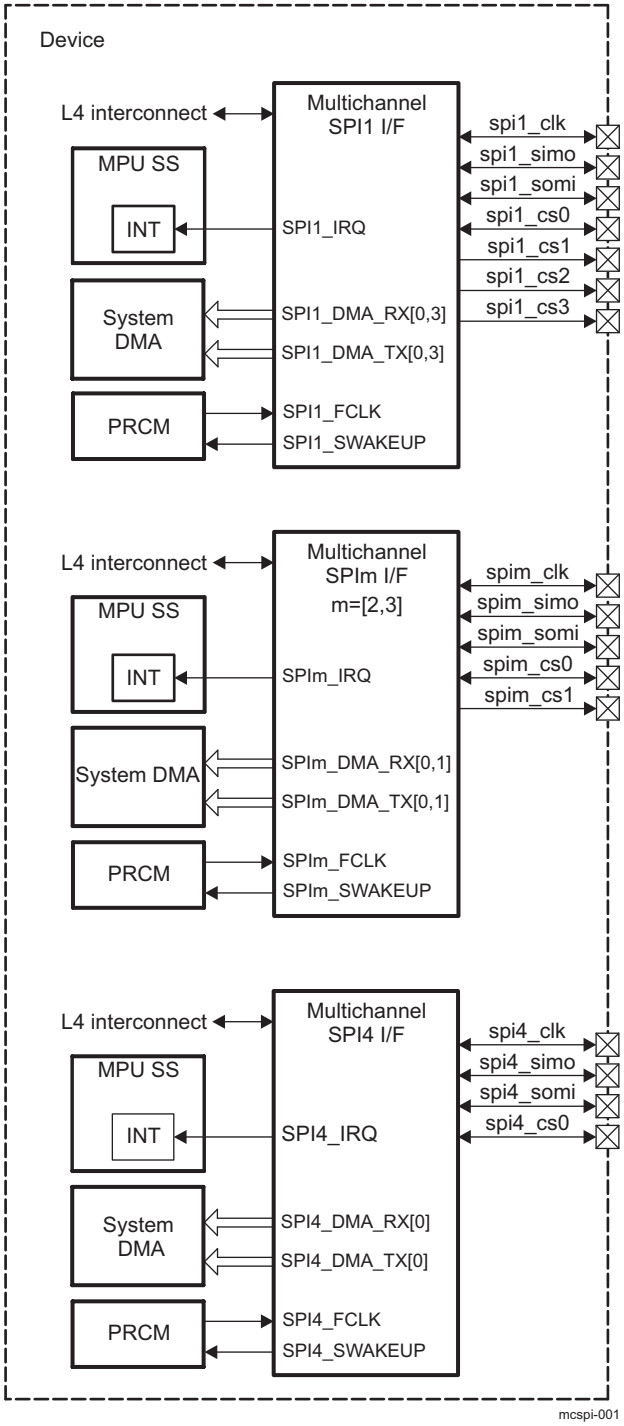
16.1 McSPI Overview

The multichannel serial port interface (McSPI) is a master/slave synchronous serial bus. There are four separate McSPI modules (SPI1, SPI2, SPI3, and SPI4) in the device (see Figure 16-1). The McSPI modules differ as follows: SPI1 supports up to four peripherals, SPI2 and SPI3 support up to two peripherals, and SPI4 supports only one peripheral.

NOTE: In this chapter, *m=[1,4]* represents the module instance and *x* represents the channel in signal and register naming. There are four McSPI instances. Each module instance has different channel numbers:

- SPI1: 4 channels (if *m=1*, *x=4*)
- SPI2: 2 channels (if *m=2*, *x=2*)
- SPI3: 2 channels (if *m=3*, *x=2*)
- SPI4: 1 channel (if *m=4*, *x=1*)

Figure 16-1. Multichannel Modules SPI1, SPI2, SPI3, and SPI4



The McSPI instances include the following main features:

- Serial clock with programmable frequency, polarity, and phase for each channel
- Wide selection of SPI word lengths ranging from 4 bits to 32 bits
- Up to four master channels or single channel in slave mode
- Master multichannel mode:
  - Full duplex/half duplex
  - Transmit-only/receive-only/transmit-and-receive modes

  - Flexible I/O port controls per channel
  - Two direct memory access (DMA) requests (read/write) per channel
- Single interrupt line for multiple interrupt source events
- Power management through wake-up capabilities
- Enable the addition of a programmable start-bit for SPI transfer per channel (start-bit mode)
- Support start-bit write command
- Support start-bit pause and break sequence
- 64 bytes built-in FIFO available for a single channel
- Force CS mode for continuous transfers

16.2 McSPI Environment

Figure 16-2 shows a simplified overview of a typical application system using the McSPI. This example is based on a TFS chipset (TFS6040-0098), including a 2.2-inch color-active matrix thin-film transistor (TFT) liquid crystal display (LCD) with a light-emitting diode (LED) front light, a four-wire resistive touch-screen panel, and LCD controllers. This chipset is associated with the TSC2005 or TSC2006 touch-screen controller, powered by a power-management unit, and driven by the device. The McSPI device interface is set to master mode; the touch-screen McSPI controller interface operates in slave mode.

Figure 16-2. Typical Application Using the McSPI

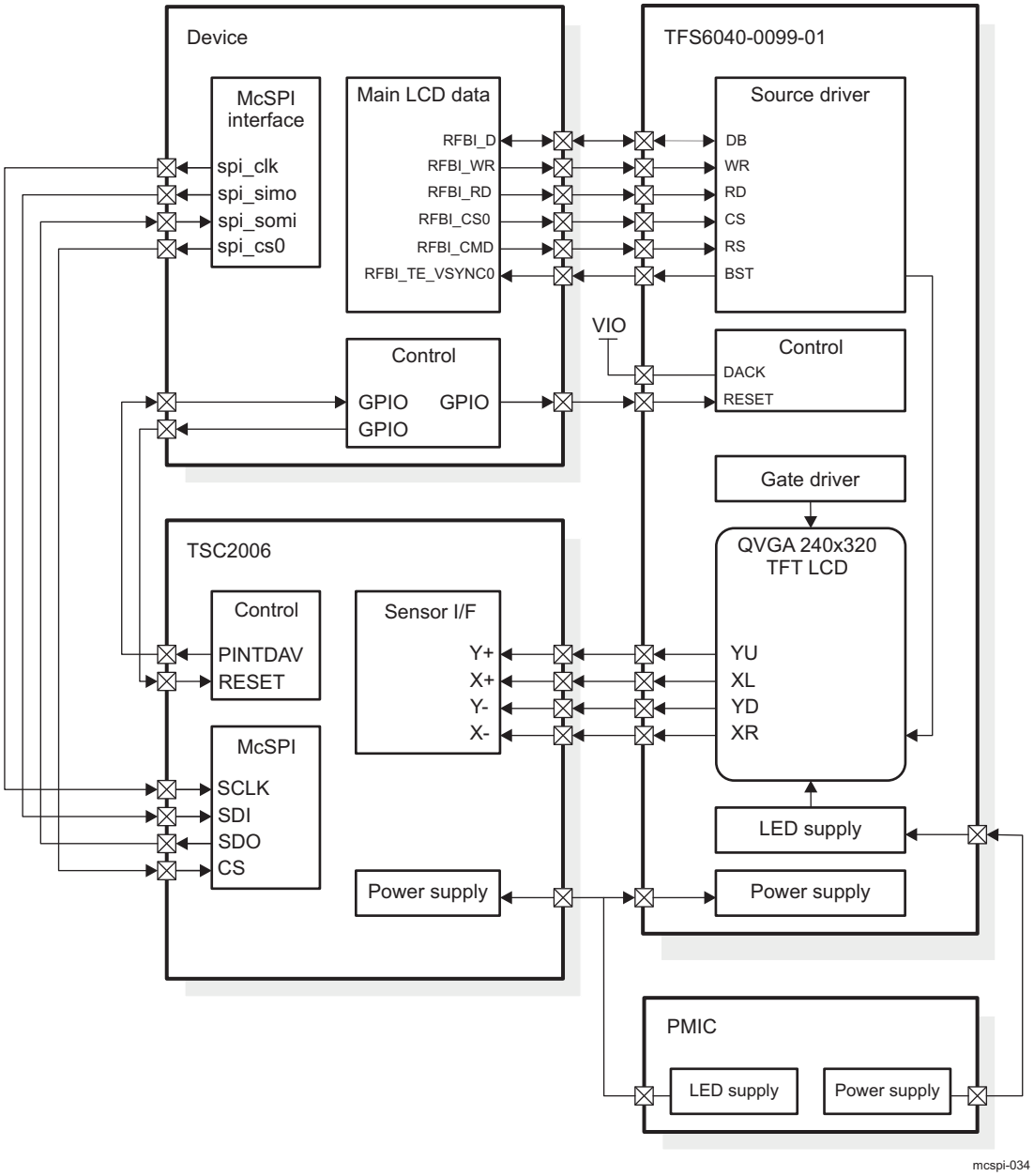
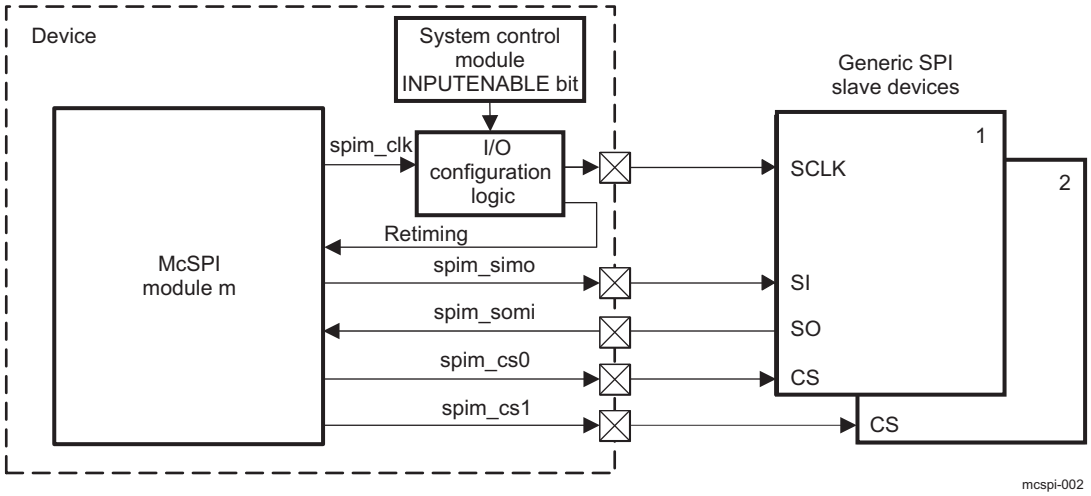


Figure 16-3 through Figure 16-8 show master mode and slave mode configurations. Each mode can be wired on a single-duplex or full-duplex configuration.

16.2.1 SPI Interface in Master Mode

Figure 16-3 shows a case in master mode (full-duplex) where the McSPI module is connected with two slave devices.

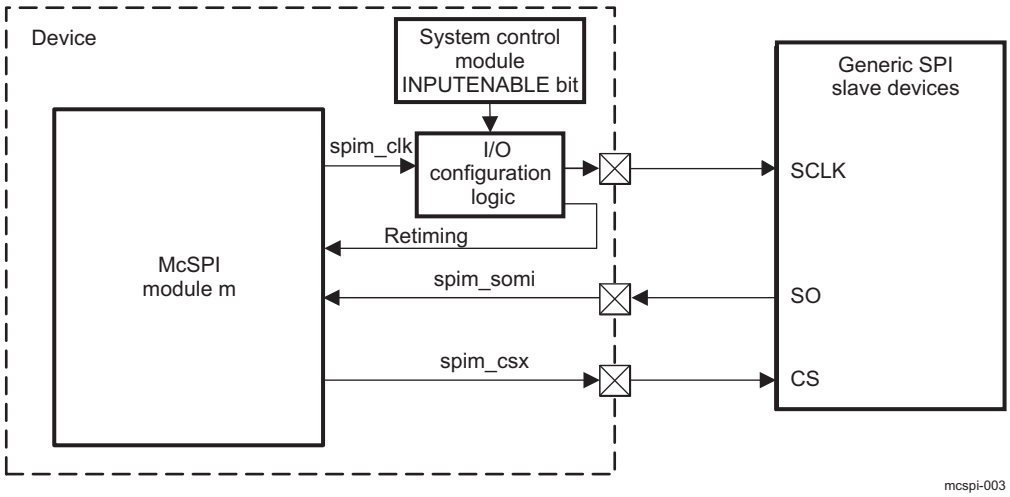
Figure 16-3. McSPI Master Mode (Full-Duplex)



NOTE: In this case m=[1,3].

Figure 16-4 shows the master single mode, which can also be configured in receive-only mode.

Figure 16-4. McSPI Master Single Mode (Receive-Only)



16.2.2 SPI Interface in Slave Mode

Figure 16-5 shows a case in slave mode (full-duplex).

NOTE: Only channel 0 can be configured as slave, and only spin_cs0 can be used as a chip-enable in slave mode.

Figure 16-5. McSPI Slave Mode (Full Duplex)

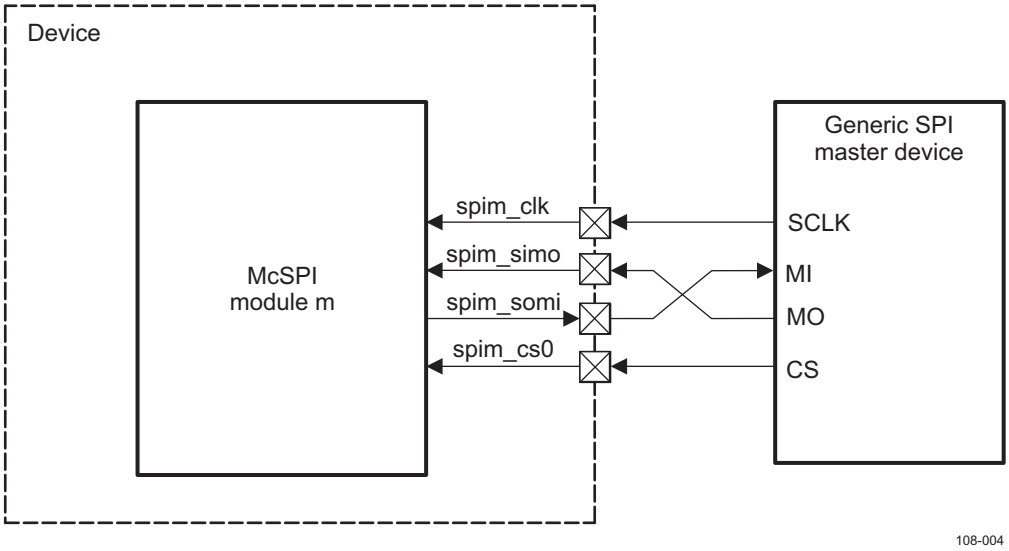
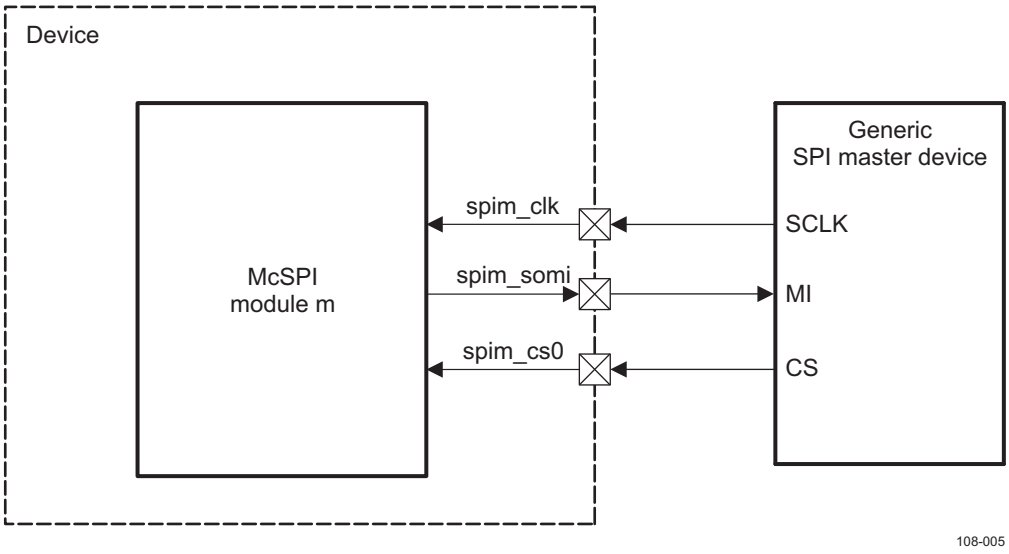


Figure 16-6 shows the slave single mode, which can also be configured in transmit-only mode.

Figure 16-6. McSPI Slave Single Mode (Transmit Only)



16.3 McSPI Functional Interface

16.3.1 Basic McSPI Pins for Master Mode

Figure 16-7 shows all of the McSPI interface signals in master mode.

Figure 16-7. McSPI Interface Signals in Master Mode

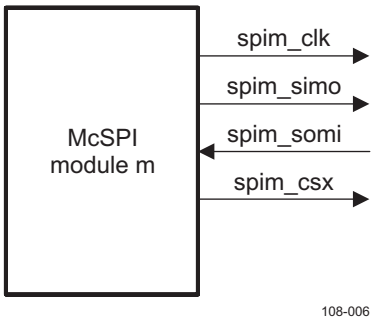


Table 16-1 describes the McSPI I/O in master mode.

Table 16-1. McSPI I/O Description (Master Mode)

Signal Name	I/O	Description	Reset [1]
spim_clk	O	SPIm module serial clock[2]	Unknown
spim_simo	O	SPIm module serial data master out (slave input, master output)	Unknown
spim_somi	I	SPIm module serial data master input (slave output, master input)	–
spim_csx	O	SPIm module chip-select x output	Low

(1) After reset, the SPI modules are in slave mode by default. This paragraph implies that the McSPI module is configured in slave mode. (See the MCSPI_MODULCTRL[2] MS bit in the module control register [MCSPI_MODULCTRL]).

(2) This output signal is also used as retiming input (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).

16.3.2 Basic McSPI Pins for Slave Mode

Figure 16-8 shows all of the McSPI interface signals in slave mode.

Figure 16-8. McSPI Interface Signals in Slave Mode

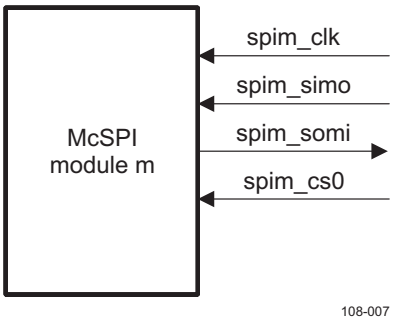


Table 16-2 describes the McSPI I/O in slave mode.

Table 16-2. McSPI I/O Description (Slave Mode)

Signal Name	I/O	Description	Reset [1]
spim_clk	I	SPIm module serial clock	Unknown
spim_simo	I	SPIm module serial data master out (slave input, master output)	Unknown
spim_somi	O	SPIm module serial data master input (slave output, master input)	–
spim_cs0	I/O	SPIm module chip-select 0 input	HiZ
spim_cs1/2/3	O	SPIm module chip-select 1/2/3 output	Low

(1) After reset, the SPI modules are in slave mode by default. This paragraph implies that the McSPI module is configured in slave mode. (See the MCSPI_MODULCTRL[2] MS bit in Module Control Register (MCSPI_MODULCTRL).)

16.3.3 Multichannel SPI Protocol and Data Format

The synchronous SPI protocol allows a master device to initiate serial data transfers to a slave device. A slave select line (spim_csx) allows selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities.

McSPI offers the flexibility to modify the following parameters to adapt to the device features:

- Word length

  McSPI supports any SPI word ranging from 4 bits to 32 bits long (SPIm.MCSPI_CHxCONF[11:7] WL field).

  SPI word length can be changed between transmissions to allow the master device to communicate with peripheral slaves that have different requirements.

- SPI enable (spim_csx, for channel x of instance m)

  The polarity of the SPI enable signals is programmable (SPIm.MCSPI_CHxCONF[6] EPOL bit). spim_csx signals can be active high or low.

  The assertion of the spim_csx signals is programmable and can be manually or automatically asserted. The manual assertion mode is available in single master mode only. spim_csx can be kept active between words with the SPIm.MCSPI_CHxCONF[20] FORCE bit.

  Two consecutive words for two different slave devices can go along with active spim_csx signals with different polarity (see the example in Section 16.6, *McSPI Basic Programming Model*).

- Programmable start-bit

  In start-bit mode a start-bit is added before the SPI word length to indicate how the next SPI word must be handled. The start-bit is enabled by setting SPIm.MCSPI_CHxCONF[23] SBE bit to 1. The SPIm.MCSPI_CHxCONF[24] SBPOL bit defines the polarity of the start-bit.

- Programmable SPI clock
  - Bit rate

    In master mode, the baud rate of the SPI serial clock is programmable using the 48-MHz reference clock (from the power, reset, and clock management [PRCM] module). Table 16-3 gives the spim_clk bit rates obtained for data transfer when programming the clock divider (SPIm.MCSPI_CHxCONF[5:2] CLKD bit field).

Table 16-3. SPI Master Clock Rates

Divider	Clock Rate
1	48 MHz
2	24 MHz
4	12 MHz
8	6 MHz
16	3 MHz
32	1.5 MHz
64	750 kHz

Table 16-3. SPI Master Clock Rates (continued)

Divider	Clock Rate
128	375 kHz
256	~187 kHz
512	~93.7 kHz
1024	~46.8 kHz
2048	~23.4 kHz
4096	~11.7 kHz
8192	~5.8 kHz
16384	~2.9 kHz
32768	~1.5 kHz

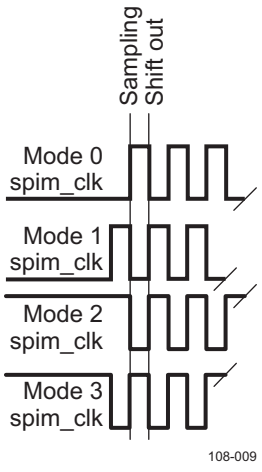
– Polarity and phase

The polarity (the SPIm.MCSPI_CHxCONF[1] POL bit) and the phase (the SPIm.MCSPI_CHxCONF[0] PHA bit) of the SPI serial clock (spim_clk) are configurable to offer four combinations. Software selects the right combination, depending on the device. See Table 16-4 and Figure 16-9.

Table 16-4. Phase and Polarity Combinations

Polarity (POL)	Phase (PHA)	SPI Mode	Comments
0	0	Mode 0	spim_clk is active high and sampling occurs on the rising edge.
0	1	Mode 1	spim_clk is active high and sampling occurs on the falling edge.
1	0	Mode 2	spim_clk is active low and sampling occurs on the falling edge.
1	1	Mode 3	spim_clk is active low and sampling occurs on the rising edge.

Figure 16-9. Phase and Polarity Combinations



16.3.3.1 Transfer Format

In both master and slave modes, McSPI drives the data lines when spim_csx is asserted.

Each word is transmitted starting with the most-significant bit (MSB).

This section explains the two cases of data transmission determined by the clock phase (PHA) and the type of data transmission using a start-bit (SBE) called the start-bit mode:

- Transmission in mode 0 and mode 2 (PHA = 0)

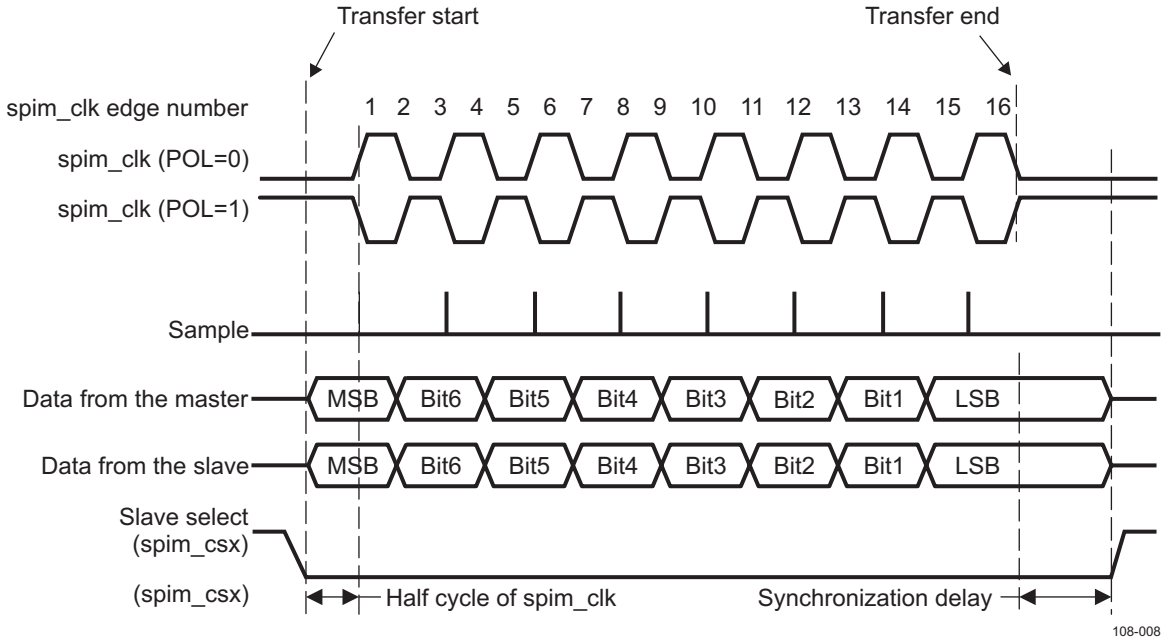
When PHA = 0, the first bit of the SPI word to transmit (on the master or the slave data output pin) is valid one half cycle of spim_clk after the spim_csx assertion.

Therefore, the first edge of the spim_clk line is used by the master to sample the first data bit sent by the slave. On the same edge, the first data bit sent by the master is sampled by the slave.

On the next spim_clk edge, the received data bit is shifted into the receive shift register and a new data bit is transmitted on the serial data line.

This process continues for a number of pulses on the spim_clk line defined by the SPI word length programmed in the master device, with data being latched on odd-numbered edges and shifted on even-numbered edges. See Figure 16-10.

Figure 16-10. Full-Duplex Transfer Format With PHA = 0



- Transmission in mode 1 and mode 3 (PHA = 1)

  When PHA = 1, the first bit of the SPI word to transmit (on the master or the slave data output pin) is valid on the following spim_clk edge (one half cycle later). This is the sampling edge for the master and slave. A synchronization delay is added between the spim_csx activation and the first spim_clk edge.

  The received data bit is shifted into the shift register on the third spim_clk edge.

  This process continues for a number of pulses on the spim_clk line defined by the SPI word length programmed in the master device, with data being latched on even-numbered edges and shifted on odd-numbered edges.

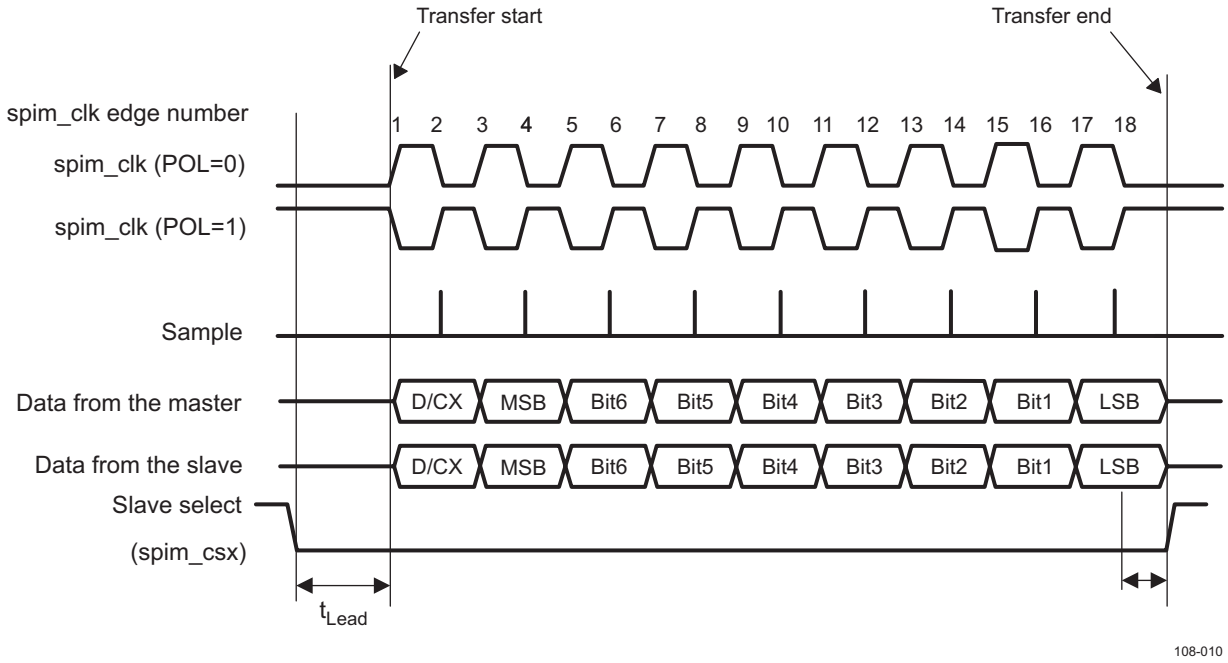
  NOTE: The minimum synchronization delay is one cycle of spim_clk, if the spim_clk frequency equals the SPIm_FCLK (McSPIm functional clock) frequency in master mode. The minimum synchronization delay is one-half cycle of spim_clk, if the spim_clk frequency is lower than the SPIm_FCLK frequency in both master and slave modes.

- Transmission with a start-bit (SBE = 1)

  When the SPIm.MCSPI_CHxCONF[23] SBE bit = 1, a start-bit is added before the MSB to indicate whether the next SPI word must be handled as a command or as data.

  Figure 16-11 shows an example of a data transfer with an extra start-bit.

Figure 16-11. Extended SPI Transfer With a Start-Bit (SBE = 1)

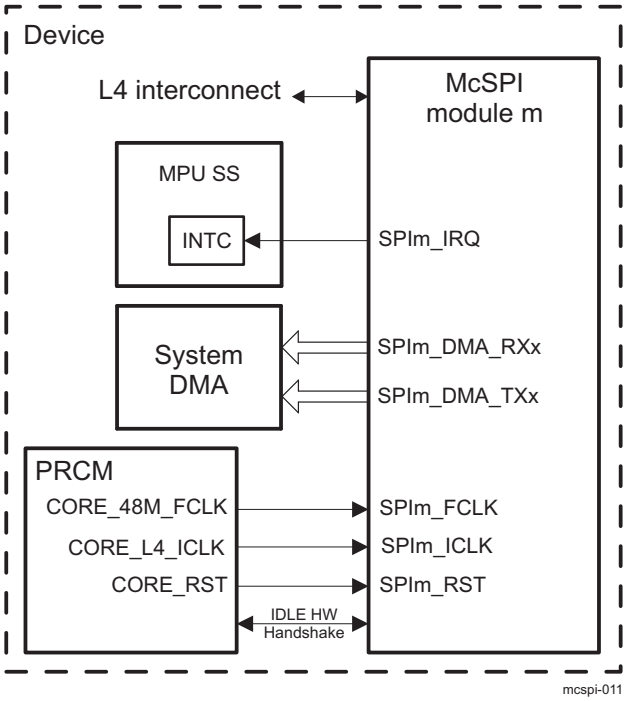


16.4 McSPI Integration

16.4.1 McSPI Description

Figure 16-12 highlights the McSPI module integration in the device.

Figure 16-12. McSPI Integration



16.4.2 Clocking, Reset, and Power-Management Scheme

16.4.2.1 Clocking

Each McSPI module is clocked with an independent functional clock of 48 MHz from the PRCM module (SPIm_FCLK with m = [1,4]) and with an interface clock, CORE_L4_ICLK (L4 interconnect):

- SPIm_FCLK is the McSPI module *m* functional clock and is used to clock the McSPI internal logic. The source of SPIm_FCLK is the PRCM CORE_48M_FCLK output clock.
- SPIm_ICLK is the McSPI module *m* interface clock and is used to synchronize the McSPI L4 port to the L4 interconnect. All accesses from the interconnect are synchronous with SPIm_ICLK. The source of SPIm_ICLK is the PRCM CORE_L4_ICLK output clock.

From a global system power management perspective, when one or both of the McSPI clocks is not required, the McSPI module can be deactivated at the PRCM level in the corresponding registers.

Table 16-5 describes the McSPI module PRCM clock control bits.

Table 16-5. McSPI Clocks

McSPI Clock	Associated PRCM Clock Output	Enable Bit	Autoidle Bit
SPIm_FCLK			
	CORE_48M_FCLK	PRCM.CM_FCLKEN1_CORE.EN_MCSPIm (with m=[1,4])	N/A
SPIm_ICLK			
	CORE_L4_ICLK	PRCM.CM_ICLKEN1_CORE.EN_MCSPIm (with m=[1,4])	PRCM.CM_AUTOIDLE1_CORE.AUTO_MCSPIm (with m=[1,4])

NOTE:

- The PRCM CORE_48M_FCLK output is gated at the PRCM level, assuming that all modules sharing it are disabled in the corresponding register. Disabling the McSPI module is a necessary but insufficient condition.
- The PRCM CORE_L4_ICLK output is gated at the PRCM level, assuming that all modules sharing it are disabled in the corresponding register. Disabling the McSPI module is a necessary but insufficient condition.
- The PRCM.CM_AUTOIDLE1_CORE bit is used to link/unlink the McSPI module from CORE_L4_ICLK-related clock domain transitions.

For more information about source clocks gating and domain transitions, see , *Power, Reset, and Clock Management*. For more information on power saving management, see Section 16.5.7.

16.4.2.2 Power Domain

The McSPI modules belong to the CORE power domain (see Table 16-6).

Table 16-6. Power Domain

Peripherals	Power Domain
McSPI modules	CORE power domain

16.4.2.3 Hardware Reset

As part of the CORE power domain, CORE_RST is issued by the PRCM module (for more information about the CORE power domain implementation and CORE_RST signal, see , *Power, Reset, and Clock Management*). The module is reset by hardware when an active-low reset signal is asserted. The hardware reset signal has a global reset effect on the module. All configuration registers and all state-machines are reset in all clock domains (see Table 16-7).

Table 16-7. McSPI Hardware Reset

Peripherals	Name	Comments
McSPI module	CORE_RST	Same as global reset

16.4.2.4 Software Reset

The SPIm.MCSPI_SYSCONFIG[1] SOFTRESET bit controls the software reset of the SPI interface. Writing 1 to this bit enables active software reset functionality, which is equivalent to a hardware reset. The bit is automatically reset by hardware.

NOTE: The SPIm.MCSPI_SYSCONFIG register is not sensitive to software reset.

16.4.3 Hardware Requests

16.4.3.1 DMA Requests

Each channel includes two DMA requests, one for reception and one for transmission (see Table 16-8).

Table 16-8. DMA Requests

Attributes	DMA Request	Name	Mapping	Comments
SPI1				
	8	SPI1_DMA_TX0	S_DMA_34	Destination is system DMA (sDMA).
		SPI1_DMA_RX0	S_DMA_35	
		SPI1_DMA_TX1	S_DMA_36	
		SPI1_DMA_RX1	S_DMA_37	
		SPI1_DMA_TX2	S_DMA_38	
		SPI1_DMA_RX2	S_DMA_39	
		SPI1_DMA_TX3	S_DMA_40	
		SPI1_DMA_RX3	S_DMA_41	
SPI2				
	4	SPI2_DMA_TX0	S_DMA_42	Destination is sDMA.
		SPI2_DMA_RX0	S_DMA_43	
		SPI2_DMA_TX1	S_DMA_44	
		SPI2_DMA_RX1	S_DMA_45	
SPI3				
	4	SPI3_DMA_TX0	S_DMA_14	Destination is sDMA.
		SPI3_DMA_RX0	S_DMA_15	
		SPI3_DMA_TX1	S_DMA_22	
		SPI3_DMA_RX1	S_DMA_23	
SPI4				
	2	SPI4_DMA_TX0	S_DMA_69	Destination is sDMA.
		SPI4_DMA_RX0	S_DMA_70	

16.4.3.2 Interrupt Requests

Each McSPI module handles one interrupt line (see Table 16-9).

Table 16-9. Interrupt Requests

Attributes	Interrupt Request	Name	Mapping	Comments
SPI1				
	1	SPI1_IRQ	M_IRQ_65	Destination is the microprocessor unit (MPU) interrupt controller.
SPI2				
	1	SPI2_IRQ	M_IRQ_66	Destination is the MPU interrupt controller.
SPI3				
	1	SPI3_IRQ	M_IRQ_91	Destination the MPU interrupt controller.
SPI4				
	1	SPI4_IRQ	M_IRQ_48	Destination is the MPU interrupt controller.

16.4.3.3 Wake-Up Requests

Table 16-10 lists the wake-up requests.

Table 16-10. Wake-Up Requests

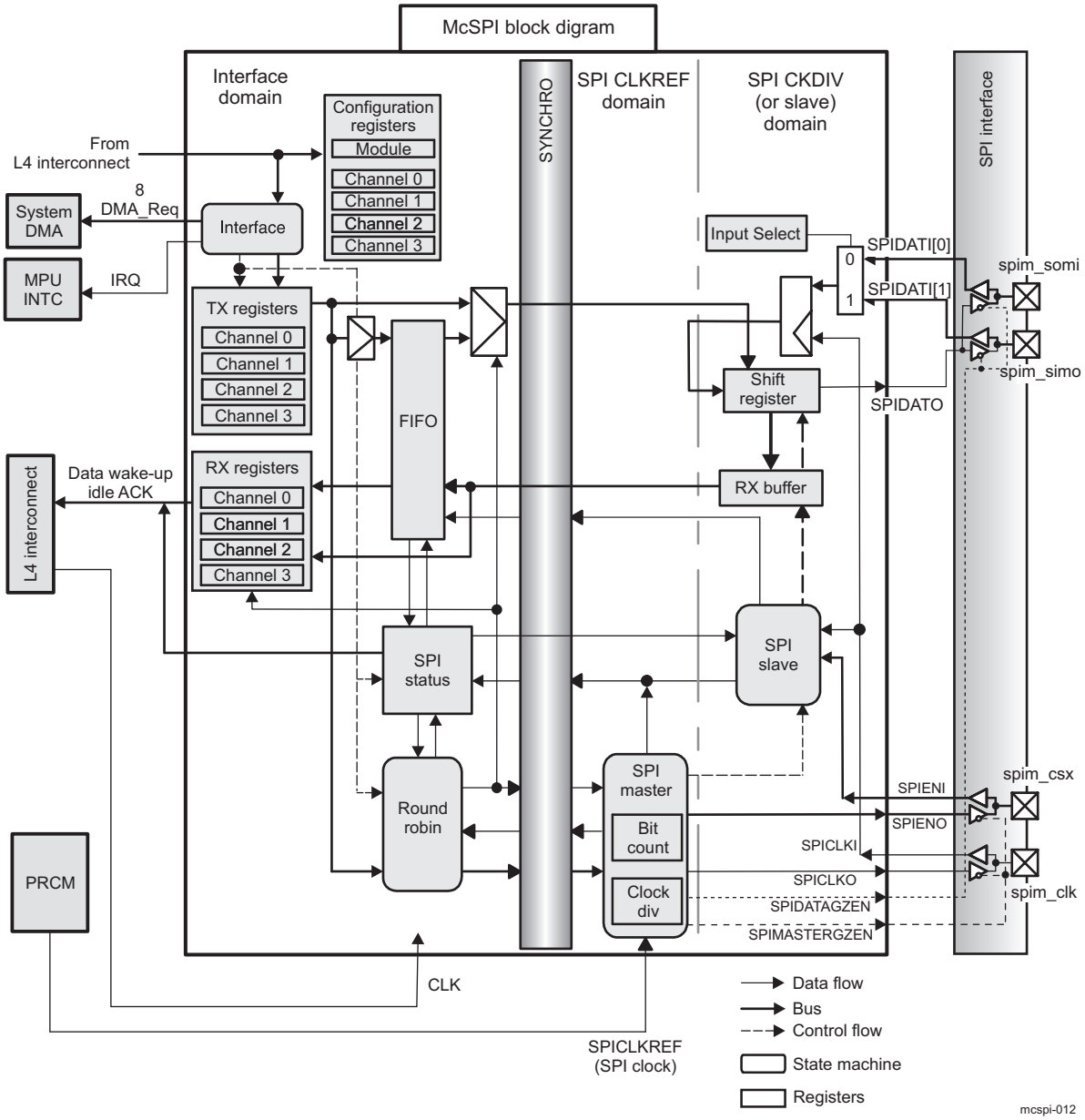
Attributes	Wake Request	Name	Mapping	Comments
SPI1				
	1	spi1_cs0	SPI1_ SWAKEUP	Destination is the PRCM module.
SPI2				
	1	spi2_cs0	SPI2_ SWAKEUP	Destination is the PRCM module.
SPI3				
	1	spi3_cs0	SPI3_ SWAKEUP	Destination is the PRCM module.
SPI4				
	1	spi4_cs0	SPI4_ SWAKEUP	Destination is the PRCM module.

16.5 McSPI Functional Description

16.5.1 McSPI Block Diagram

Figure 16-13 shows the McSPI module.

Figure 16-13. McSPI Block Diagram



16.5.2 Master Mode

16.5.2.1 Master Mode Features

The McSPI master mode supports multichannel communication with up to four independent SPI communication channel contexts. The McSPI initiates a data transfer on the data lines (spim_simo and spim_somi) and generates clock (spim_clk) and control signals (spim_csx).

Connected to multiple external devices, the McSPI exchanges data with one SPI device at a time through two main modes (available in slave mode):

- Two-data-pins interface mode (transmit-and-receive mode for full-duplex transmission)
- Single-data-pin interface mode (recommended for half-duplex transmission)

Two DMA request events (read and write) allow synchronized accesses of the DMA controller with the activity of McSPI.

Three interrupt events can be used for data transmission and reception in master mode (for more information on interrupts, see Section 16.5.5.1, *Interrupt Events in Master Mode*).

16.5.2.2 Master Transmit-and-Receive Mode (Full Duplex)

In full-duplex transmission, data is transmitted (shifted out serially on spim_simo) and received (shifted in serially on spim_somi) simultaneously on separate data lines.

The master transmit-and-receive mode is programmable per channel (the SPIm.MCSPI_CHxCONF[13:12] TRM field).

Channel access to the shift registers for transmission/reception is based on the MCSPI_TXx transmitter register state, the MCSPI_RXx receiver register state, and round robin arbitration.

Channels that meet the following rules are included in the round robin list of active channels scheduled for transmission and/or reception. The arbiter skips channels that do not meet the rules and searches in the rotation for the next enabled channel.

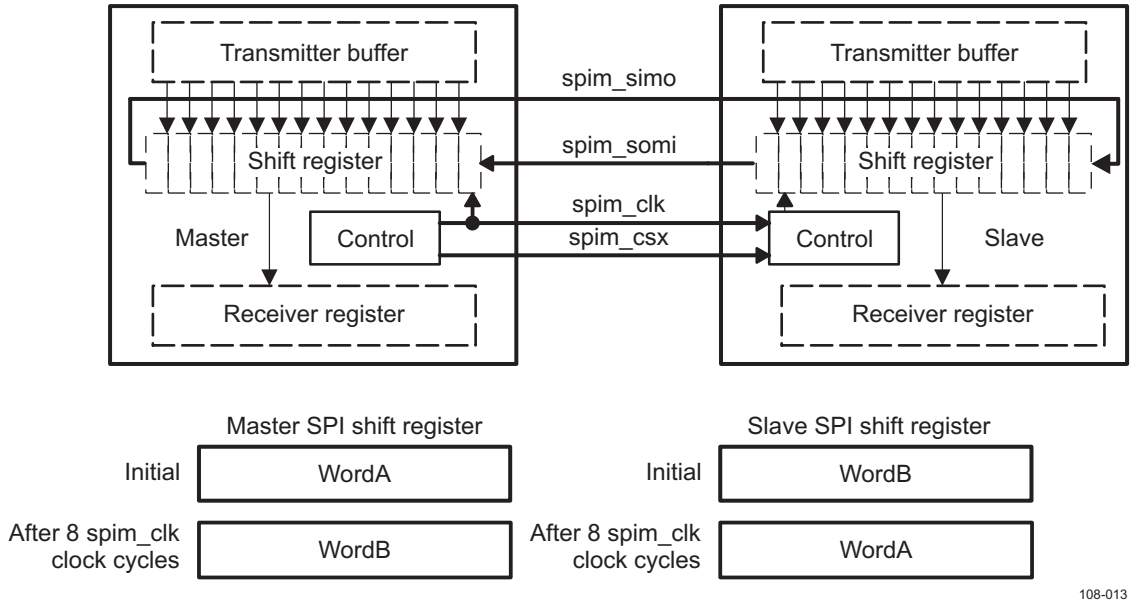
- Rule 1: Only enabled channels (the SPIm.MCSPI_CHxCTRL[0] EN bit) can be scheduled for transmission and/or reception.
- Rule 2: If its MCSPI_TXx transmitter register is not empty (the SPIm.MCSPI_CHxSTAT[1] TXS bit), an enabled channel can be scheduled when the shift register is assigned. If the MCSPI_TXx register is empty when the shift register is assigned, the TXx_UNDERFLOW event is activated, and the next enabled channel with new data to transmit is scheduled (see also the transmit-only mode).
- Rule 3: An enabled channel can be scheduled if its receive register is not full (the SPIm.MCSPI_CHxSTAT[0] RXS bit) when the shift register is assigned (see also the receive-only mode). Therefore, the MCSPI_RXx register cannot be overwritten. Note that the SPI1.MCSPI_IRQSTATUS[3] RX0_OVERFLOW bit is never set to this mode.

When SPI word transfer completes (the SPIm.MCSPI_CHxSTAT[2] EOT bit is set), the updated MCSPI_TXx register of the next scheduled channel is loaded into the shift register. The serialization (transmit-and-receive) starts depending on the channel communication configuration. When serialization completes, the received data transfers to the channel receive register.

The serial clock (spim_clk) synchronizes shifting and sampling of the information on the two serial data lines (spim_simo and spim_somi). Each time a bit transfers out from the master, 1 bit transfers in from the slave.

Figure 16-14 shows an example of a full-duplex system with a master device (McSPI module *m*) on the left and a slave device on the right. After eight cycles of the serial clock spim_clk, WordA transfers from the master to the slave. At the same time, WordB transfers from the slave to the master.

Figure 16-14. SPI Full-Duplex Transmission (Example)



16.5.2.3 Master Transmit-Only Mode (Half Duplex)

The master transmit-only mode prevents the MPU from reading the MCSPI_RXx register (minimizing data movement) when only transmission is meaningful.

The master transmit-only mode is programmable per channel (the SPIm.MCSPI_CHxCONF[13:12] TRM field). Transmission starts only after data is loaded into the MCSPI_TXx register.

Rule 1 and Rule 2, defined in Section 16.5.2.2, are applicable in this mode.

Rule 3, defined in Section 16.5.2.2, is not applicable.

In master transmit-only mode, the MCSPI_RXx register state FULL does not prevent transmission and the MCSPI_RXx register is always overwritten with the new SPI word. This event is not significant when only transmission is meaningful. Thus, the RX0_OVERFLOW bit in the SPIm.MCSPI_IRQSTATUS register is never set in this mode.

The hardware automatically disables the RX_FULL interrupt and the DMA read requests.

The transfer status is given by the SPIm.MCSPI_CHxSTAT[2] EOT bit.

16.5.2.4 Master Receive-Only Mode (Half Duplex)

The master receive mode prevents the MPU from refilling the MCSPI_TXx register (minimizing data movement) when only reception is meaningful.

The master receive mode is programmable per channel (the SPIm.MCSPI_CHxCONF[13:12] TRM field).

The master receive-only mode enables channel scheduling only on the empty state of the MCSPI_RXx register.

Rule 1 and Rule 3, defined in Section 16.5.2.2, are applicable in this mode.

Rule 2, defined in Section 16.5.2.2, is not applicable.

In the master receive-only mode, software must write dummy data to the MCSPI_TXx register and only after the TX buffer is empty (check the MCSPI_CH0STAT[2] bit in the software). Only one dummy write is enough to receive any number of words from the slave. Software should ensure that the MCSPI_TXx register is always full (the TXx_EMPTY bits of SPIm.MCSPI_IRQSTATUS) when receiving. The content of the MCSPI_TXx register is always loaded into the shift register when the shift register is assigned. After writing the dummy data to the MCSPI_TXx register, the TXx_EMPTY and TXx_UNDERFLOW bits in the SPIm.MCSPI_IRQSTATUS register are never set in receive-only mode.

The SPIm.MCSPI_CHxSTAT[2] EOT bit gives the status of serialization. The RXx_FULL bits of the SPIm.MCSPI_IRQSTATUS register are set when received data is loaded from the shift register to the corresponding MCSPI_RXx register. The SPIm.MCSPI_IRQSTATUS[3] RX0_OVERFLOW bit is never set in this mode.

16.5.2.5 Single-Channel Master Mode

When the McSPI is configured as a master device with a single enabled channel, the assertion of the spim_csx signal can be controlled in two different ways:

- If the MCSPI_MODULCTRL[0] SINGLE bit is set to 0, spim_csx assertion/deassertion after each SPI word is automatically controlled by the McSPI module (see subsections of Section 16.5.2.1, *Master Mode Features*.
- If the MCSPI_MODULCTRL[0] SINGLE bit and the MCSPI_CHxCONF[20] FORCE bit are set to 1: spim_csx assertion/deassertion is controlled by software (see Section 16.5.2.5.1, *Programming Tips When Switching to Another Channel*.

16.5.2.5.1 Programming Tips When Switching to Another Channel

When a single channel is enabled and data transfer is ongoing:

- Wait for completion of the SPI word transfer (wait until the SPIm.MCSPI_CHxSTAT[2] EOT bit is set to 1) before disabling the current channel and enabling a different channel.
- Disable the current channel first, and then enable the other channel.

16.5.2.5.2 Force spim_csx Mode

Continuous transfers are manually allowed by keeping the spim_csx signal active for successive SPI words transfer. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the spim_csx line. This mode is supported by all channels and any master sequence can be used (transmit-receive, transmit-only, receive-only).

Keeping the spim_csx active mode is supported when:

- A single channel is used (with the SPIm.MCSPI_MODULCTRL[0] SINGLE bit set to 1).
- Transfer parameters are loaded in the configuration register of the appropriate channel (SPIm.MCSPI_CHxCONF).

  The state of the spim_csx signal is programmable.

  – Writing 1 to the SPIm.MCSPI_CHxCONF[20] FORCE bit drives the spim_csx line high when the SPIm.MCSPI_CHxCONF[6] EPOL bit is set to 0. spim_csx is driven low when the SPIm.MCSPI_CHxCONF[6] EPOL bit is set to 1.

  – Writing 0 to the SPIm.MCSPI_CHxCONF[20] FORCE bit drives the spim_csx line low when the SPIm.MCSPI_CHxCONF[6] EPOL bit is set to 0. spim_csx is driven high when the SPIm.MCSPI_CHxCONF[6] EPOL bit is set to 1.
- A single channel is enabled (the SPIm.MCSPI_CHxCTRL[0] EN bit is set to 1). The first enabled channel activates the spim_csx line.

When the channel is enabled, the spim_csx signal activates with the programmed polarity. As in the multichannel master mode, the transfer start depends on the status of the MCSPI_TXx register (the SPIm.MCSPI_CHxSTAT[1] TXS bit), the status of the MCSPI_RXx register (the SPIm.MCSPI_CHxSTAT[1] RXS bit), and the defined mode (the SPIm.MCSPI_CHxCONF[13:12] TRM field) of the channel enabled.

The SPIm.MCSPI_CHxSTAT[2] EOT bit gives the transfer status of each SPI word. The RXx_FULL bit in the SPIm.MCSPI_IRQSTATUS register is set when received data is loaded from the shift register to the MCSPI_RXx register.

A change in the configuration parameters is propagated directly on the SPI interface. If the spim_csx signal is activated, ensure that the configuration is changed only between SPI words to avoid corrupting the current transfer.

NOTE: To avoid data corruption, spim_csx polarity and spim_clk phase and spim_clk polarity must not be modified when the spim_csx signal is activated.

A delay between SPI words that requires the connected SPI slave device to switch from one configuration (transmit-only for instance) to another (receive-only for instance) must be handled by software.

At the end of the last SPI word, the channel must be deactivated (the SPIm.MCSPI_CHxCTRL[0] EN bit set to 0) and spim_csx can be forced to its inactive state using the SPIm.MCSPI_CHxCONF[20] FORCE bit.

Figure 16-15 and Figure 16-16 show successive transfers with spim_csx maintained active low with a different configuration for each SPI word in single-data-pin and dual-data-pin interface modes, respectively.

Figure 16-15. Continuous Transfers With spim_csx Maintained Active (Single-Data-Pin Interface Mode)

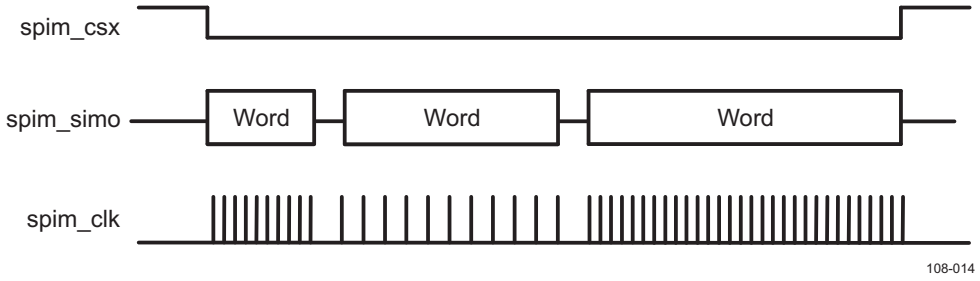
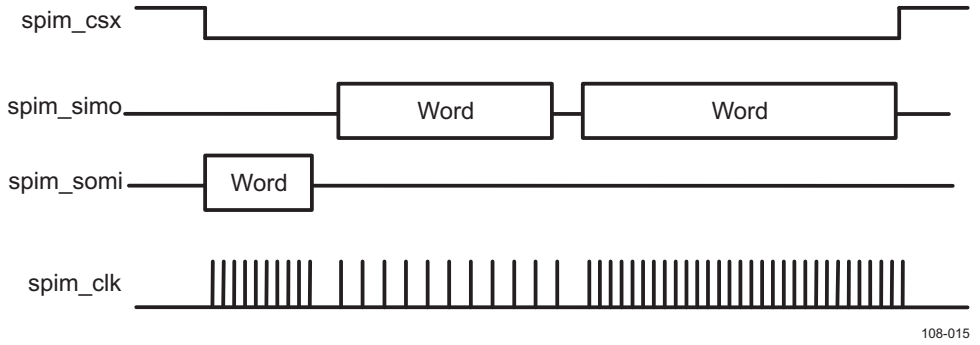


Figure 16-16. Continuous Transfers With spim_csx Maintained Active (Dual-Data-Pin Interface Mode)



NOTE: The turbo mode described in Section 16.5.2.5.3, *Turbo Mode*, maintains spim_csx in active mode when the following conditions are met:

- A single channel is explicitly used (the SPIm.MCSPI_MODULCTRL[0] SINGLE bit is set to 1).
- Turbo mode is enabled in the configuration of the channel. (The SPIm.MCSPI_CHxCONF[19] TURBO bit is set to 1.)

16.5.2.5.3 Turbo Mode

The turbo mode improves the throughput of the SPI interface when a single channel is enabled by allowing transfers until the shift register and the MCSPI_RXx register are full. The turbo mode is useful (time savings) when a transfer exceeds two words. This mode is programmable per channel (via the SPI1.MCSPI_CHxCONF[9] TURBO bit).

When several channels are enabled, the TURBO bit has no effect and the channel access to the shift registers remains as previously described.

In turbo mode, Rule 1 and Rule 2 applies, but Rule 3 does not (see Section 16.5.2.2). An enabled channel can be scheduled if its receive register is full (the SPIm.MCSPI_CHxSTAT[0]) RXS bit) at the time of the shift-register assignment until the shift register is full.

The MCSPI_RXx register cannot be overwritten in turbo mode. Consequently, the SPIm.MCSPI_IRQSTATUS[3] RX0_OVERFLOW bit is never set in this mode.

16.5.2.6 Start Bit Mode

In start bit mode, an extended bit is added before the SPI word in order to indicate whether the next SPI word must be handled as a command or as data. This feature is only available in master mode. The start bit mode cannot be used at the same time as turbo mode and/or force spim_csx mode. In this case, only one channel can be used; round-robin arbitration is not possible.

This mode is programmable per channel by setting the SPIm.MCSPI_CHxCONF[23] SBE bit to 1. The polarity of the extended bit is programmable per channel. When the SPIm.MCSPI_CHxCONF[24] SBPOL bit is set to 0, the SPI word must be handled as a command. When the SPIm.MCSPI_CHxCONF[24] SBPOL bit is set to 1, the SPI word must be handled as data. Moreover, start-bit polarity can be changed dynamically during start bit transfer without disabling the channel for reconfiguration; in this case, users must configure the SPIm.MCSPI_CHxCONF[24] SBPOL bit before writing the SPI word to be transmitted to the TX register.

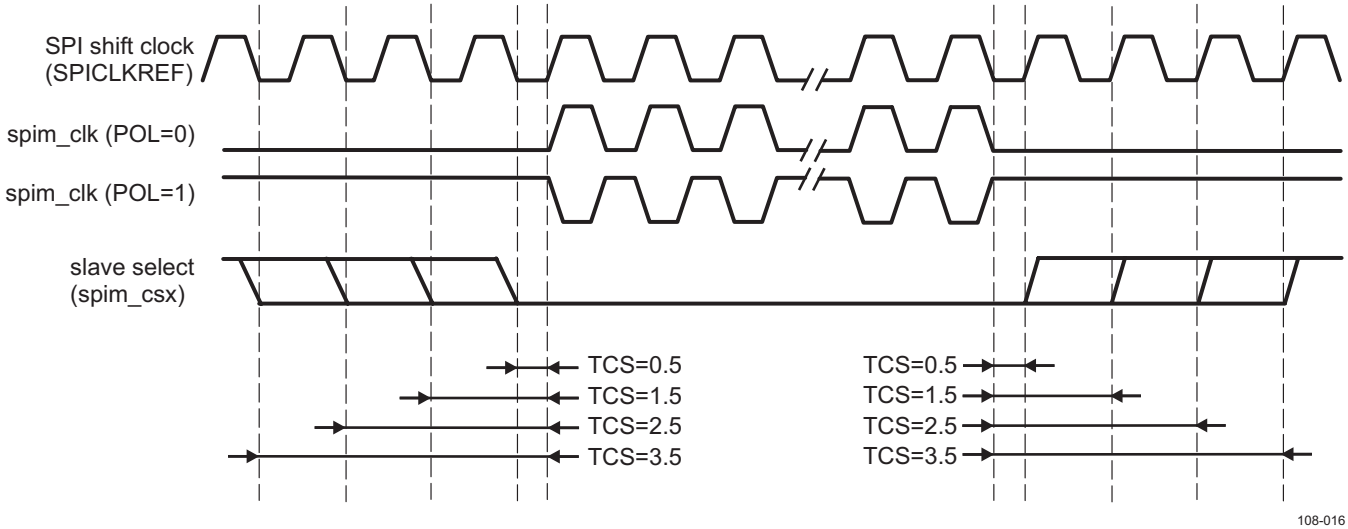
16.5.2.7 Chip-Select Timing Control

The chip select timing control is only available in master mode with automatic chip select generation (MCSPI_MODULCTRL[0] SINGLE bit field set to 0), to add a programmable delay between chip select assertion and first clock edge, or chip select removal and last clock edge.

This mode is programmable per channel (the TCS bit of the SPIm.MCSPI_CHxCONF register).

Figure 16-17 shows the chip-select SPIEN timing control.

Figure 16-17. Chip-Select SPIEN Timing Controls



NOTE: Because of the design implementation for transfers using a clock divider ratio set to 1 (clock bypassed), a half cycle must be added to the value between chip-select assertion and the first clock edge with PHA = 1 or between chip-select removal and the last clock edge with PHA = 0.

16.5.2.8 Programmable SPI Clock (spim_clk)

In master mode, the baud rate of the SPI serial clock is programmable.

An internal reference clock, SPIm_FCLK, is used as input of a programmable divider (the SPIm.MCSPI_CHxCONF[5:2] CLKD field) to generate the bit rate of the serial output clock spim_clk. Table 16-11 summarizes the supported divisor values.

Table 16-11. SPI Master Clock Rates

Divider	Clock Rate
1	48 MHz
2	24 MHz
4	12 MHz
8	6 MHz
16	3 MHz
32	1.5 MHz
64	750 kHz
128	375 kHz
256	~187 kHz
512	~93.7 kHz
1024	~46.8 kHz
2048	~23.4 kHz
4096	~11.7 kHz
8192	~5.8 kHz
16384	~2.9 kHz
32768	~1.5 kHz

16.5.2.8.1 Clock Ratio Granularity

By default the clock division ratio is defined by the SPIm.MCSPI_CHxCONF[5:2] CLKD bit field with power of two granularity leading to a clock division in range 1 to 4096, in this case the duty cycle is always 50%. With the SPIm.MCSPI_CHxCONF[29] CLKG bit, clock division granularity can be changed to one clock cycle, in that case the SPIm.MCSPI_CHxCTRL[15:8] EXTCLK bit field is concatenated with SPIm.MCSPI_CHxCONF[5:2] CLKD bit field to give a 12-bit width division ratio in range 1 to 4096.

When granularity is one clock cycle (CLKG set to 1), for odd value of clock ratio the clock high level lasts one clock cycle more than low level depending on SPIm.MCSPI_CHxCONF[1] POL bit and SPIm.MCSPI_CHxCONF[0] PHA bit refer to Table 16-12.

Table 16-12. CLKSPIO High/Low Time Computation

Clock ratio Fratio	CLKSPIO High Time	CLKSPIO Low Time
1	Thigh_ref	Tlow_ref
Even ≥ 2	T_ref * (Fratio/2)	T_ref * (Fratio/2)
Odd ≥ 3 (POL=PHA)	T_ref * (Fratio-1) /2	T_ref * (Fratio+1) /2
Odd ≥ 3 (POL≠PHA)	T_ref * (Fratio+1) /2	T_ref * (Fratio-1) /2

NOTE: Fratio = spi1_clk frequency (Fout) division ratio.
Thigh = spi1_clk High Time period.
Tlow = spi1_clk Low Time period.
T_ref = SPI1_FCLK period.
Thigh_ref = SPI1_FCLK high Time period.
Tlow_ref = SPI1_FCLK low Time period.

If CLKG = 1: Fratio = EXTCLK concatenated with CLKD + 1 as shown below:

Fratio - 1											
11	10	9	8	7	6	5	4	3	2	1	0
SPI1.MCSPI_CHxCTRL[15:8] EXTCLK bit field								SPI1.MCSPI_CHxCONF[5:2] CLKD bit field			
15	14	13	12	11	10	9	8	5	4	3	2

For odd ratio values, the duty cycle is calculated as below:

Duty_cycle = (1-1/Fratio)/2

Table 16-13 shows clock granularity examples with a clock source frequency of 48 MHz.

Table 16-13. Clock Granularity Examples

EXTCLK	CLKD	CLKG	Fratio	PHA	POL	Thigh (ns)	Tlow (ns)	Tperiod (ns)	Duty Cycle	Fout (MHz)
X	0	0	1	X	X	10.4	10.4	20.8	50-50	48
X	1	0	2	X	X	20.8	20.8	41.6	50-50	24
X	2	0	4	X	X	41.6	41.6	83.2	50-50	12
X	3	0	8	X	X	83.2	83.2	166.4	50-50	6
0	0	1	1	X	X	10.4	10.4	20.8	50-50	48
0	1	1	2	X	X	20.8	20.8	41.6	50-50	24
0	2	1	3	1	0	41.6	20.8	62.4	66-33	16
0	2	1	3	1	1	20.8	41.6	62.4	33-66	16
0	3	1	4	X	X	41.6	41.6	83.2	50-50	12
5	0	1	81	1	0	852.8	832	1684.8	50.6-49.4	0.592
5	7	1	88	X	X	915.2	915.2	1830.4	50-50	0.545

16.5.3 Slave Mode

To select the McSPI slave mode, set the SPIm.MCSPI_MODULCTRL[2] MS bit.

A McSPI slave device can be connected to up to four external SPI master devices but handles transactions with one SPI master device at a time.

In slave mode, the McSPI initiates data transfer on the data lines (spim_simo and spim_somi) when it is selected by an active control signal (spim_csx) and receives an SPI clock (spim_clk) from the external SPI master device. Only channel 0 can be configured as a slave. In slave mode, the McSPI uses the edge of spim_csx to detect word length. For this reason, spim_csx must become inactive between each word.

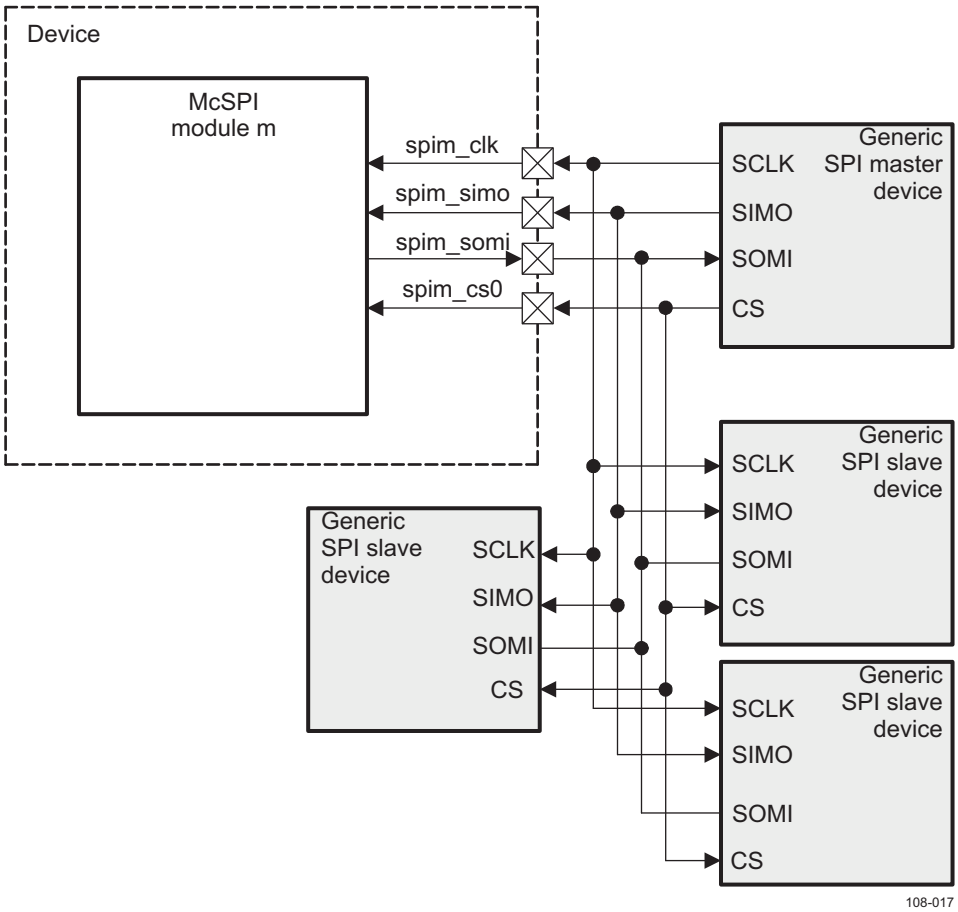
The McSPI does not support spim_csx active between SPI words. It uses the edge to detect word length.

16.5.3.1 Dedicated Resources

Only channel 0 can be enabled in slave mode. In this section registers name such as SPI1.MCSPI_CHxCTRL stand for SPI1.MCSPI_CH0CTRL where x=0 (channel 0 control register).

Figure 16-18 shows an example of four slaves wired on a single master device.

Figure 16-18. Example of McSPI Slave With One Master and Multiple Slave Devices on Channel 0



The channel 0 in slave mode has the following resources:

- Its own channel enable, programmable with the SPIm.MCSPI_CHxCTRL[0] EN bit (with x=0). This channel must be enabled before transmission and reception.
- For this mode, the slave-select signal can be detected only on spin_cs0.
- Its own transmitter register, SPIm.MCSPI_TXx (with x=0), on top of the common transmit shift register. If the MCSPI_TXx register is empty, the SPIm.MCSPI_CHxSTAT[1] TXS bit (with x=0) is set. If McSPI is selected by an external master (the active signal on the spim_csx port assigned to channel 0), the MCSPI_TXx register content of channel 0 is always loaded into the shift register, whether its content is updated or not. The MCSPI_TXx register must be loaded before McSPI is selected by a master.
- Its own receiver register, SPIm.MCSPI_RXx (with x=0), on top of the common receive shift register. If the MCSPI_RXx register is full, the SPIm.MCSPI_CHxSTAT[0] RXS bit (with x=0) is set.

NOTE: The MCSPI_TXx register and MCSPI_RXx registers of the other channels are not used. Reading from or writing to a channel register other than channel 0 has no effect.

- Its own communication configuration with the following parameters through the SPIm. MCSPI_CHxCONF register (with x=0):
  - Transmit and receive modes, programmable with the TRM field
  - Interface mode (two data pins or single data pin) and data pins assignment, both programmable with the IS and DPE bits. (The SPIm modules are in slave mode after reset and must be properly configured for the modules to act in master mode.)
  - SPI word length, programmable with the WL bit
  - spim_csx polarity, programmable with the EPOL bit

– spim_clk polarity, programmable with the POL bit
– spim_clk phase, programmable with the PHA bit

The spim_clk frequency of a transfer is controlled by the external SPI master connected to the McSPI slave device. The SPIm.MCSPI_CHxCONF[5:2] CLKD field (with x=0) is not used in slave mode.

NOTE: The configuration of the channel can be loaded in the SPIm.MCSPI_CHxCONF register (with x=0) only when the channel is disabled.

- Two DMA request events, read and write, synchronize read/write accesses of the DMA controller with the activity of McSPI. DMA requests are asserted using the SPIm.MCSPI_CHxCONF[15] DMAR bit (with x=0) for reading and the SPIm.MCSPI_CHxCONF[14] DMAW bit (with x=0) for writing.
- Four interrupt events (see Section 16.5.5.2, *Interrupt Events in Slave Mode*)

16.5.3.2 Slave Transmit-and-Receive Mode

The slave receive mode is programmable (set the SPIm.MCSPI_CHxCONF[13:12] TRM field (with x=0) to 0x0).

In slave transmit-and-receive mode, the MCSPI_TXx register must be loaded before McSPI is selected by an external SPI master device.

After a channel is enabled, transmission and reception proceed with interrupt and DMA request events.

The MCSPI_TXx register content is always loaded in the shift register whether it is updated or not. The event TXx_UNDERFLOW is activated accordingly and does not prevent transmission.

When an SPI word transfer completes (the SPIm.MCSPI_CHxSTAT0[2] EOT bit (with x=0) set to 1), the received data is transferred to the channel receive register.

To use McSPI as a slave transmit-only device, the RXx_FULL and RX0_OVERFLOW interrupts and DMA read requests must be disabled due to the MCSPI_RXx register state (see Section 16.5.5.2, *Interrupt Events in Slave Mode*).

16.5.3.3 Slave Transmit-Only Mode

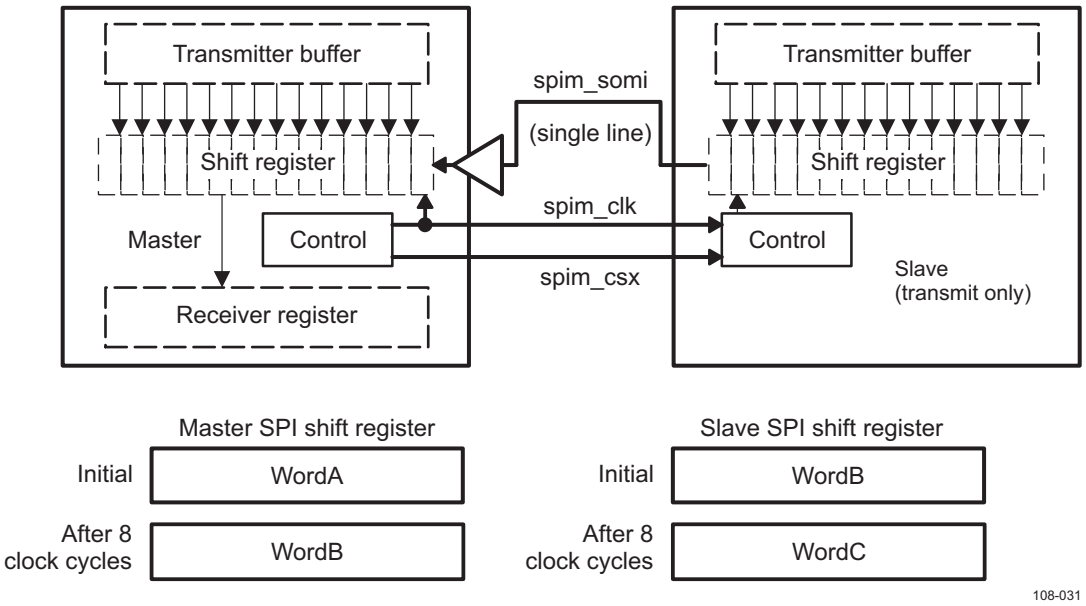
The slave transmit-only mode is programmable (set the SPIm.MCSPI_CHxCONF[13:12] TRM field (with x=0) to 0x2) and avoids the requirement for the MPU to read the MCSPI_RXx register (minimizing data movement) only when transmission is meaningful.

To use the McSPI as a slave transmit-only device, the RXx_FULL and RX0_OVERFLOW interrupts and DMA read requests must be disabled because of the MCSPI_RXx register state.

When the SPI word transfer completes, the SPIm.MCSPI_CHxSTAT[2] EOT bit is set (with x=0).

Figure 16-19 shows a half-duplex system with a master device on the left and a transmit-only slave device on the right. Each time a bit transfers out from the slave device, 1 bit transfers in the master. After eight cycles of the serial clock spim_clk, WordB transfers from the slave to the master.

Figure 16-19. SPI Half-Duplex Transmission (Transmit-Only Slave)



16.5.3.4 Slave Receive-Only Mode

The slave receive mode is programmable (set the SPIm.MCSPI_CHxCONF[13:12] TRM field (with x=0) to 0x1).

In receive-only mode, the MCSPI_TXx register must be loaded before the McSPI is selected by an external SPI master device. The MCSPI_TXx register content is always loaded into the shift register whether it is updated or not. The TXx_UNDERFLOW event is activated accordingly and does not prevent transmission.

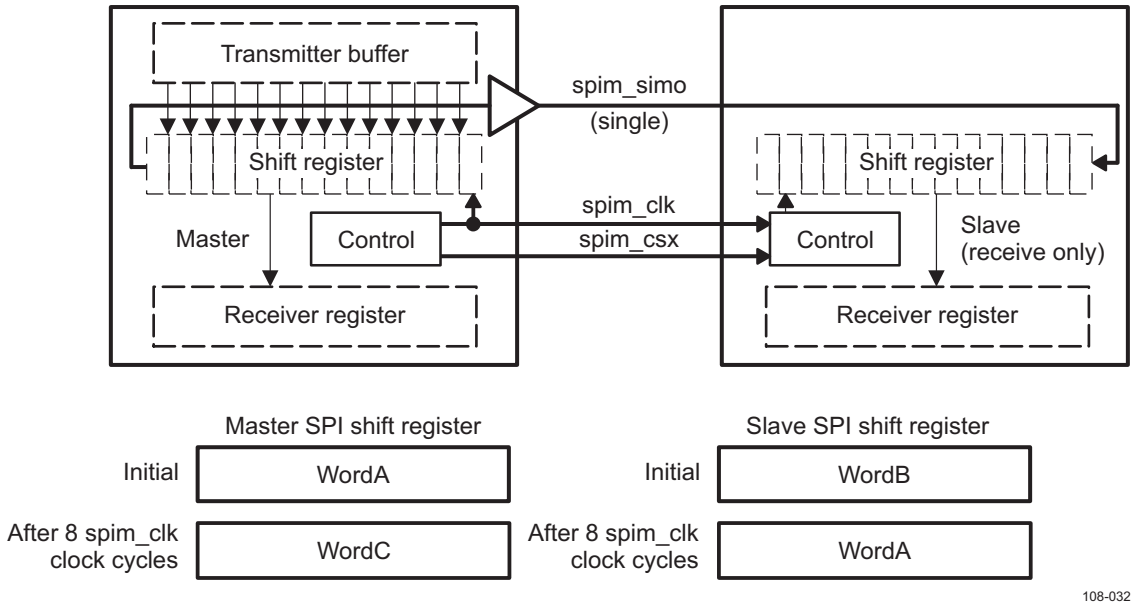
When an SPI word transfer completes (the SPIm.MCSPI_CHxSTAT0[2] EOT bit (with x=0) is set to 1), the received data is transferred to the channel receive register.

To use the McSPI as a slave receive-only device, the TXx_EMPTY and TXx_UNDERFLOW interrupts and the DMA write requests must be disabled due to the MCSPI_TXx register state.

For a full-duplex transmission, the serial clock (spim_clk) synchronizes shifting and sampling of the information on the single serial data line. For full duplex, two data lines are required. If spim_clk synchronizes on a single serial data line, the data line should be half-duplex.

Figure 16-20 shows an example of a half-duplex system with a master device on the left and a receive-only slave device on the right. Each time a bit transfers out from the master, 1 bit transfers in from the slave. After eight cycles of the serial clock spim_clk, Word A transfers from the master to the slave.

Figure 16-20. SPI Half-Duplex Transmission (Receive-Only Slave)



16.5.4 FIFO Buffer Management

The McSPI controller has a built-in 64 bytes buffer to unload DMA or interrupt handler and improve data throughput.

This buffer can be used by only one channel at once and is selected by setting SPIm.MCSPI_CHxCONF[28] FFER bit or SPIm.MCSPI_CHxCONF[27] FFEW bit to 1. If several channel are selected and several FIFO enable bit fields set to 1, the controller forces buffer not to be used, it is the responsibility of the driver to set only one FIFO enable bit field.

The buffer can be used in the modes defined below:

- Master or Slave mode.
- Transmit only, Receive only or Transmit/Receive mode.
- Single channel or turbo mode, or in normal round robin mode. In round robin mode the buffer is used by only one channel.
- Every word length SPIm.MCSPI_CHxCONF[11:7] WL are supported.

In transmit/receive mode, the buffer can be used in transmit (see Figure 16-21) or receive (see Figure 16-22) directions, or in both directions. If only one direction is chosen in transmit/receive mode, the full buffer is used for this direction. In both directions, the buffer is split into two 32-byte buffers, one for each direction. See Figure 16-23.

Figure 16-21. Buffer Use in Transmit Direction Only

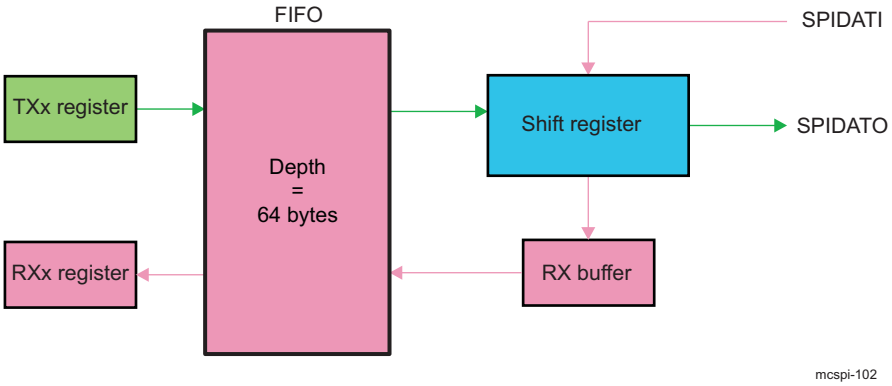


Figure 16-22. Buffer Use in Receive Direction Only

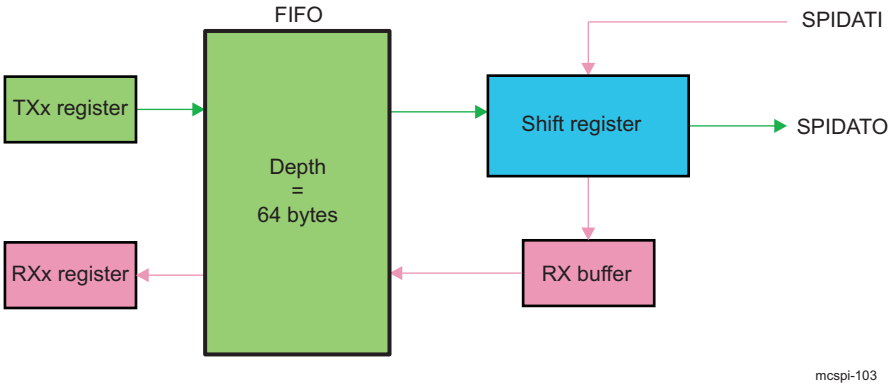
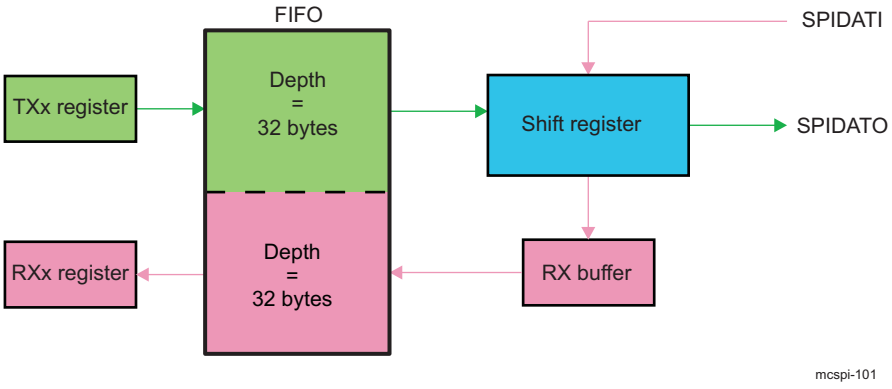


Figure 16-23. Buffer Used For Both Transmit/Receive Directions



Two levels SPIm.MCSPI_XFERLEVEL[5:0] AEL and SPIm.MCSPI_XFERLEVEL[13:8] AFL rule the buffer management. The granularity of these levels is one byte, then it is not aligned with SPI word length. It is the responsibility of the driver to set these values as a multiple of SPI word length defined in WL. Table 16-14 shows the number of byte written in the FIFO depending on the word length.

Table 16-14. FIFO Writes, Word Length Relationship

SPI Word Length WL	3 ≤ WL ≤ 7	8 ≤ WL ≤ 15	16 ≤ WL ≤ 31
Number of byte written in the FIFO	1 byte	2 bytes	4 bytes

The FIFO buffer pointers are reset when the corresponding channel is enabled or FIFO configuration changes.

16.5.4.1 Buffer Almost Full

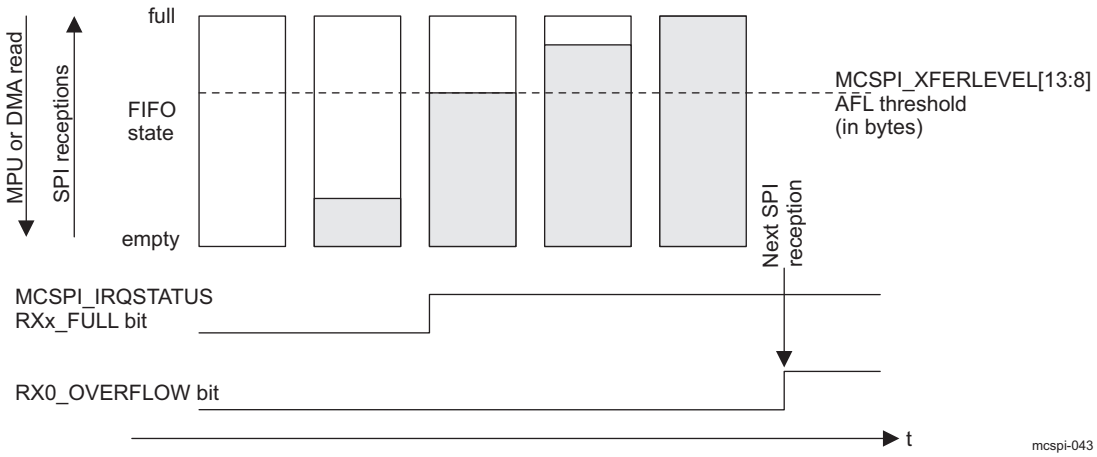
The MCSPI_XFERLEVEL[13:8] AFL bit field is needed when the buffer is used to receive SPI word from a slave (MCSPI_CHxCONF[28] FFER bit must be set to 1). It defines the Almost Full buffer status. See Figure 16-24.

When FIFO pointer reaches this level an interrupt or a DMA request is sent to the MPU to enable system to read AFL+1 bytes from Receive register. Be careful AFL+1 must correspond to a multiple value of MCSPI_CHxCONF[11:7] WL bit field.

When DMA is used, the request is de-asserted after the first Receive register read.

No new request will be asserted again as long as system has not performed the right number of read accesses.

Figure 16-24. Buffer Almost Full Level (AFL)



NOTE: MCSPI_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the SPIm_DMA_RXx request is asserted on the same conditions than the MCSPI_IRQSTATUS RXx_FULL flag.

16.5.4.2 Buffer Almost Empty

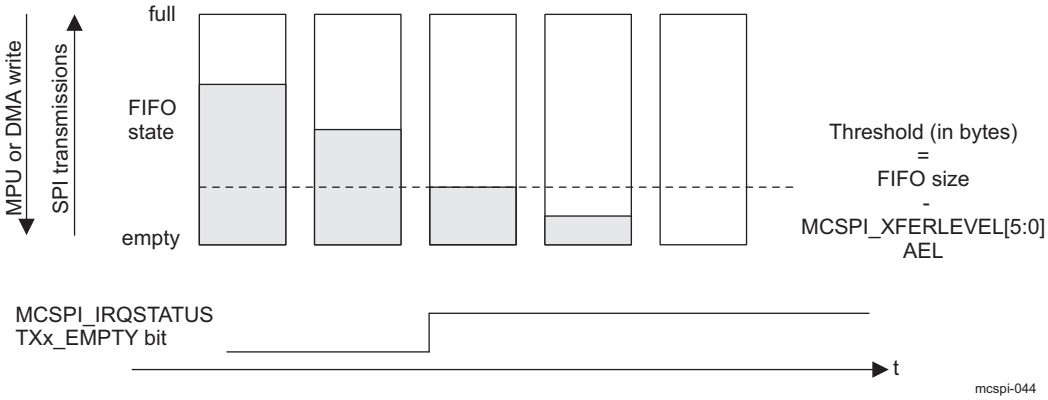
The MCSPI_XFERLEVEL[5:0] AEL bit field is needed when the buffer is used to transmit SPI word to a slave (MCSPI_CHxCONF[27] FFEW bit must be set to 1). It defines the Almost Empty buffer status. See Figure 16-25.

When FIFO pointer has not reached this level an interrupt or a DMA request is sent to the MPU to enable system to write AEL+1 bytes to Transmit register. Be careful AEL+1 must correspond to a multiple value of MCSPI_CHxCONF[11:7] WL bit field.

When DMA is used, the request is de-asserted after the first Transmit register write.

No new request will be asserted again as long as system has not performed the right number of write accesses.

Figure 16-25. Buffer Almost Empty Level (AEL)



NOTE: MCSPI_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the SPIm_DMA_TXx request is asserted on the same conditions than the MCSPI_IRQSTATUS TXx_EMPTY flag.

16.5.4.3 End of Transfer Management

When the FIFO buffer is enabled for a channel, the user shall previously configure in the MCSPI_XFERLEVEL register the AEL and AFL levels and especially the MCSPI_XFERLEVEL[31:16] WCNT bit field to define the number of SPI words to be transferred using the FIFO before enabling the channel.

This counter allows the controller to stop the transfer correctly after a defined number of SPI word transfers. If WNCT is set to 0x0000, the counter is not used and the user must stop the transfer manually by disabling the channel, in this case the user does not know how many SPI transfers have been done. For received words, software shall poll the CHxSTAT[5] RXFFE bit and read the MCSPI_RXx Receive register to empty the FIFO buffer.

When the End Of Word count interrupt is generated (MCSPI_IRQSTATUS[17] EOW bit set), the user can disable the channel and poll the MCSPI_CHxSTAT[5] RXFFE bit to know it lasts SPI words in the FIFO buffer and read them.

No new request will be asserted again as long as system has not performed the right number of write accesses.

NOTE: The RXFFE status bit shows only the FIFO status. The data received are stored not only in the FIFO, but also in the shift register and the MCSPI_RX register. Therefore, the FIFO can be empty even after receiving two words.

16.5.5 Interrupts

Each channel can issue interrupt events.

Each interrupt event has status bits in the SPIm.MCSPI_IRQSTATUS register (RXx_FULL, TXx_UNDERFLOW, TXx_EMPTY, ...) with x = [0,3] that indicate if service is required. Each status bit has an interrupt enable bit (a mask) in the SPIm.MCSPI_IRQENABLE register (RXx_FULL_ENABLE, TXx_UNDERFLOW_ENABLE, TXx_EMPTY_ENABLE, ...).

When an interrupt occurs and a mask is later applied on it, the interrupt line is not asserted again, even if the interrupt source is not serviced.

The McSPI supports interrupt-driven and polling operations.

16.5.5.1 Interrupt Events in Master Mode

In master mode, the interrupt events related to the MCSPI_TXx register state are TXx_EMPTY and TXx_UNDERFLOW. The interrupt event related to the MCSPI_RXx register state is RXx_FULL.

16.5.5.1.1 TXx_EMPTY

The TXx_EMPTY event is activated when a channel is enabled and its MCSPI_TXx register is empty (transient event). Enabling a channel automatically triggers this event, except in master receive-only mode (see Section 16.5.2.4, *Master Receive-Only Mode*). When the FIFO buffer is enabled (MCSPI_CHxCONF[27] FFEW bit set to 1), the MCSPI_IRQSTATUS TXx_EMPTY bit is set as soon as there is enough space in buffer to write a number of bytes defined by the MCSPI_XFERLEVEL[5:0] AEL bit field.

The MCSPI_TXx register must be loaded with data to remove the source of the interrupt; the SPIm.MCSPI_IRQSTATUS TXx_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXx_EMPTY event will be asserted as soon as the MPU has not performed the number of write into the MCSPI_TXx register defined by MCSPI_XFERLEVEL[5:0] AEL bit field. It is the responsibility of the MPU to perform the right number of writes.

16.5.5.1.2 TXx_UNDERFLOW

The event TXx_UNDERFLOW is activated when the channel is enabled and if the MCSPI_TXx register or if the FIFO is empty (not updated with new data) when an external master device starts a data transfer with the McSPI (transmit and receive).

The TXx_UNDERFLOW is a harmless warning in master mode.

To avoid having TXx_UNDERFLOW event at the beginning of a transmission, the event TXx_UNDERFLOW is not activated when no data has been loaded into the MCSPI_TXx register since channel has been enabled. To avoid having TXx_UNDERFLOW event, the MCSPI_TXx register must be loaded seldom.

The SPIm.MCSPI_IRQSTATUS TXx_UNDERFLOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

16.5.5.1.3 RXx_ FULL

The RXx_FULL event is activated when channel is enabled and MCSPI_RXx register becomes filled (transient event). When FIFO buffer is enabled (MCSPI_CHxCONF[28] FFER bit set to 1), the RXx_ FULL is asserted as soon as the number of bytes holds in the FIFO to be read reaches the MCSPI_XFERLEVEL[13:8] AFL threshold.

The MCSPI_RXx register must be read to remove the source of the interrupt; the MCSPI_IRQSTATUS RXx_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXx_FULL event will be asserted as soon as the MPU has not performed AFL+1 reads into MCSPI_RXx. It is the responsibility of MPU to perform the right number of reads.

16.5.5.1.4 End Of Word Count

The MCSPI_IRQSTATUS[17] EOW event (End Of Word count) is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in the MCSPI_XFERLEVEL[31:16] WCNT bit field. If WCNT is set to 0x0000, the counter is not enable and this interrupt is not generated.

The End of Word count interrupt also indicates that the SPI transfer is halt on channel using the FIFO buffer as soon as MCSPI_XFERLEVEL[31:16] WCNT is not reloaded and the channel is not re-enabled.

The MCSPI_IRQSTATUS[17] EOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

16.5.5.2 Interrupt Events in Slave Mode

In slave mode, the interrupt events related to the MCSPI_TXx register state are TXx_EMPTY and TXx_UNDERFLOW. The interrupt events related to the MCSPI_RXx register state are RXx_FULL and RX0_OVERFLOW (channels 1, 2, and 3 do not have a receiver overflow status bit). See the MCSPI_IRQSTATUS register.

16.5.5.2.1 TXx_EMPTY

The TXx_EMPTY event is activated when a channel is enabled and its MCSPI_TXx register is empty. Enabling the channel automatically raises this event. If the FIFO buffer is enabled (MCSPI_CHxCONF[27] FFEW bit set to 1), the TXx_EMPTY event is asserted as soon as there is enough space in buffer to write a number of byte defined by the MCSPI_XFERLEVEL[5:0] AEL bit field.

The MCSPI_TXx register must be loaded with data to remove the source of the interrupt; the SPIm.MCSPI_IRQSTATUS TXx_EMPTY interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new TXx_EMPTY event will be asserted as soon as the MPU has not performed the number of write into the MCSPI_TXx register defined by MCSPI_XFERLEVEL[5:0] AEL bit field. It is the responsibility of the MPU to perform the right number of writes.

16.5.5.2.2 TXx_UNDERFLOW

The TXx_UNDERFLOW event is activated when a channel is enabled and if the MCSPI_TXx register is empty (not updated with new data) when an external master device starts a data transfer with the McSPI (transmit and receive).

When FIFO is enabled, the data emitted while underflow event is raised is not the last data written in the FIFO but the next data in the FIFO (an old transmitted value or a dummy data is the FIFO has been reset).

TXx_UNDERFLOW indicates an error (data loss) in slave mode.

To avoid having a TXx_UNDERFLOW event at the beginning of a transmission, the TXx_UNDERFLOW event is not activated when data is not loaded into the MCSPI_TXx register because the channel is enabled.

The SPIm.MCSPI_IRQSTATUS TXx_UNDERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

16.5.5.2.3 RXx_FULL

The RXx_FULL event is activated when a channel is enabled and the MCSPI_RXx register is being filled (transient event). When FIFO buffer is enabled (MCSPI_CHxCONF[28] FFER bit set to 1), RXx_FULL is asserted as soon as there is a number of bytes holds in buffer to read defined by the MCSPI_XFERLEVEL[13:8] AFL bit field.

The MCSPI_RXx register must be read to remove the source of the interrupt; the SPIm.MCSPI_IRQSTATUS RXx_FULL interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

When FIFO is enabled, no new RXx_FULL event will be asserted as soon as the MPU has not performed AFL+1 reads into MCSPI_RXx. It is the responsibility of MPU to perform the right number of reads.

16.5.5.2.4 RX0_OVERFLOW

The RX0_OVERFLOW event is activated in slave mode in either transmit-and-receive or receive-only mode, when a channel is enabled and the MCSPI_RXx register or FIFO is full when a new SPI word is received. The MCSPI_RXx register is always overwritten with the new SPI word. If the FIFO is enabled data within the FIFO are overwritten, it must be considered as corrupted. The RX0_OVERFLOW event should not appear in slave mode using the FIFO.

The RX0_OVERFLOW indicates an error (data loss) in slave mode.

The MCSPI_IRQSTATUS[3] RX0_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

16.5.5.2.5 End Of Word Count

The MCSPI_IRQSTATUS[17] EOW event (End Of Word count) is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in the MCSPI_XFERLEVEL[31:16] WCNT bit field. If WCNT is set to 0x0000, the counter is not enabled and this interrupt is not generated.

The End of Word count interrupt also indicates that the SPI transfer is halt on channel using the FIFO buffer as soon as WCNT is not reloaded and channel re-enabled.

The MCSPI_IRQSTATUS[17] EOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

16.5.5.3 Interrupt-Driven Operation

An interrupt enable bit, in SPIm.MCSPI_IRQENABLE register, can be set to enable each event to generate interrupt requests when the corresponding event occurs. Status bits are automatically set by hardware logic conditions.

When an event occurs (the single interrupt line is asserted), the MPU must:

- Read the SPIm.MCSPI_IRQSTATUS register to identify which event occurred.
- Read the MCSPI_RXx register that corresponds to the event to remove the source of an RXx_FULL event or write into the MCSPI_TXx register that corresponds to the event to remove the source of a TXx_EMPTY event. No action is required to remove the source of the WKS (wake-up), TXx_ UNDERFLOW, and RX0_OVERFLOW events.
- Write 1 into the corresponding bit of the SPIm.MCSPI_IRQSTATUS register to clear an interrupt status and then release the interrupt line.

The interrupt status bit must always be reset after channel enabling and before events are enabled as interrupt sources.

16.5.5.4 Polling

When the interrupt capability of an event is disabled in the SPIm.MCSPI_IRQENABLE register, the interrupt line is not asserted, but the status bits in the SPIm.MCSPI_IRQSTATUS register can be polled by software to detect when the corresponding event occurs.

Once the expected event occurs:

- RXx_FULL: To remove the source of the event, the MPU must read the corresponding MCSPI_RXx register.
- TXx_EMPTY: To remove the source of the event, the MPU must write into the corresponding MCSPI_TXx register.
- WKS (wake-up), TXx_UNDERFLOW, and RX0_OVERFLOW: No action is required to remove the source of the event.

To clear an interrupt, set the corresponding status bit of the SPIm.MCSPI_IRQSTATUS register to 1. This does not affect the interrupt line state.

16.5.6 DMA Requests

The sDMA controller module manages DMA accesses. The sDMA controller advantage is to lower the MPU charge for data transfers.

Each McSPI channel can issue DMA requests if they are enabled. There are two DMA request lines per McSPI channel (one for read and one for write).

The DMA read request line is asserted when the McSPI channel is enabled and new data is available in the receive register of the McSPI channel. A DMA read request can be individually masked with the SPI1.MCSPI_CHxCONF[15] DMAR bit. The DMA read request line is deasserted on read completion of the MCSPI_RXx register of the McSPI channel.

The DMA write request line is asserted when the McSPI channel is enabled and the MCSPI_TXx register of the McSPI channel is empty. A DMA write request can be individually masked with the SPI1.MCSPI_CHxCONF[14] DMAW bit. The DMA write request line is deasserted on load completion of the MCSPI_TXx register of the channel.

16.5.7 Power Saving Management

Power consumption can be optimized by switching off internal clocks (interface and functional clock) when there is no activity. The McSPI is compliant with the idle and wake-up system handshake protocol.

16.5.7.1 Normal Mode

In normal mode, internal SPI module clocks are automatically switched off (autogating) when there is no activity in slave or master mode.

Autogating of the module interface clock and functional clock occurs when the following conditions are met:

- The SPIm.MCSPI_SYSCONFIG[0] AUTOIDLE bit is set.
- In master mode, there is no data to transmit or receive in all channels.
- In slave mode, the McSPI is not selected by the external master and there are no register accesses.

Autogating of the module interface clock and functional clock stops when the following conditions are met:

- In master mode, an internal access occurs.
- In slave mode, an internal access occurs or the McSPI is selected by the external master.

16.5.7.2 Idle Mode

At the PRCM module level, when all conditions are met to shut off the CORE_48M_FCLK or CORE_L4_ICLK output clocks (see , *Power, Reset, and Clock Management*, for details), the PRCM module automatically launches a hardware handshake protocol to ensure that the McSPI is ready to have its clocks switched off. Namely, the PRCM module asserts an idle request to the McSPI.

Although this handshake is completely hardware-oriented and out of software control, the method in which the McSPI module acknowledges the PRCM idle request is configurable through the McSPI.SYSCONFIG[4:3] SIDLEMODE bit.

The following list details the settings of the SIDLEMODE bit and the related acknowledgment modes:

- Force-idle mode (the SPIm.MCSPI_SYSCONFIG[4:3] SIDLEMODE bit set to 0x0): the McSPI module acknowledges unconditionally the idle request from the PRCM module, regardless of its internal operations. This mode must be used carefully in this case because it does not prevent the loss of data when the clock is switched off.
- No-idle mode the (SIDLEMODE bit set to 0x1): The McSPI module never acknowledges an idle request from the PRCM module and is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient to save power because it does not allow the PRCM output clock to be shut off and thus the power domain to be set to a lower power state.
- Smart-idle mode (the SIDLEMODE bit set to 0x2): The McSPI module acknowledges the idle request, basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, IRQs, or DMA requests are treated. This is the best approach for an efficient system power management.

When configured in smart-idle mode, the McSPI module also offers an additional granularity on the CORE_48M_FCLK and CORE_L4_ICLK gating. The SPIm.SYSCONFIG[9:8] CLOCKACTIVITY bit field determines which clock shuts down (the CORE_48M_FCLK, the CORE_L4_ICLK, neither clock, or both clocks).

The CLOCKACTIVITY setting is used internally to McSPI to determine on which part of the module the conditions to acknowledge the PRCM idle request are tested. For example, if CORE_48M_FCLK is not shut down on a PRCM idle request, McSPI considers only CORE_L4_ICLK and the associated pending activities before acknowledging the request.

Some McSPI features are associated with CORE_L4_ICLK and others with CORE_48M_FCLK. Using the CLOCKACTIVITY bit field along with the smart-idle mode ensures that the features associated with the clock that remains active are always enabled, even if McSPI acknowledges an idle request.

The following list details CLOCKACTIVITY settings and the associated features:

- CLOCKACTIVITY set to 00: ICLK OFF and FCLK OFF, both ICLK and FCLK are taken into account for generating the acknowledge. This setting also means that both FCLK and ICLK are likely to be shut down on a PRCM idle request.
- CLOCKACTIVITY set to 01: ICLK ON and FCLK OFF, ICLK is not shut down on a PRCM idle request; only FCLK is concerned.
- CLOCKACTIVITY set to 10: ICLK OFF and FCLK ON, FCLK is not shut down on a PRCM idle request; only ICLK is concerned.
- CLOCKACTIVITY set to 11: ICLK ON and FCLK ON, none of the clocks are shut down. This means McSPI can potentially acknowledge the idle request without checking the internal functionalities linked to its clocks.

CAUTION

The PRCM module does not have a hardware means of reading the CLOCKACTIVITY settings. Therefore, the software must ensure consistent programming between the CLOCKACTIVITY and the PRCM CORE_48M_FCLK and CORE_L4_ICLK control bits. If the McSPI is disabled in both the CM_FCLKEN and CM_ICLKEN PRCM registers while CLOCKACTIVITY is set to 11, nothing prevents the PRCM module from asserting its idle request, which is acknowledged regardless of the features associated with the McSPI clocks. This can lead to unpredictable behavior.

16.5.7.2.1 Wake-Up Event in Smart-Idle Mode

The module wake-up feature is enabled when both the SPIm.MCSPI_SYSCONFIG[2] ENAWAKEUP and SPIm.MCSPI_WAKEUPENABLE[0] WKEN bits are set. Wake-up capability is relevant only when the module is configured in slave mode.

The module generates an asynchronous wake-up request to the system power manager to switch the interface clock and the functional clock back. A wake-up is requested when channel 0 is enabled and an asynchronous selection occurs on the spim.csx port associated with channel 0 (see the definition for the SPIm.MCSPI_CHxCONF SPIENSLV field (with x=0) in the register description table).

After the McSPI wake-up request, the system power manager must reactivate the interface clock:

- *Before the beginning* of the second SPI word serialization when McSPI is in slave transmit-only mode or in slave transmit-and-receive mode
- *Before the end* of the second received SPI word in slave receive-only mode. To avoid data loss, the first received SPI word must be read from the SPIm.MCSPI_RXx register (with x=0) before the completion of the second SPI word serialization.

Table 16-15 lists the supported cases in wake-up mode.

Table 16-15. Smart-Idle Mode and Wake-Up Capabilities

Mode	Interface Clock	SPI Clock Ref	Functionality	Wake-Up Event
Master	Must be maintained	Must be maintained	Full functionality, but the module does not generate a new interrupt or DMA request until the system exits wake-up mode.	No wake-up event

Table 16-15. Smart-Idle Mode and Wake-Up Capabilities (continued)

Mode	Interface Clock	SPI Clock Ref	Functionality	Wake-Up Event
Slave	Can be switched off	Can be switched off	An SPI word can be transmitted and/or received, but the module does not generate any new interrupts or DMA requests until the system exits wake-up mode.	The module asynchronously sends a wake-up request if an event on the spim_csx port associated to channel 0 is detected.

In wake-up mode, the interrupt and DMA request lines are no longer asserted.

Any access to the module in wake-up mode generates an error as long as the interface clock is alive.

16.5.7.2.2 Transitions From Smart-Idle Mode to Normal Mode

The McSPI detects the end of the wake period through the idle and wake-up hardware handshake protocol.

The interrupt status register (the SPIm.MCSPI_IRQSTATUS[16] WKS bit) is updated with the event causing the wake-up; the wake-up event at the origin of the transition to the normal mode is converted to its corresponding interrupt when enabled by the SPIm.MCSPI_IRQENABLE[16] WKE bit or the DMA request.

Interrupts and wake-up events have independent enable/disable controls, accessible through the SPIm.MCSPI_IRQENABLE and SPIm.MCSPI_WAKEUPENABLE registers. Software must ensure the overall consistency.

The interrupt status register SPIm.MCSPI_IRQSTATUS is updated with the event causing the wake-up; the wake-up event at the origin of the transition to normal mode is converted to its corresponding interrupt request or DMA request. The module is fully operational.

16.5.7.2.3 Force-Idle Mode

Force-idle mode is enabled and exited as follows:

- Force-idle mode is enabled when the SPIm.MCSPI_SYSCONFIG[4:3] SIDLEMODE field is set to 0x0.

  In force-idle mode, McSPI responds unconditionally to the idle request by deasserting unconditionally the interrupt and DMA request lines, if asserted. In addition, the wake-up capability is totally inhibited even if both the SPIm.MCSPI_SYSCONFIG[2] ENAWAKEUP and SPIm.MCSPI_WAKEUPENABLE[0] WKEN bits are set.

  The transition from normal mode to idle mode does not affect the interrupt event bits of the SPIm.MCSPI_IRQSTATUS register.

  In force-idle mode, the module must be disabled so the interrupt and DMA request lines are likely deasserted. The interface clock and SPI clock provided to the McSPI can be switched off.

  An idle request during an SPI data transfer can lead to an unexpected and unpredictable result. The software must avoid such a request.
- The module exits force-idle mode through the idle and wake-up hardware handshake protocol.

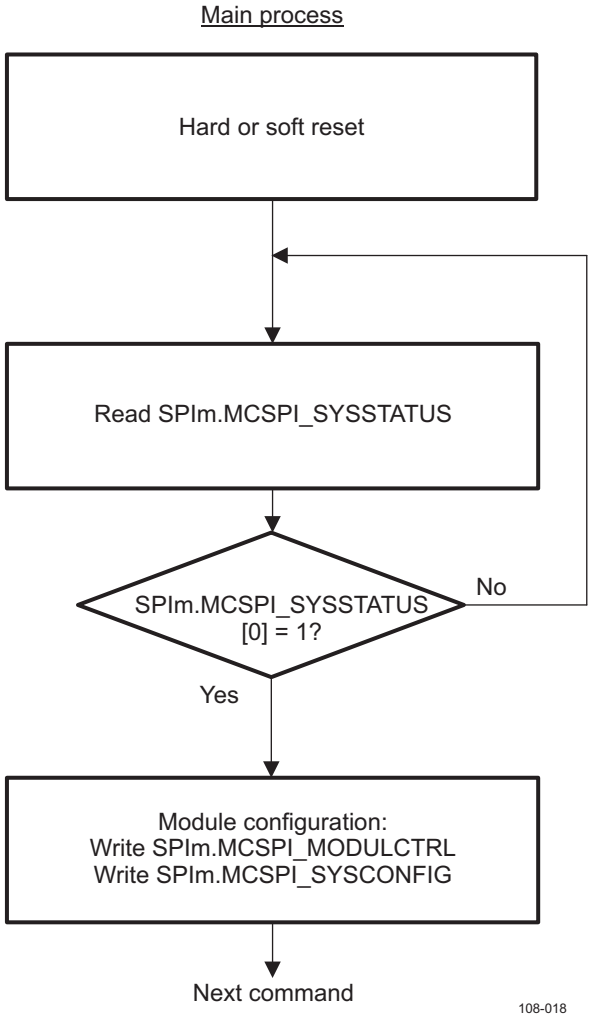
  The module is fully operational. The interrupt and DMA request lines are optionally asserted one clock cycle later.

16.6 McSPI Basic Programming Model

16.6.1 Initialization of Modules

Figure 16-26 shows the overview of the module initialization flow process. Section 16.6.2 and Section 16.6.3 show the steps required to configure McSPI modes.

Figure 16-26. Module Initialization Flow



NOTE: Before the SPIm.MCSPI_SYSSTATUS[0] RESETDONE bit is set, the CLK and CLKSPIREF clocks must be provided to the module.

To avoid unpredictable behavior, reset the module before changing from master mode to slave mode, or vice versa.

16.6.2 Transfer Procedures without FIFO

In the subsections below, the transfer procedures are described without FIFO using (MCSPI_CHxCONF[27:28] FFER and FFEW = 0).

The McSPI allows the transfer of one or more words based on the different modes:

- Master normal, master turbo, slave
- Transmit-and-receive, transmit-only, receive-only
- Write and read requests: Interrupts, DMA

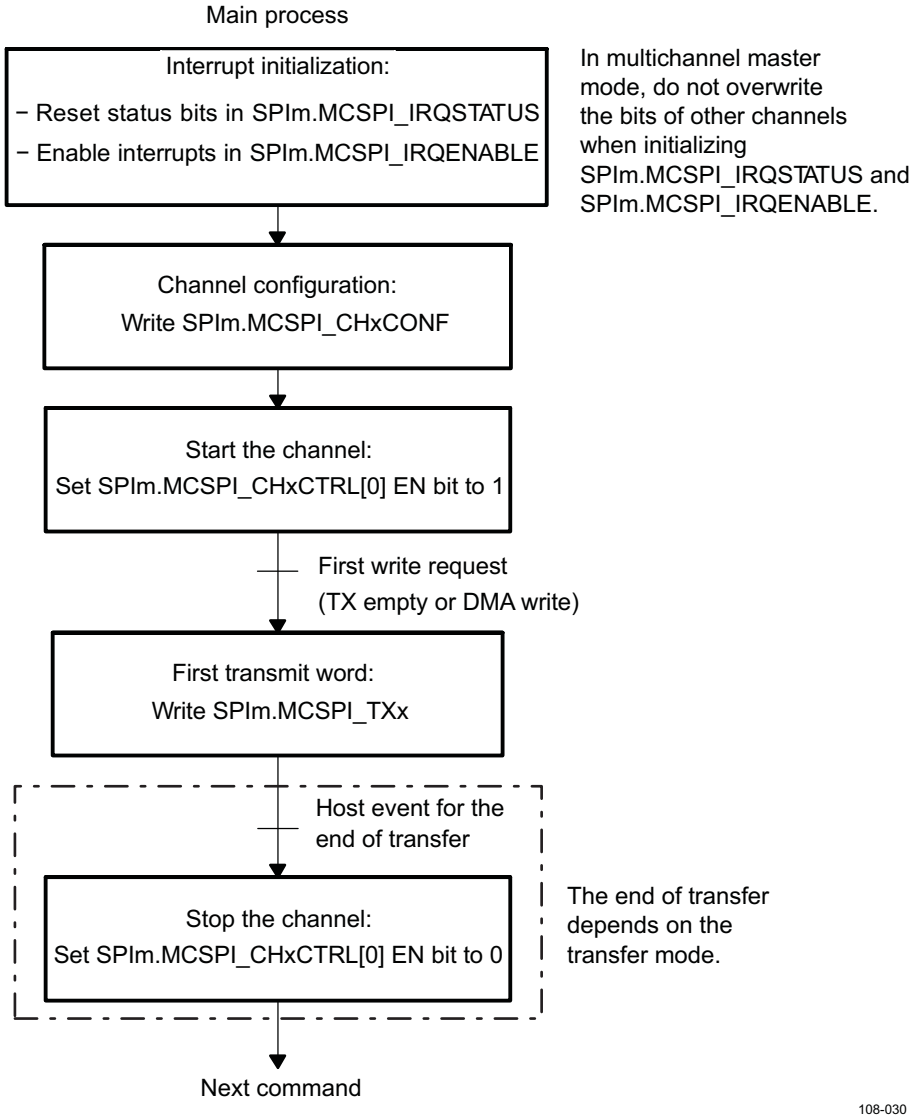
- spi1_csx line assertion/deassertion: Automatic, manual

For these flows, the host process contains the main process and the interrupt routines. The interrupt routines are called on the interrupt signals or by an internal call if the module is used in polling mode.

16.6.2.1 Common Transfer Procedure

Figure 16-27 shows the main sequence common to all transfers. In multichannel master mode, the flows of different channels can be run simultaneously.

Figure 16-27. Common Transfer Sequence: Main Process



16.6.2.2 End-of-Transfer Procedure

For transfers carried out with DMA or interrupt mode, the end of transfer must be achieved by following the steps shown in the flowcharts corresponding to Table 16-16, which summarizes the end-of-transfer types per transfer mode and provides cross-references for further information.

Table 16-16. End-of-Transfer Sequences

		Transmit Receive		Transmit Only		Receive Only	
		Interrupt	DMA	Interrupt	DMA	Interrupt	DMA
Master normal	End of transfer sequence	See Section 16.6.2.3		See Section 16.6.2.4	See Section 16.6.2.4	See Section 16.6.2.5.1	See Section 16.6.2.5.1
	Minimum number of words	1	1	1	1	1	1
	DMA transfer size[1]		w		w		w – 1
Master turbo	End of transfer sequence	See Section 16.6.2.3		See Section 16.6.2.4	See Section 16.6.2.4	See Section 16.6.2.5.2	See Section 16.6.2.5.2
	Minimum number of words	1	1	1	1	2	3
	DMA transfer size[1]		w		w		w – 2
Slave	End of transfer sequence	See Section 16.6.2.3		See Section 16.6.2.4	See Section 16.6.2.4	See Section 16.6.2.5.3	
	Minimum number of words	1	1	1	1	1	1
	DMA transfer size[2]		w		w	w	w

(1) w = number of words to transfer
(2) w = number of words to transfer

The different sequences can be merged in one process to manage transfers of several types. The end-of-transfer sequences are described from the start of the channel.

In these sequences and in later sections of this chapter, some software variables are used:

- WRITE_COUNT (= 0 at initialization): Contains the number of words to transfer
- READ_COUNT (= 0 at initialization): Contains the number of words to receive
- CHANNEL_ENABLE (= false at initialization)
- LAST_TRANSFER (= false at initialization): Indicates that the last word is in transmission
- LAST_REQUEST (= false at initialization): Indicates that the last request is in progress

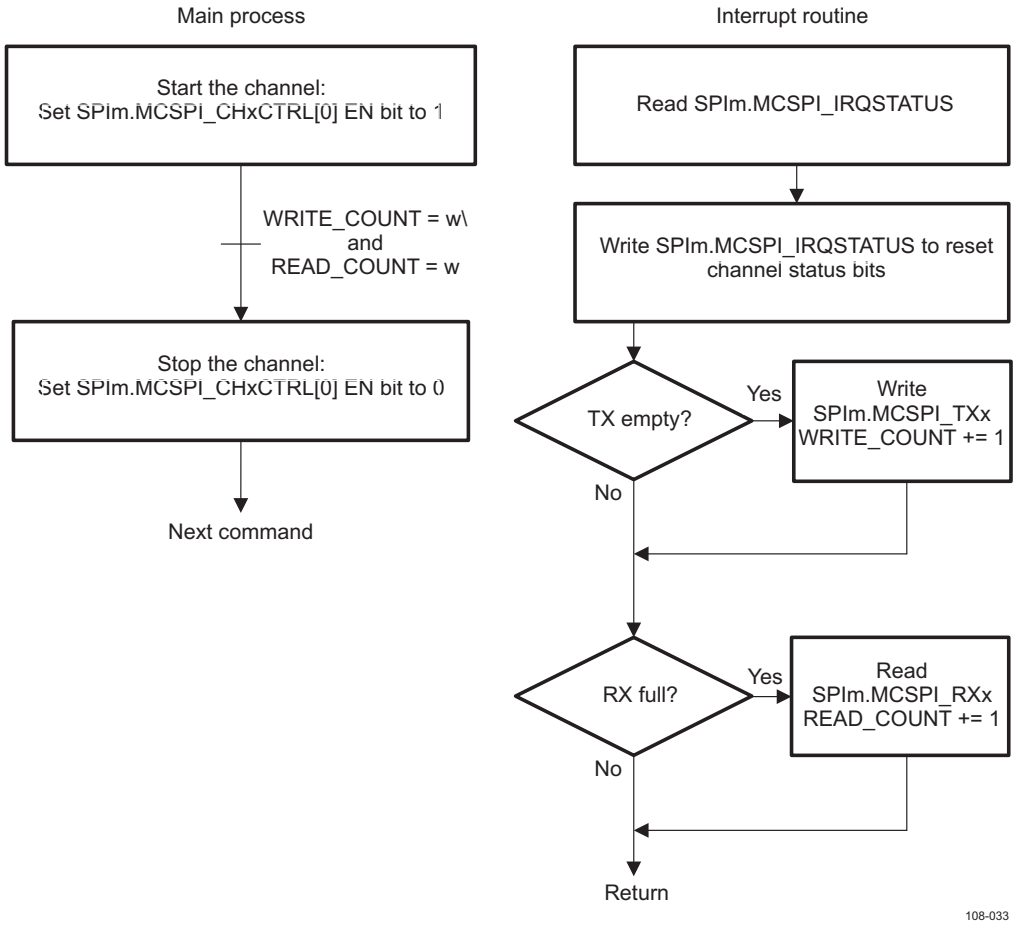
All variables are initialized before starting the channel.

16.6.2.3 Transmit and Receive Procedure

Figure 16-28 shows the handling procedure for words received and transmitted by interrupt in master and slave modes. The main process flow shows how the end of the transfer must be done after all words are received for this mode.

If the requests are configured in DMA, WRITE_COUNT and READ_COUNT are assigned with the value w when the DMA handler completes w interface accesses.

Figure 16-28. Transmit and Receive (Master and Slave)

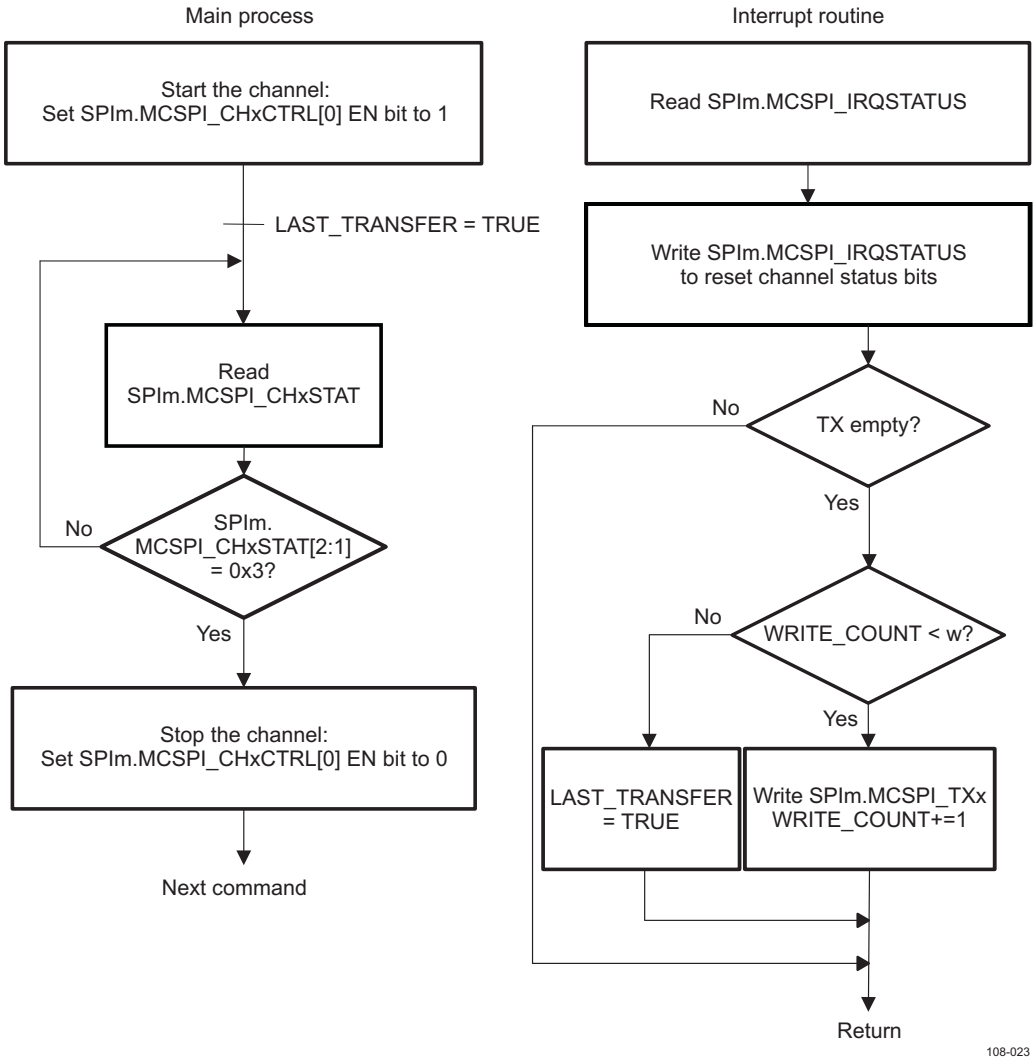


16.6.2.4 Transmit-Only Procedure

16.6.2.4.1 Based on Interrupt Requests

Figure 16-29 shows the handling procedure for words transmitted by interrupt in transmit-only mode. The main process flow shows how the end-of-transfer must be done after all words are received for this mode.

Figure 16-29. Transmit-Only With Interrupts (Master and Slave)

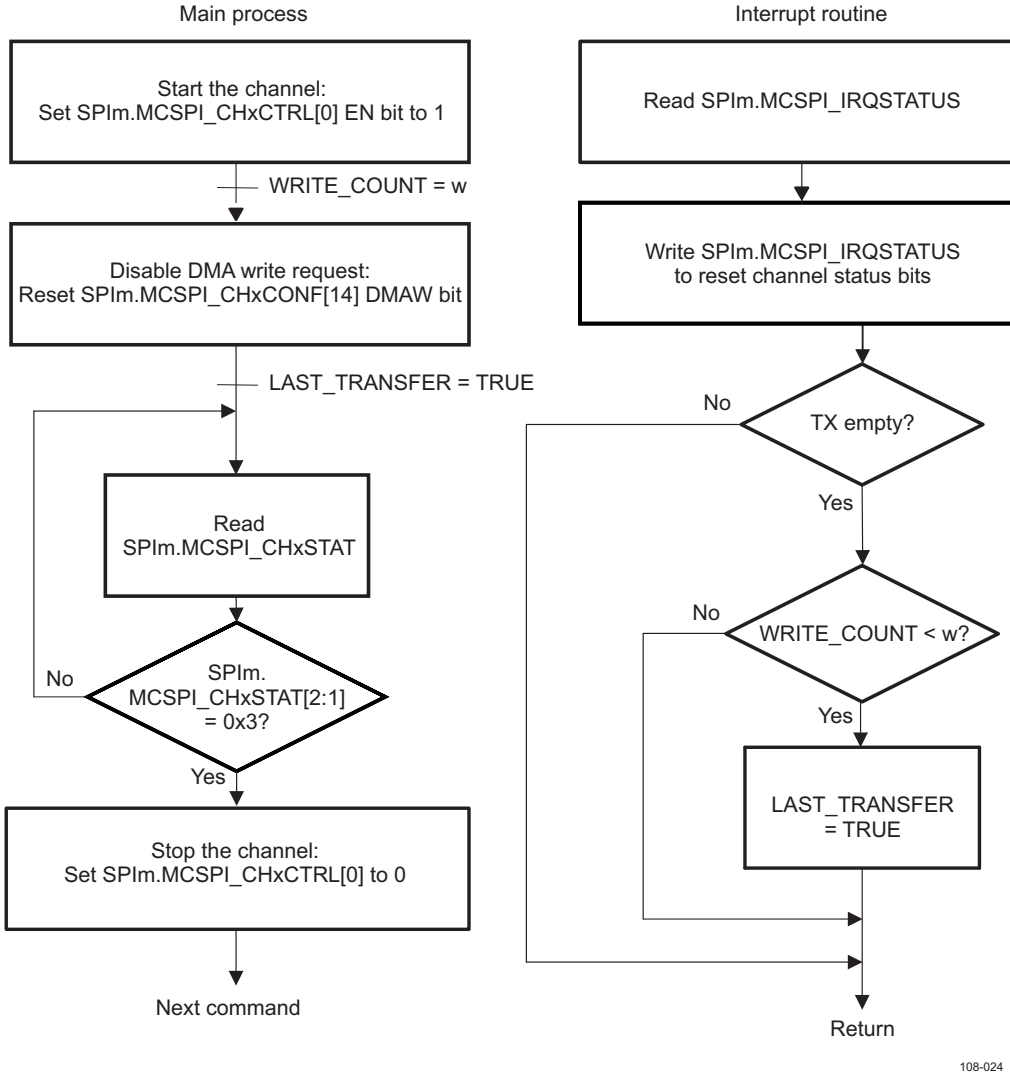


16.6.2.4.2 Transmit-Only Based on DMA Write Requests

In Figure 16-30, the main process shows completion of the transfer of words in transmit-only mode with DMA write requests.

When the DMA handler completes w interface accesses, WRITE_COUNT is assigned the value *w*.

Figure 16-30. Transmit-Only With DMA (Master and Slave)



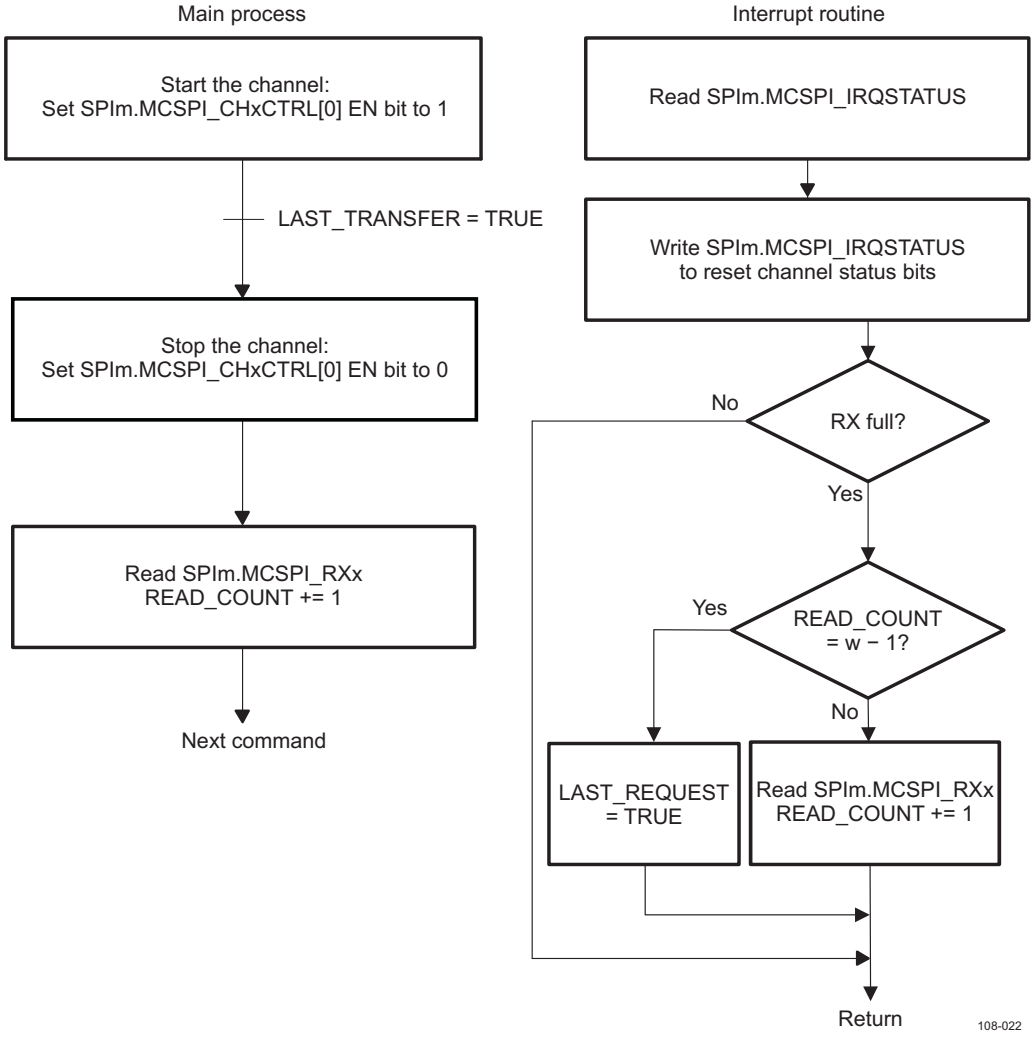
16.6.2.5 Receive-Only Procedure

16.6.2.5.1 Master Normal Receive-Only Procedure

16.6.2.5.1.1 Based on Interrupt Requests

Figure 16-31 shows the handling procedure for words received by interrupt in master normal receive-only mode. The main process flow shows how the end-of-transfer must be done after all words are received for this mode.

Figure 16-31. Receive Only With Interrupt (Master Normal)

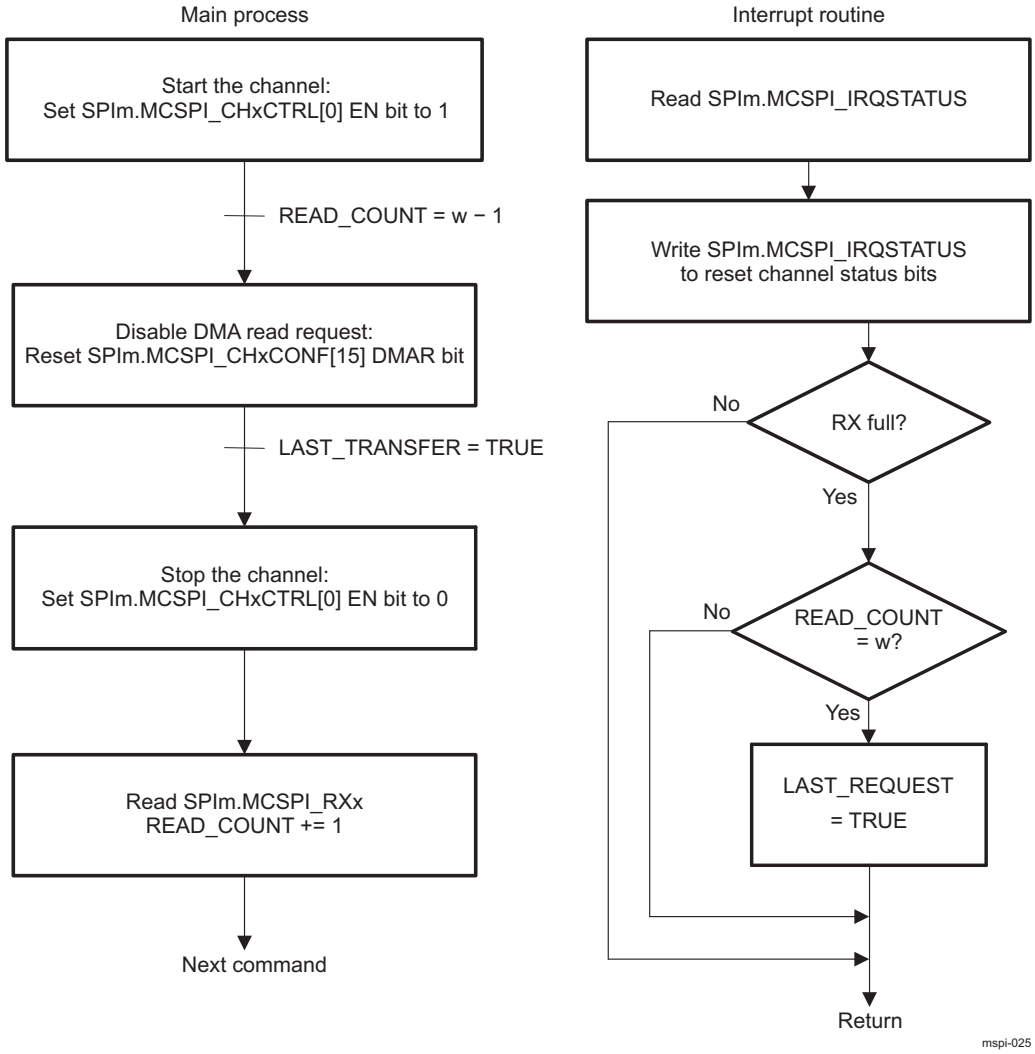


16.6.2.5.1.2 Receive-Only Based on DMA Read Requests

In Figure 16-32, the main process shows the completion of a word transfer in receive-only mode with DMA read requests.

When the DMA handler completes *w* - 1 interface accesses, READ_COUNT is assigned the value *w* - 1.

Figure 16-32. Receive-Only With DMA (Master Normal)

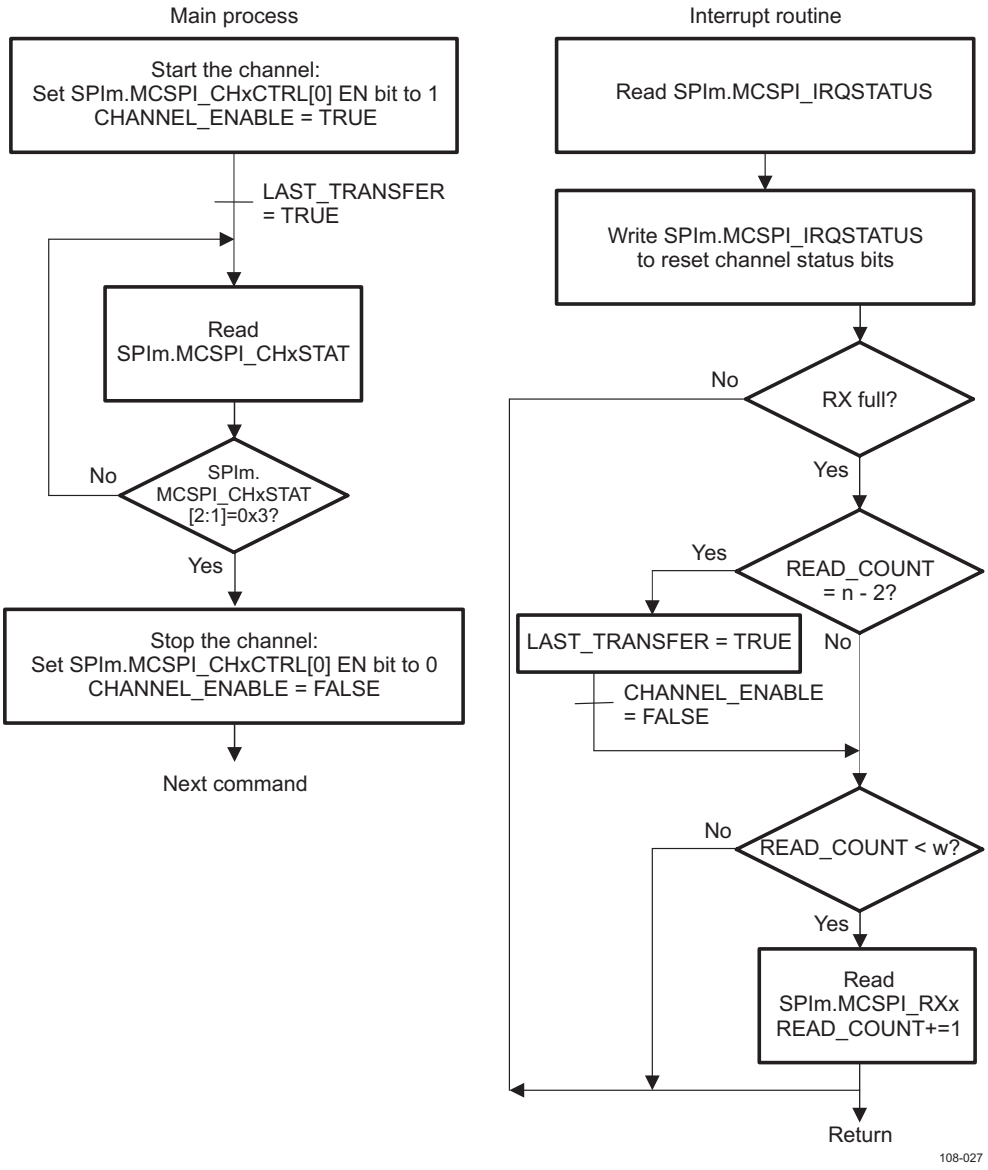


16.6.2.5.2 Master Turbo Receive-Only Procedure

16.6.2.5.2.1 Based on Interrupt Requests

Figure 16-33 shows the handling procedure for words received by interrupt in master turbo receive-only mode. The main process shows how the end-of-transfer must be done after all words are received for this mode.

Figure 16-33. Receive-Only With Interrupt (Master Turbo)

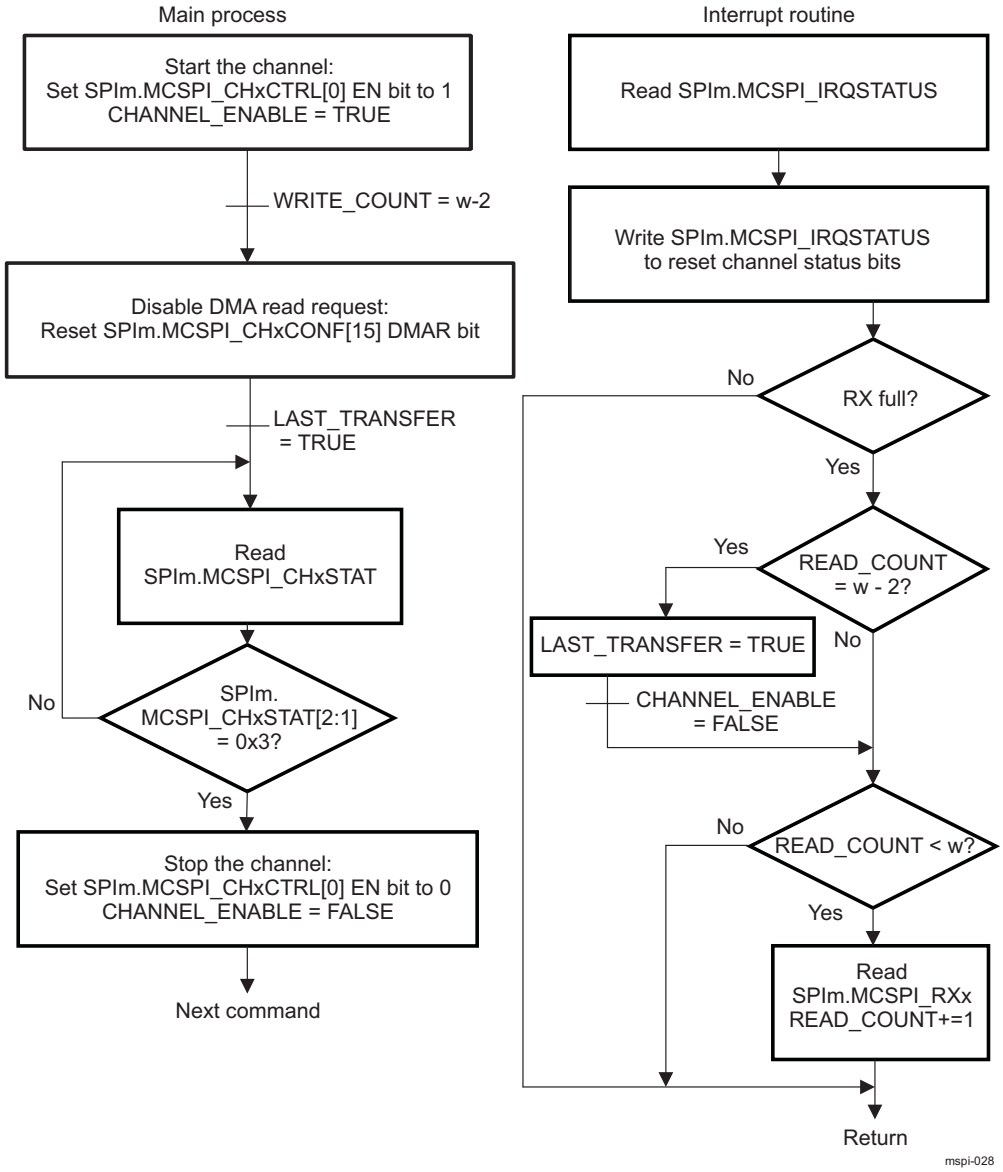


16.6.2.5.2.2 Based on DMA Read Requests

In Figure 16-34, the main process shows the completion of a word reception in master turbo receive-only mode with DMA write requests.

When the DMA handler completes *w* - 2 interface accesses, READ_COUNT is assigned the value *w* - 2.

Figure 16-34. Receive-Only With DMA (Master Turbo)

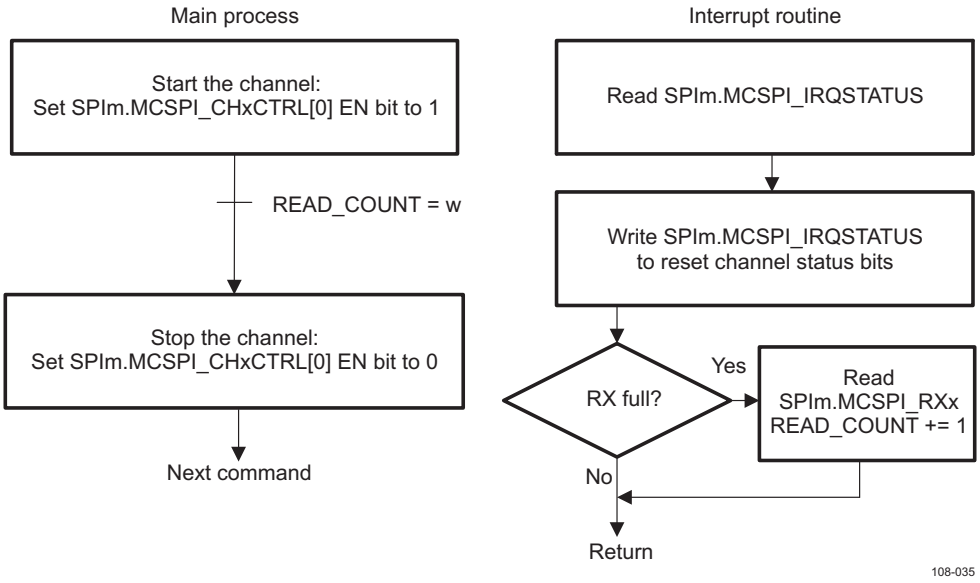


16.6.2.5.3 Slave Receive-Only Procedure

Figure 16-35 shows the handling procedure for words received by interrupt in slave receive-only mode. The main process shows how the end-of-transfer must be done after all words are received for this mode.

If the requests are configured in DMA, READ_COUNT is assigned the value *w* when the DMA handler completes *w* interface processes.

Figure 16-35. Receive Only (Slave)



16.6.2.6 McSPI Configuration and Operations Example

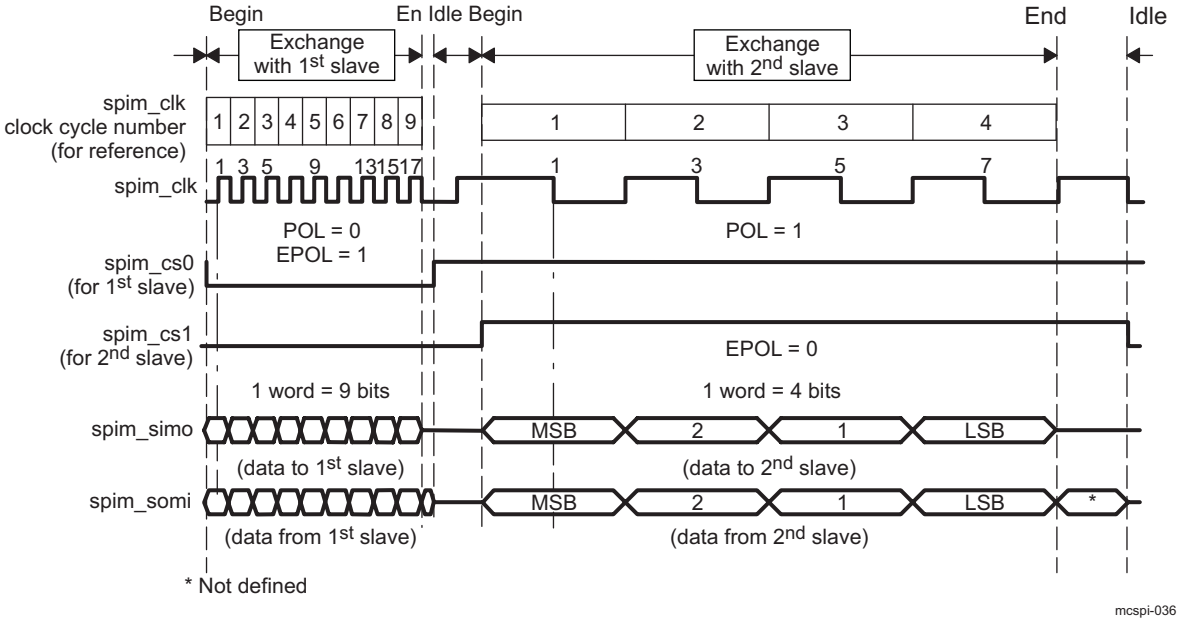
This section details an example of a typical configuration.

Figure 16-36 shows an implementation of successive transfers between two devices (slave) and the McSPI1 (master) in transmit-and-receive mode.

McSPI1 is the master, and spi1_simo shifts out the data to the external slaves. spi1_somi is connected to the data output port of two slave devices. The McSPI1 controls the first device with the spi1_cs0 signal (active low) for the exchange of 9-bit words synchronized with an active-high SPI clock.

The second device is controlled by the spi1_cs1 control signal (active high) for the exchange of 4-bit words synchronized with a SPI clock (active low) with a slower frequency than used with the first slave device.

Figure 16-36. Two SPI Transfers With PHA = 0 (Flexibility of McSPI)



This section details the steps required for this type of transmission.

16.6.2.6.1 McSPI Initialization Sequence

As shown in Figure 16-26, the McSPI module must first reset all registers and all state-machines.

For a software reset:

1. Set the SPI1.MCSPI_SYSCONFIG[1] SOFTRESET bit to 1.
2. Read the SPI1.MCSPI_SYSSTATUS[0] RESETDONE bit and check that it is set to 1.

16.6.2.6.2 Operations for the First Slave (On Channel 0)

16.6.2.6.2.1 Programming in Polling Mode

16.6.2.6.2.1.1 Mode Selection

The SPI1.MCSPI_CHxCONF register (with x = 0) allows configuration of the operating mode:

1. Set the SPI1.MCSPI_CHxCONF[18] IS bit to 0 for the spi1_somi pin in receive mode.
2. Set the SPI1.MCSPI_CHxCONF[17] DPE1 bit to 0 and the SPI1.MCSPI_CHxCONF[16] DPE0 bit to 1 for the spi1.simo pin in transmit mode.
3. Set the SPI1.MCSPI_CHxCONF[13:12] TRM field to 0x0 for transmit and receive mode.
4. Write 0x8 in the SPI1.MCSPI_CHxCONF[11:7] WL field for 9-bit word length.

5. Set the SPI1.MCSPI_CHxCONF[6] EPOL bit to 1 for spi1_cs0 activated low during active state.
6. Set the SPI1.MCSPI_CHxCONF[1] POL bit to 0 for spi1_clk held high during active state.
7. Set the SPI1.MCSPI_CHxCONF[0] PHA bit to 0 for data latched on odd-numbered edges of the SPI clock.

   Clock Initialization and spi1_cs0 Enable

   In master mode, the SPI must provide the clock and enable the channel:
8. Set the SPI1.MCSPI_MODULCTRL[2] MS bit to 0 to provide the clock.
9. Set the INPUTENABLE bit of the corresponding CONTROL_PADCONF_x register to achieve synchronous clock by using loopback through the output and input buffers of the pad.
10. Set the SPI1.MCSPI_CHxCTRL[0] EN bit to 1 (where x = 0) to enable channel 0.

16.6.2.6.2.1.2 Write Operation

1. Write 1 to the SPI1.MCSPI_IRQSTATUS[0] TX0_EMPTY bit to reset the status.
2. Write the command/address or data value in the SPI1.MCSPI_TX0 register to transmit the value.
3. If the SPI1.MCSPI_IRQSTATUS[0] TX0_EMPTY bit is set to 1, write 1 to it and return to Step 2 (polling method).

16.6.2.6.2.1.3 Read Operation

1. Read the SPI1.MCSPI_IRQSTATUS[2] RX0_FULL bit and if it is set to 1, go to Step 2.
2. If the SPI1.MCSPI_IRQSTATUS[2] RX0_FULL bit is set to 1, write 1 to it and return to Step 1 (polling method).

NOTE: Write and read operations can be performed simultaneously.

16.6.2.6.3 Programming in Interrupt Mode

This section follows the flow of Figure 16-26.

1. Initialize software variables: WRITE_COUNT = 0 and READ_COUNT = 0.
2. Initialize interrupts: Write 0x7 in the SPI1.MCSPI_IRQSTATUS[3:0] field and set the SPI1.MCSPI_IRQENABLE[3:0] field to 0x7.
3. Follow the steps described in Section 16.6.2.6.2.1.1, *Mode Selection*.
4. If WRITE_COUNT = w and READ_COUNT = w, write SPI1.MCSPI_CHxCTRL[0] = 0x0 (x = 0) to stop the channel.

This interrupt routine follows the flow of Table 16-16 and Figure 16-28.

1. Read the SPI1.MCSPI_IRQSTATUS[3:0] field.
2. If the SPI1.MCSPI_IRQSTATUS[0] TX0_EMPTY bit is set to 1:

   (a) Write the command/address or data value in SPI1.MCSPI_TXx (where x = 0).

   (b) WRITE_COUNT + = 1

   (c) Write SPI1.MCSPI_IRQSTATUS[0] = 0x1.
3. If the SPI1.MCSPI_IRQSTATUS[2] RX0_FULL bit is set to 1:

   (a) Read SPI1.MCSPI_RXx (where x = 0)

   (b) READ_COUNT += 1

   (c) Write SPI1.MCSPI_IRQSTATUS[2] = 0x1

16.6.2.6.4 Operations for the Second Slave (on Channel 1) in Polling Mode

16.6.2.6.4.1 Mode Selection

The SPI1.MCSPI_CHxCONF register (with x = 1) allows configuration of the operating mode:

1. Set the SPI1.MCSPI_CH1CONF[18] IS bit to 0 for the spi1_somi pin in receive mode.
2. Set the SPI1.MCSPI_CH1CONF[17] DPE1 bit to 0 and the SPI1.MCSPI_CH1CONF[16] DPE0 bit to 1 for the spi1_somi pin in transmit mode.
3. Set the SPI1.MCSPI_CH1CONF[13:12] TRM field to 0x0 for transmit-and-receive mode.
4. Write 0x3 in the SPI1.MCSPI_CH1CONF[11:7] WL field.
5. Set the SPI1.MCSPI_CH1CONF[6] EPOL bit to 0 for spi1_cs1 activated high during the active state.
6. Set the SPI1.MCSPI_CH1CONF[1] POL bit to 1 for spi1_clk held low during the active state.
7. Set the SPI1.MCSPI_CH1CONF[0] PHA bit to 1 for data latched on even numbered edges of spi1_clk.

   Clock Initialization and spi1_cs1 Enable

   The SPI1.MCSPI_MODULCTRL[2] MS bit was set to 0 (providing the clock).

   In master mode, the SPI must provide the clock and enable the channel:
8. Set the SPI1.MCSPI_CH0CTRL[0] EN bit to 0 to disable channel 0, and set the SPI1.MCSPI_CH1CTRL[0] EN bit to 1 to enable channel 1.

NOTE: Read and write operations for the second slave are identical to those for the first slave.

16.6.2.6.4.2 Write Operation

1. Write 1 to the SPI1.MCSPI_IRQSTATUS[4] TX1_EMPTY bit to reset the status.
2. Write the command/address or data value in the SPI1.MCSPI_TX1 register to transmit the value.
3. If the SPI1.MCSPI_IRQSTATUS[4] TX1_EMPTY bit is set to 1, write 1 to it and return to Step 2 (polling method).

16.6.2.6.4.3 Read Operation

1. Read the SPI1.MCSPI_IRQSTATUS[6] RX1_FULL bit and if it is set to 1, go to Step 2.
2. If the SPI1.MCSPI_IRQSTATUS[6] RX1_FULL bit is set to 1, write 1 to it and return to Step 1 (polling method).

NOTE: Write and read operations can be performed simultaneously.

16.6.3 Transfer Procedures with FIFO

In the subsections below, the transfer procedures are described with FIFO using (MCSPI_CHxCONF[27:28] FFER or/and FFEW = 1).

The MCSPI module allows the transfer of one or more words, according to different modes:

- Master normal, master turbo, slave
- Transmit-and-receive, transmit-only, receive-only
- Write and read requests: Interrupts, DMA

For these flows, the host process contains the main process and the interrupt routine, which is called on the IRQ signals or by an internal call if the module is used in polling mode.

16.6.3.1 Common Transfer Procedure

The common transfer sequence is the host sequence for a transfer of any type defined above.

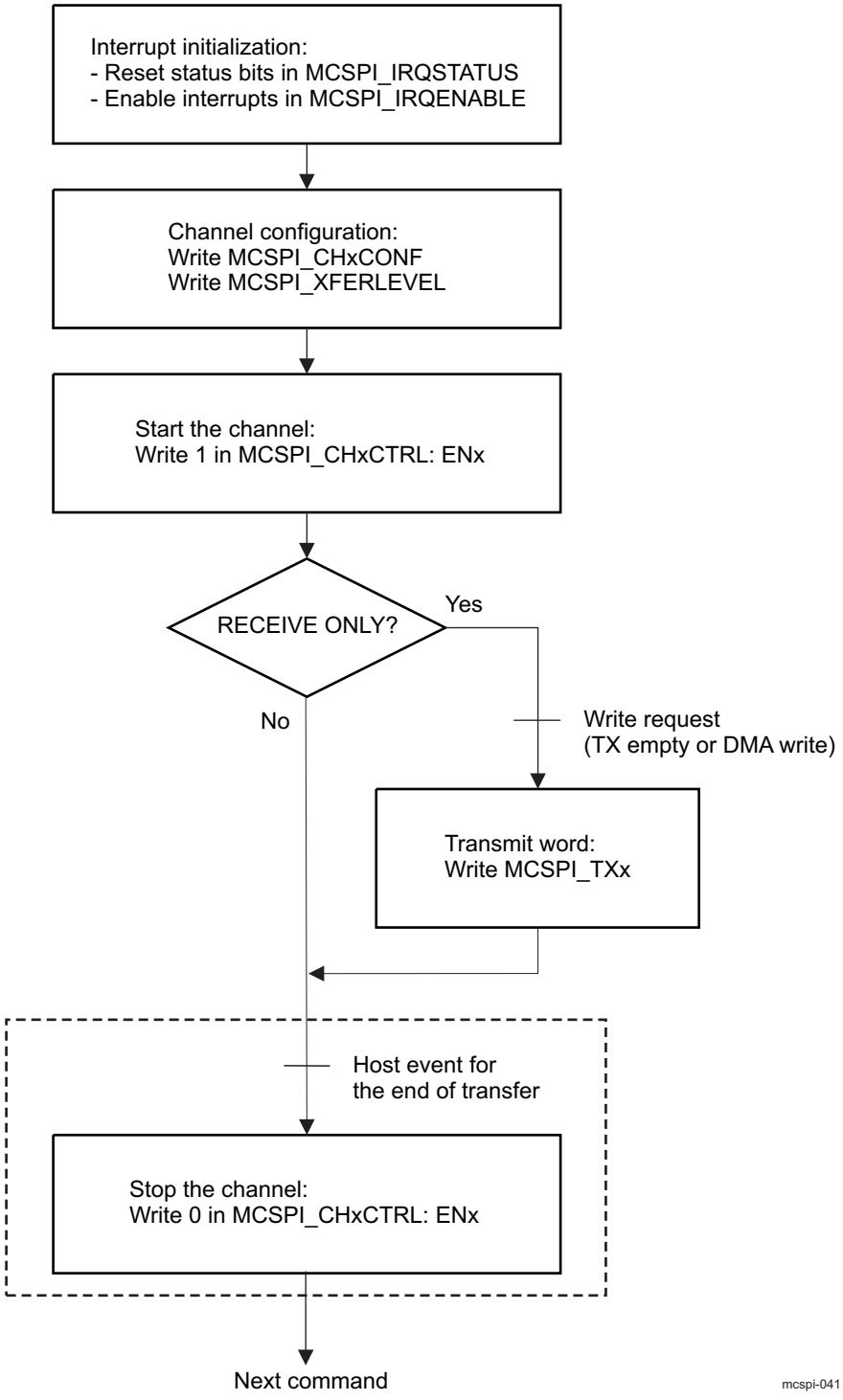
In multichannel, only one channel can use the FIFO. Before enabling the FIFO for a channel (FFExW and FFExR bits in the MCSPI_CHx_CONF register), the host must ensure that the FIFO is not enabled for another channel, even if these channels are not used.

In transmit/receive mode, the FIFO can be enabled for write or read request only, without FIFO for the other request.

In slave mode, only channel 0 only can be activated. The correct spim_csx line is chosen in the MCSPI_CH0CONF register with the SPIENSLV bits.

The MCSPI module can start the transfer only when the first write request is released by writing the MCSPI_TXx register, even in receive-only mode (only one write request occurs in this case). Figure 16-37 shows the main process of the common transfer sequence.

Figure 16-37. Common Transfer Sequence/Main Process



The MCSPI module can start the transfer only after the first write request is released by writing the MCSPI_TXx register, even in receive-only mode (only one write request occurs in this case).

This first write request can be managed by the IRQ routine or DMA handler, as are as other requests. It appears in Figure 16-37 to show this point.

The end of the transfer (dotted line in Figure 16-37) is more complex and depends on the transfer type. Table 16-17 describes the different types of end-of-transfer and the transfer types to which they are applicable.

Table 16-17. End-of-Transfer Types

Word Count	Transmit/Receive	Transmit Only	Receive Only
Yes	See Section 16.6.3.2, *Transmit-Receive With Word Count*.	See Section 16.6.3.4, *Transmit-Only*.	See Section 16.6.3.5 , *Receive-Only With Word Count*.
No	See Section 16.6.3.3, *Transmit-Receive Without Word Count*.	See Section 16.6.3.4, *Transmit-Only*.	See Section 16.6.3.6, *Receive-Only Without Word Count*.

The sequence differs depending on whether word count is used (MCSPI_XFERLEVEL: WCNT is set or not). The AEL and/or AFL values can be different, but they must be multiples of the word size in the FIFO: 1, 2, or 4 bytes, according to word length.

In these sequences, the transfer to execute has a size of N words.

When accessing the FIFO, only one word is written or read for each OCP access.

In these sequences, the number of words written or read for each write or read FIFO request is:

- write_request_size
- read_request_size

If they are not submultiples of N, the last request sizes are:

- last_write_request_size (< write_request_size)
- ast_read_request_size. (< read_request_size)

The different sequences can be merged in one process to manage transfers of several types.

The end-of-transfer sequences are described from the start of the channel.

In these sequences, some soft variables are used:

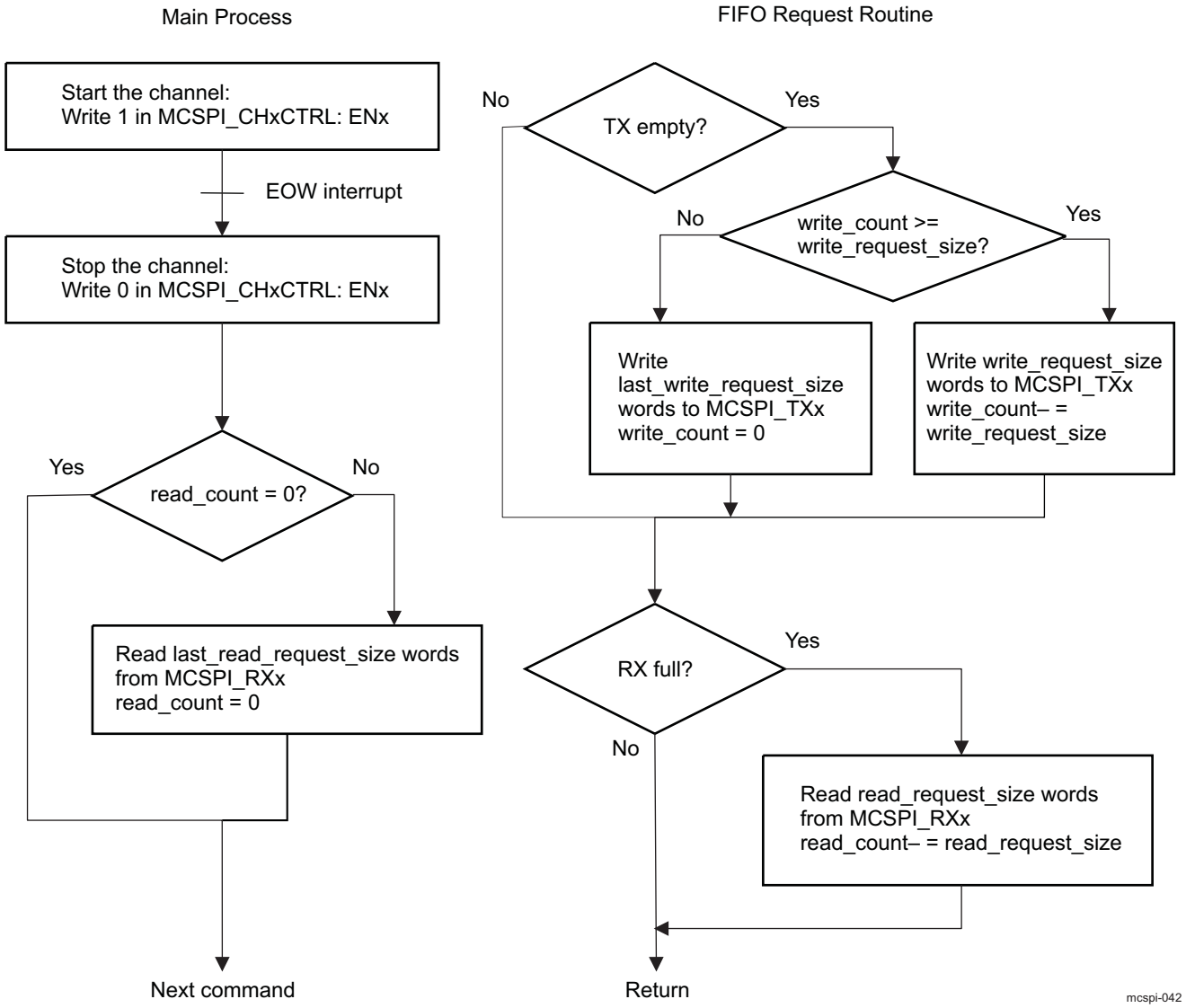
- write_count = N
- read_count = N
- last_request = FALSE

They are initialized before starting the channel.

16.6.3.2 Transmit-Receive Procedure With Word Count (WCNT≠0)

Figure 16-38 shows the flow of a transfer in transmit-receive mode, with word count.

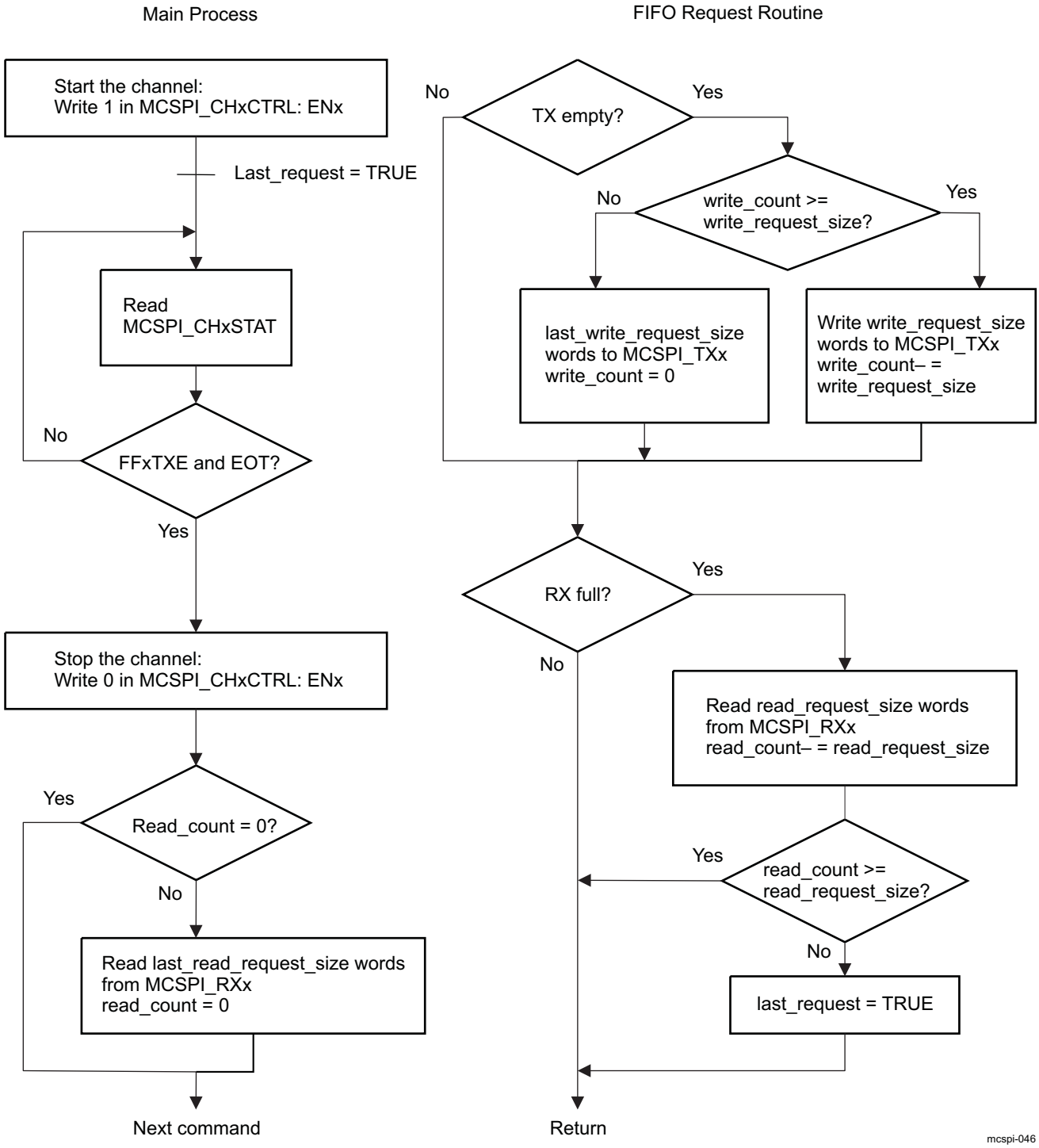
Figure 16-38. Transmit-Receive With Word Count



16.6.3.3 Transmit-Receive Procedure Without Word Count (WCNT=0)

Figure 16-39 shows the flow of a transfer in transmit-receive mode, without word count.

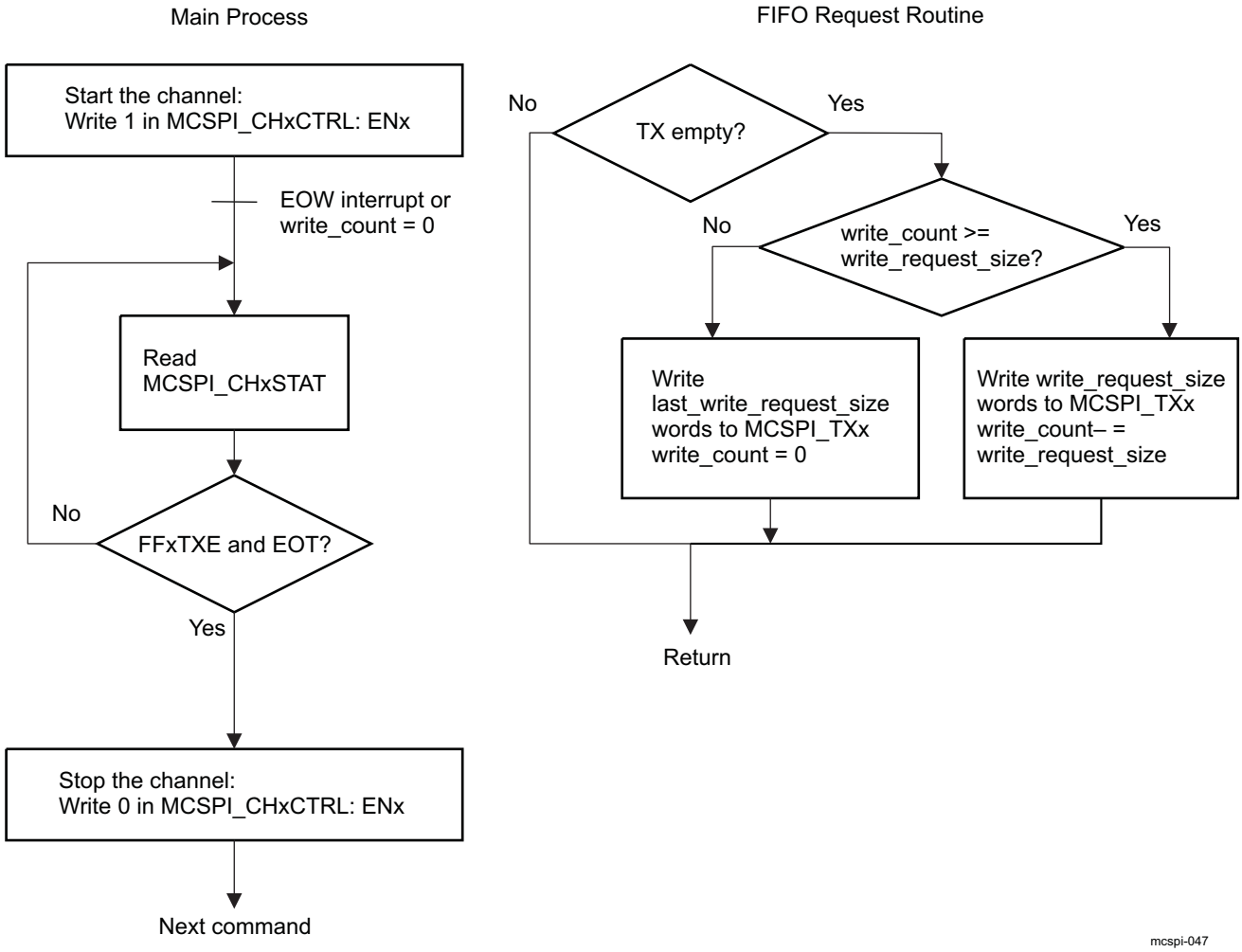
Figure 16-39. Transmit-Receive Without Word Count



16.6.3.4 Transmit-Only Procedure

Figure 16-40 shows the flow of a transfer in transmit only mode, with our without word count.

Figure 16-40. Transmit-Only



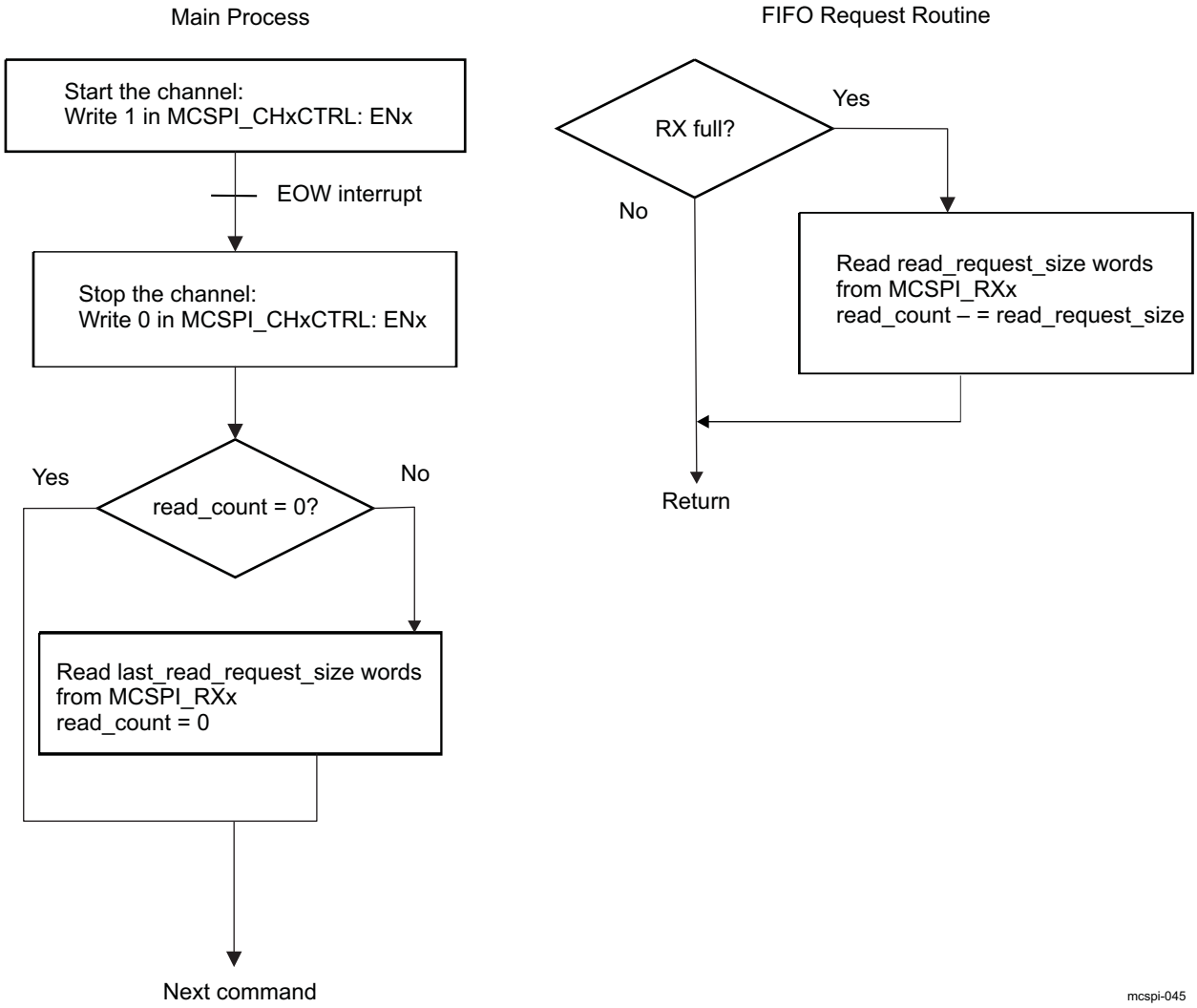
The difference between word count enabled or not is the condition after starting the channel:

- Word count enable: Wait for EOW interrupt.
- Word count disable: Wait for write_count = 0.

16.6.3.5 Receive-Only Procedure With Word Count (WCNT≠0)

Figure 16-41 shows the flow of a transfer in receive-only mode, with word count.

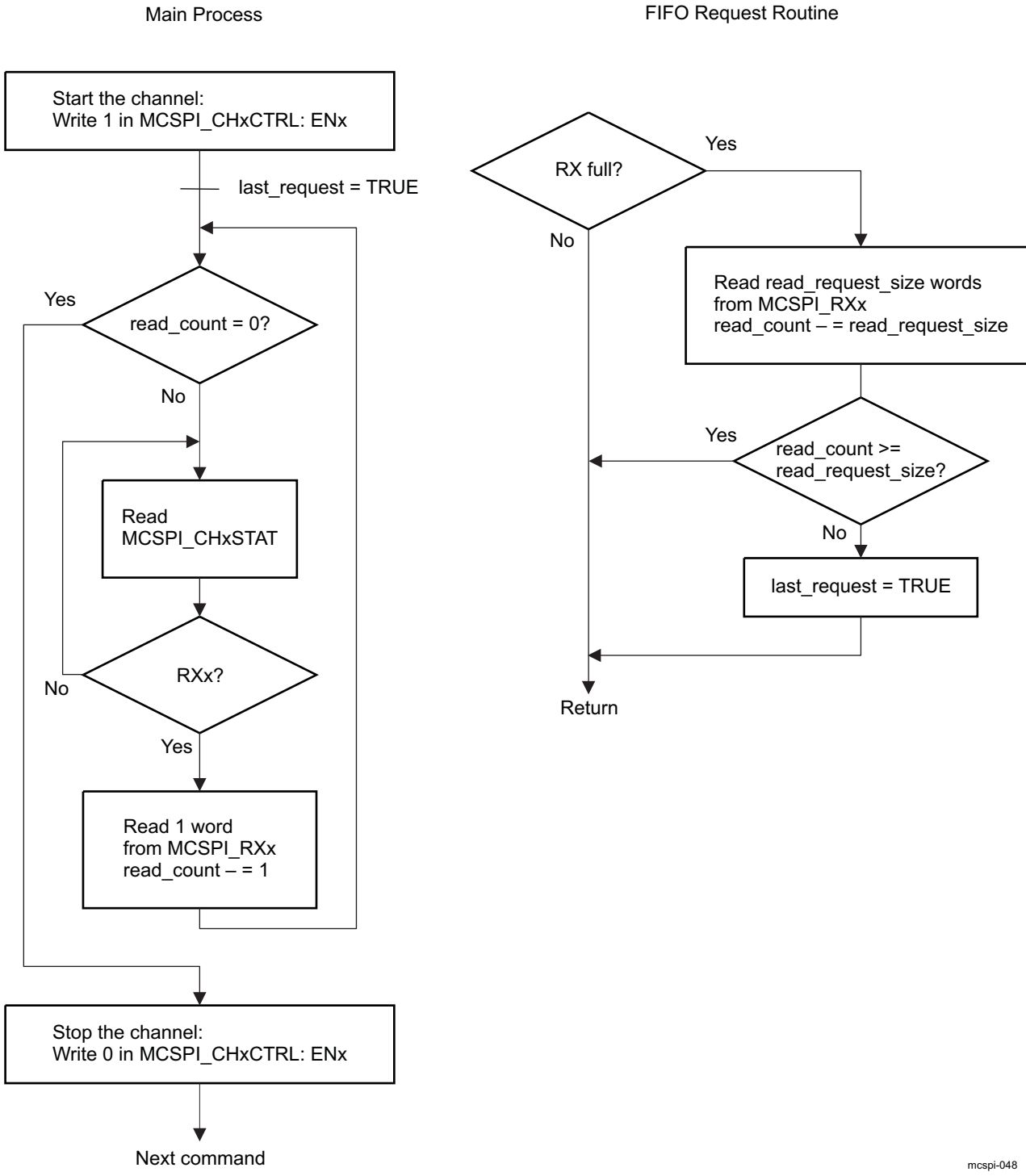
Figure 16-41. Receive-Only With Word Count



16.6.3.6 Receive-Only Procedure Without Word Count (WCNT=0)

Figure 16-42 shows the flow of a transfer in receive-only mode, without word count.

Figure 16-42. Receive-Only Without Word Count



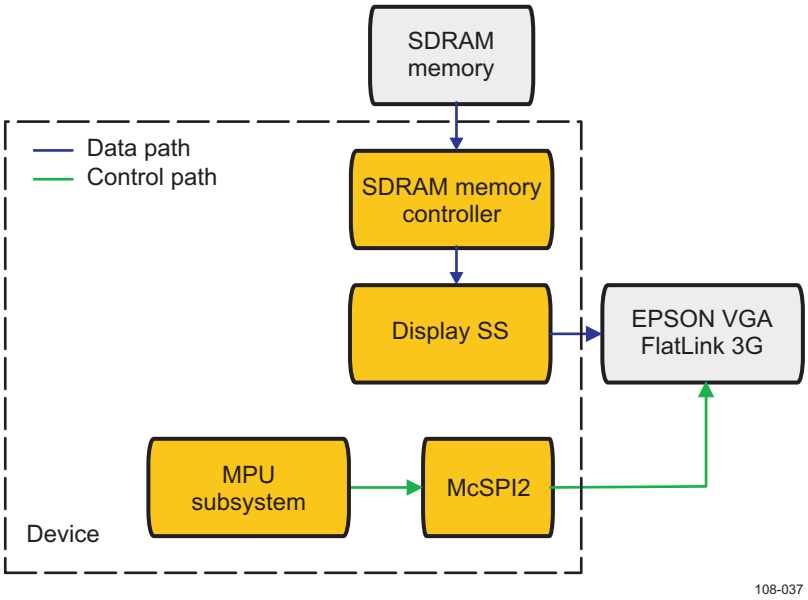
16.7 McSPI Use Cases and Tips

16.7.1 How to Configure the McSPI Interface When Connected with an EPSON VGA FlatLink™ 3G Device

16.7.1.1 Overview

This section deals with configuring an EPSON VGA FlatLink™3G device through the McSPI interface. Before transferring data from the external SDRAM memory to the external EPSON VGA display (Figure 16-43), the display subsystem must be set up via the McSPI interface. Several commands must be sent to the EPSON VGA for its configuration.

Figure 16-43. Overview



In this use case the configuration is the following:

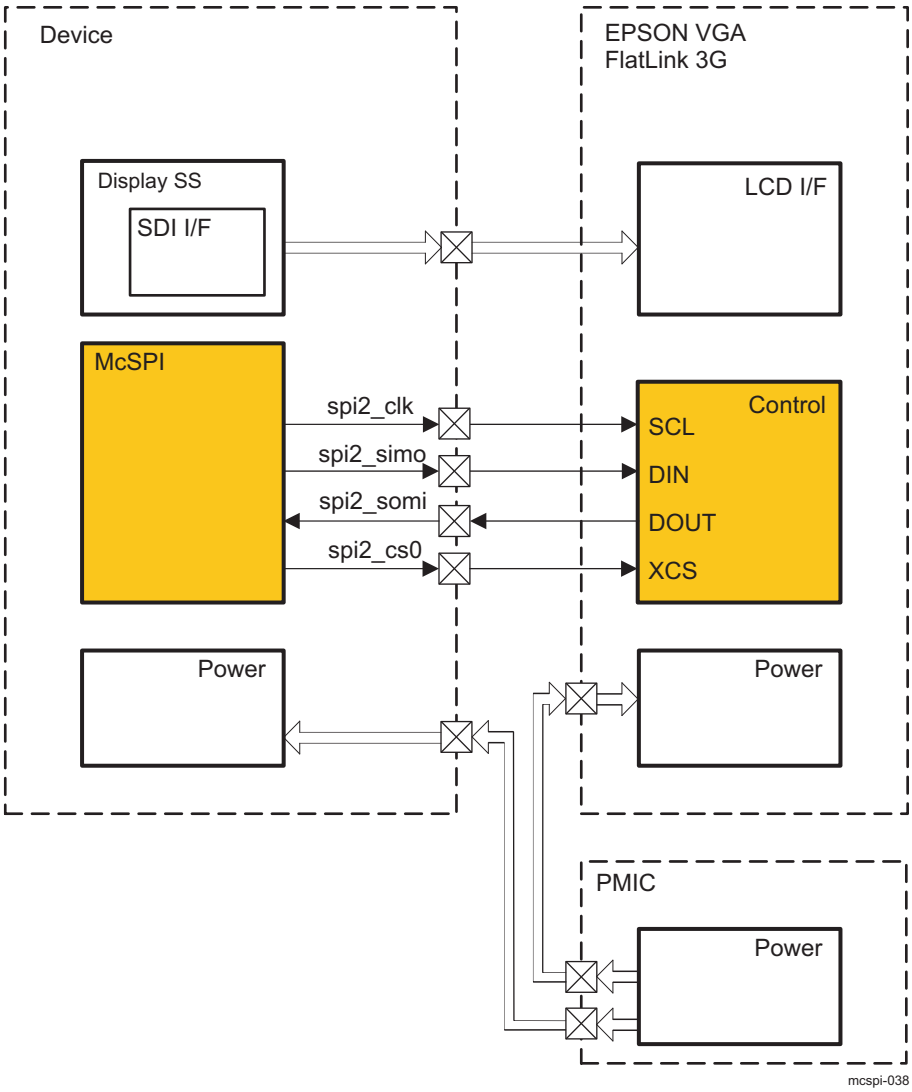
- McSPI2 channel 0 in master mode, EPSON VGA display in slave mode
- Transmit-only and receive-only modes
- No DMA requests and no interrupts used: transmit and receive flag polling

16.7.1.2 Environment

In this use case, the EPSON VGA is connected to the device through channel 0 of the McSPI instance 2 as shown in Figure 16-44 (through mcspi2_somi, mcspi2_simo, mcspi2_cs0).

The display clocks are provided to the EPSON VGA by the device display subsystem. The display configuration clock comes from the McSPI interface (mcspi2_clk).

Figure 16-44. Environment

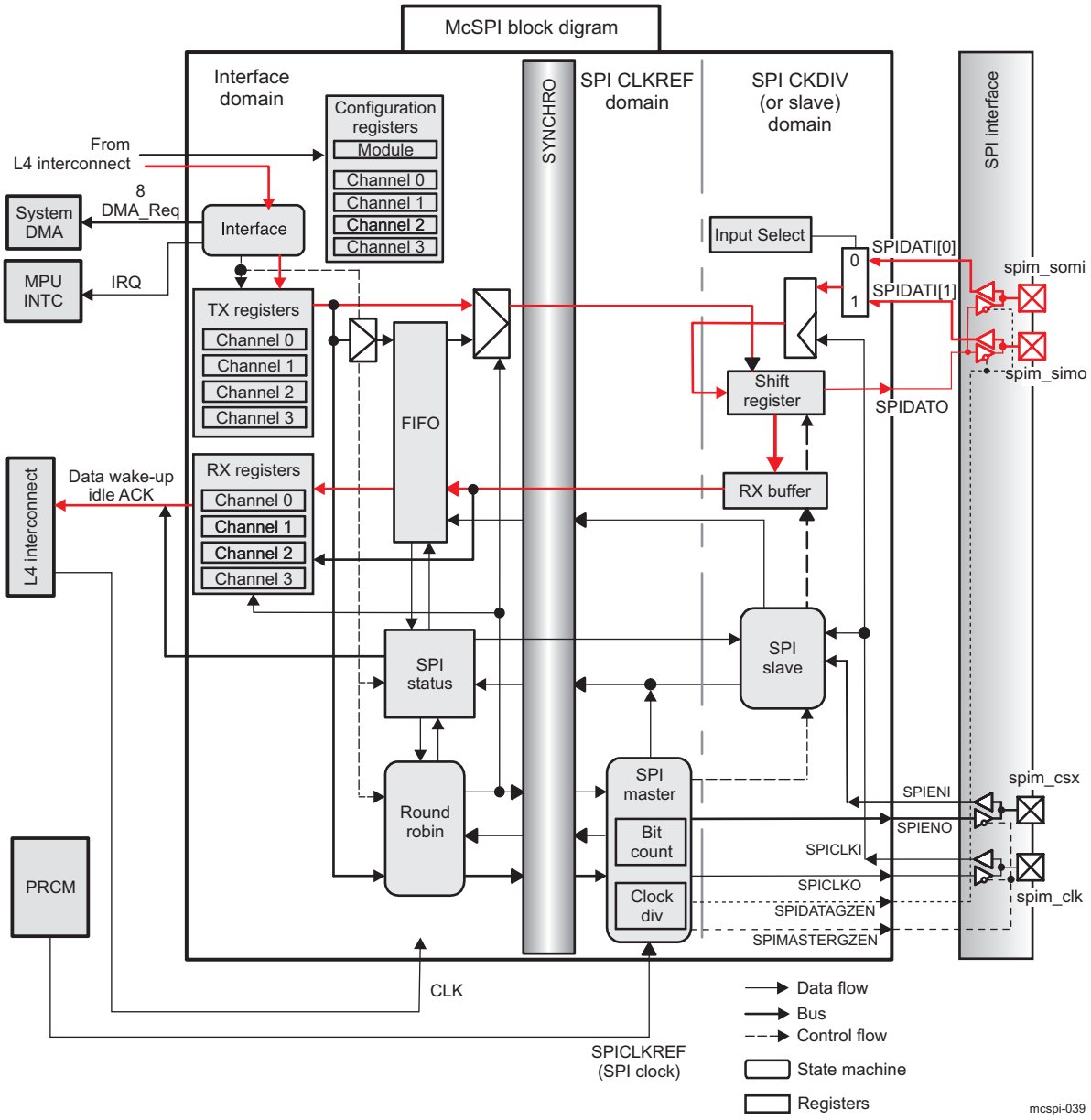


16.7.1.3 Data Path

In this use case, the McSPI2 manages the control path between the MPU subsystem and the EPSON VGA FlatLink™ 3G.

- Display configuration commands are transmitted via the channel 0 transmit register (TX Registers) and the shift register.
- A display status is read via the shift register, the receive buffer, and the channel 0 receive register (RX Registers).

Figure 16-45. McSPI Data Flow



16.7.1.4 Programming Flow

The initialization of the EPSON VGA is done in five steps:

1. McSPI module configuration
2. Send a 'SOFT RESET' command to the EPSON VGA
3. Send a 'SLEEP OUT' command
4. Send a 'DISPLAY ON' command
5. Send a 'READ DISPLAY STATUS' command and receive status data.

The four EPSON VGA commands needed in the EPSON VGA configuration are described in Table 16-18.

Table 16-18. EPSON VGA Configuration Commands

	D/CX	D7	D6	D5	D4	D3	D2	D1	D0	Dummy bit
SOFT RESET (001h)	0	0	0	0	0	0	0	0	1	n/a
SLEEP OUT (011h)	0	0	0	0	1	0	0	0	1	n/a
DISPLAY ON (029h)	0	0	0	1	0	1	0	0	1	n/a
READ DISPLAY STATUS (009h and dummy bit '0')	0	0	0	0	0	1	0	0	1	0

The following subsections describe each of these initialization steps.

16.7.1.4.1 McSPI Module Configuration

Before configuring the external display VGA, the multi-channel serial port interface must be initialized as detailed in the following steps:

1. Enable the McSPI2 functional and interface clocks through the PRCM.CM_FCLKEN1_CORE[18] EN_MCSPI2 and PRCM.CM_ICLKEN1_CORE[18] EN_MCSPI2 fields.
2. Proceed to a McSPI software reset through the MCSPI_SYSCONFIG[1] SOFTRESET field. The SOFTRESET bit is automatically reset by hardware.
3. Poll the MCSPI_SYSSTATUS[0] RESETDONE field to check the software reset status.
4. Disable channel 0 through the MCSPI_CHxCTRL[0] EN field (with x=0).
5. Disable all interrupts (MCSPI_IRQENABLE register) then clear interrupt status register (MCSPI_IRQSTATUS). Interrupts are disabled once at this stage. Even if some interrupts of the MCSPI_IRQSTATUS register are set, no interrupt can go out to the MPU subsystem.
6. Set the module in Master mode and multichannel mode through the MCSPI_MODULCTRL register.
7. Configure channel 0 in transmit-only mode, with a word length of 9-bit through the MCSPI_CHxCONF register (with x=0). A world length of 9 bits is required to transmit the required command (SOFT RESET, SLEEP OUT, or DISPLAY ON) to the EPSON display, as detailed in Table 16-18. Those 9 bits are serially transmitted on spi2_simo in the sequence D/CX, D7 to D0.

Table 16-19 shows all registers to be configured with the required value for the McSPI module configuration and the use case values.

Table 16-19. McSPI Configuration Registers Print

Register Name (with x=0)	Physical Address	Value	Value Description
PRCM.CM_FCLKEN1_CORE[18]	0x4800 4A00	0x1	Enable McSPI2 functional clock
PRCM.CM_ICLKEN1_CORE[18]	0x4800 4A10	0x1	Enable McSPI2 interface clock
MCSPI_SYSCONFIG[1]	0x4809 A010	0x1	Initiate a software reset
MCSPI_SYSSTATUS	0x4809 A014	0x0000 0001	The RESETDONE field is set at 1 when the reset is complete
MCSPI_CHxCTRL	0x4809 A034	0x0000 0000	Disable channel 0
MCSPI_IRQENABLE	0x4809 A01C	0x0000 0000	Disable all interrupts
MCSPI_IRQSTATUS	0x4809 A018	0x0001 777F	Clear all interrupts
MCSPI_MODULCTRL	0x4809 A028	0x0000 0000	Master mode, multichannel

Table 16-19. McSPI Configuration Registers Print (continued)

MCSPI_CHxCONF	0x4809 A02C	0x0001 2453	CLKD = 4 (CLKSPIREF divided internally by 16) WL = 9-bit Transmit-only Receive on spi2_somi Transmit on spi2_simo

16.7.1.4.2 'SOFT RESET', 'SLEEP OUT' and 'DISPLAY ON' Commands

Once the McSPI interface is configured, three commands must be send to the external display VGA: a software reset to reinitiate the external display, a 'SLEEP OUT' command to wake it up, and a 'DISPLAY ON' to turn on the display. Those three commands follow the same procedure, described here below, only the command opcode changes.

1. Enable channel 0 through the MCSPI_CHxCTRL[0] EN field (with x=0).
2. Write the command opcode to the MCSPI_TXx (with x=0) register so that it could be transmitted to the external display VGA. To transmit the SOFT RESET command, write 001h in the MCSPI_TXx (with x=0) register. To transmit a SLEEP OUT command write 011h and to transmit a DISPLAY ON command write 029h.
3. Poll the MCSPI_CHxSTAT[1] TXS field (with x=0) to check for transmit status. When the word has been transmitted, TXS is automatically set to 1.
4. Disable channel 0 through the MCSPI_CHxCTRL[0] EN field (with x=0).

Table 16-20 shows all registers to be configured with the required value for the those commands and the use case values.

Table 16-20. Display Configuration Registers Print

Register Name (with x=0)	Physical Address	Value	Value Description
MCSPI_CHxCTRL	0x4809 A034	0x0000 0001	Enable channel 0
MCSPI_TXx	0x4809 A038	0x001 0x011 0x029	Transmit a 'SOFT RESET' command to the EPSON VGA display Transmit a 'SLEEP OUT' command Transmit a 'DISPLAY ON' command
MCSPI_CHxSTAT[1]	0x4809 A030	0x1	The word in MCSPI_TXx has been transmitted when the TXS flag is set to 1
MCSPI_CHxCTRL	0x4809 A034	0x0000 0000	Disable channel 0

16.7.1.4.3 'READ DISPLAY STATUS' Command

Once the EPSON VGA display is initialized, a 'READ DISPLAY STATUS' command is sent to the external display VGA to read and check the status of the component, such as Partial mode, Sleep In mode, Display status, Horizontal and Vertical Synchronization On or Off states.

The 'READ DISPLAY STATUS' command is not 9- but 10-bit long because the EPSON VGA display requires a dummy cycle between the transmission of the last command bit (D0 in Table 16-18) and the reception of the first data bit (the display status data is 32-bit long). In other words, a dummy bit is added as less significant bit of the command. Hence, those 10 bits are serially transmitted on spi2_simo in the sequence D/CX, D7 to D0, dummy bit.

1. Set the module in Master mode and single channel mode through the MCSPI_MODULCTRL register.
2. Configure channel 0 in transmit-only mode, with a word length of 10-bit through the MCSPI_CHxCONF register (with x=0).
3. Enable channel 0 through the MCSPI_CHxCTRL[0] EN field.

4. Shift left of one bit the command opcode 009h so that the required dummy bit ('0') can be sent after the D/CX and D7 to D0 bits to the EPSON VGA display. Refer to Table 16-18 for more information. Then, write this command (012h) to the MCSPI_TXx (with x=0) register so that it could be transmitted to the EPSON VGA display.
5. Poll the MCSPI_CHxSTAT[1] TXS bit (with x=0) to check transmit status. When the word has been transmitted, TXS is automatically set to 1.
6. Read the MCSPI_CHxSTAT[1] RXS bit to check receive status. If data has already been received (RXS is set to 1) read the MCSPI_RXx (with x=0) receive register; this will automatically deactivate the RXS flag (RXS is automatically cleared).
7. Configure channel 0 in receive-only mode, with a word length of 32-bit through the MCSPI_CHxCONF register.
8. Poll the MCSPI_CHxSTAT[1] RXS bit to check receive status. When data has been received, RXS is automatically set to 1. The 32-bit display status information is available in the McSPI2 channel 0 receive register: MCSPI_RXx (with x=0).
9. Set channel 0 in transmit-only mode through the MCSPI_CHxCONF register.

Table 16-21 shows all registers to be configured with the required value for the McSPI module configuration and the use case values.

Table 16-21. Display Status Check Registers Print

Register Name (with x=0)	Physical Address	Value	Value Description
MCSPI_MODULCTRL	0x4809 A028	0x0000 0001	Master mode, single channel
MCSPI_CHxCONF	0x4809 A02C	0x0011 24D3	CLKD = 4 (CLKSPIREF divided internally by 16) WL = 10-bit Transmit-only Receive on spi2_somi Transmit on spi2_simo
MCSPI_CHxCTRL	0x4809 A034	0x0000 0001	Enable channel 0
MCSPI_TXx	0x4809 A038	0x0000 0012	Send a 'READ DISPLAY STATUS' command. See Table 16-18.
MCSPI_CHxSTAT[1] TXS	0x4809 A030	0x1	The word in MCSPI_TXx has been transmitted if the TXS flag is set to 1
MCSPI_CHxSTAT[0] RXS	0x4809 A030	-	Read RXS flag. Data has been received in MCSPI_RXx if the RXS flag is set to 1
MCSPI_RXx	0x4809 A03C	-	Read MCSPI_RXx only if the RXS bit is equal to 1 (this is a dummy read that will automatically clear the RXS flag)
MCSPI_CHxCONF	0x4809 A02C	0x0011 1FD0	CLKD = 4 (CLKSPIREF divided internally by 16) WL = 32-bit Receive-only Receive on spi2_somi Transmit on spi2_simo
MCSPI_CHxSTAT[0] RXS	0x4809 A030	0x0	Data has been received in MCSPI_RXx when the RXS flag is set to 1.
MCSPI_CHxCONF	0x4809 A02C	0x0011 2453	Back in transmit-only mode Receive on spi2_somi Transmit on spi2_simo

16.8 McSPI Register Manual

16.8.1 McSPI Instance Summary

Table 16-22 lists the McSPI instances.

Table 16-22. McSPI Instance Summary

Module Name	Base Address	Size
MCSPI1	0x4809 8000	4Kbytes
MCSPI2	0x4809 A000	4Kbytes
MCSPI3	0x480B 8000	4Kbytes
MCSPI4	0x480B A000	4Kbytes

16.8.2 McSPI Register Summary

Table 16-23 lists the McSPI registers. Each register has a 32-bit width. Table 16-24 through Table 16-50 describe the register bits.

Table 16-23. McSPI Register Summary

Register	Type	Offset Address	MCSPI1 Instance Physical Address	MCSPI2 Instance Physical Address	MCSPI3 Instance Physical Address	MCSPI4 Instance Physical Address
MCSPI_REVISION	R	0x00	0x4809 8000	0x4809 A000	0x480B 8000	0x480B A000
MCSPI_SYSCONFIG	RW	0x10	0x4809 8010	0x4809 A010	0x480B 8010	0x480B A010
MCSPI_SYSSTATUS	R	0x14	0x4809 8014	0x4809 A014	0x480B 8014	0x480B A014
MCSPI_IRQSTATUS	RW	0x18	0x4809 8018	0x4809 A018	0x480B 8018	0x480B A018
MCSPI_IRQENABLE	RW	0x1C	0x4809 801C	0x4809 A01C	0x480B 801C	0x480B A01C
MCSPI_WAKEUPENABLE	RW	0x20	0x4809 8020	0x4809 A020	0x480B 8020	0x480B A020
MCSPI_SYST	RW	0x24	0x4809 8024	0x4809 A024	0x480B 8024	0x480B A024
MCSPI_MODULCTRL	RW	0x28	0x4809 8028	0x4809 A028	0x480B 8028	0x480B A028
MCSPI_CHxCONF (1)	RW	0x2C + (0x14 * x)	0x4809 802C + (0x14 * x)	0x4809 A02C + (0x14 * x)	0x480B 802C + (0x14 * x)	0x480B A02C + (0x14 * x)
MCSPI_CHxSTAT (1)	R	0x30 + (0x14 * x)	0x4809 8030 + (0x14 * x)	0x4809 A030 + (0x14 * x)	0x480B 8030 + (0x14 * x)	0x480B A030 + (0x14 * x)
MCSPI_CHxCTRL (1)	RW	0x34 + (0x14 * x)	0x4809 8034 + (0x14 * x)	0x4809 A034 + (0x14 * x)	0x480B 8034 + (0x14 * x)	0x480B A034 + (0x14 * x)
MCSPI_TXx (1)	RW	0x38 + (0x14 * x)	0x4809 8038 + (0x14 * x)	0x4809 A038 + (0x14 * x)	0x480B 8038 + (0x14 * x)	0x480B A038 + (0x14 * x)
MCSPI_RXx (1)	R	0x3C + (0x14 * x)	0x4809 803C + (0x14 * x)	0x4809 A03C + (0x14 * x)	0x480B 803C + (0x14 * x)	0x480B A03C + (0x14 * x)
MCSPI_XFERLEVEL	RW	0x7C	0x4809 807C	0x4809 A07C	0x480B 807C	0x480B A07C

(1) x= 0 to 3 for MCSPI1.
x= 0 to 1 for MCSPI2 and MCSPI3.
x= 0 for MCSPI4.

16.8.3 McSPI Register Description

Table 16-24. MCSPI_REVISION

Address Offset	0x00		
Physical Address	0x4809 8000	**Instance**	MCSPI1
	0x4809 A000		MCSPI2
	0x480B 8000		MCSPI3
	0x480B A000		MCSPI4
Description	This register contains the hard coded RTL revision number.		
Type	R		
Write Latency	Not relevant		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REV							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads returns 0	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Example: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 16-25. Register Call Summary for Register MCSPI_REVISION

McSPI Register Manual
- McSPI Register Summary: [0]

Table 16-26. MCSPI_SYSCONFIG

Address Offset	0x10		
Physical Address	0x4809 8010	**Instance**	MCSPI1
	0x4809 A010		MCSPI2
	0x480B 8010		MCSPI3
	0x480B A010		MCSPI4
Description	This register allows control of various parameters of the module interface. It is not sensitive to software reset.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CLOCKACTIVITY		Reserved			SIDLEMODE		ENAWAKEUP		SOFTRESET		AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:10	Reserved	Reads returns 0	RW	0x000000
9:8	CLOCKACTIVITY	Clocks activity during wake up mode period	RW	0x0
		0x0: Interface and Functional clocks may be switched off.		
		0x1: Interface clock is maintained. Functional clock may be switched off.		
		0x2: Functional clock is maintained. Interface clock may be switched off.		
		0x3: Interface and Functional clocks are maintained.		
7:5	Reserved	Reads returns 0	RW	0x0
4:3	SIDLEMODE	Power management	RW	0x0
		0x0: If an idle request is detected, the McSPI acknowledges it unconditionally and goes in Inactive mode. Interrupt, DMA requests and wake up lines are unconditionally de-asserted and the module wake-up capability is deactivated even if the bit MCSPI_SYSCONFIG[ENAWAKEUP] is set.		
		0x1: If an idle request is detected, the request is ignored and the module does not switch to wake up mode and keeps on behaving normally.		
		0x2: If an idle request is detected, the module will switch to wake up mode based on its internal activity and the wake up capability can be used if the bit MCSPI_SYSCONFIG[ENAWAKEUP] is set.		
		0x3: Reserved - do not use.		
2	ENAWAKEUP	Wake-up feature control	RW	0
		0x0: Wake-up capability disabled		
		0x1: Wake-up capability enabled		
1	SOFTRESET	Software reset. Read always returns 0.	RW	0
		0x0: Normal mode.		
		0x1: Trigger a module reset. This bit is automatically reset by hardware.		
0	AUTOIDLE	Internal interface Clock gating strategy	RW	0
		0x0: interface clock is free-running		
		0x1: Automatic interface clock gating strategy is applied, based on the module interface activity		

Table 16-27. Register Call Summary for Register MCSPI_SYSCONFIG

McSPI Integration • Software Reset: [0] [1]
McSPI Functional Description • Normal Mode: [2] • Idle Mode: [3] [4] [5] [6]
McSPI Basic Programming Model • McSPI Configuration and Operations Example: [7]
McSPI Use Cases and Tips • Programming Flow: [8] [9]
McSPI Register Manual • McSPI Register Summary: [10] • McSPI Register Description: [11] [12]

Table 16-28. MCSPI_SYSSTATUS

Address Offset	0x14		
Physical Address	0x4809 8014	**Instance**	MCSPI1
	0x4809 A014		MCSPI2
	0x480B 8014		MCSPI3
	0x480B A014		MCSPI4
Description	This register provides status information about the module excluding the interrupt status information		
Type	R		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description		Type	Reset
31:1	Reserved	Reserved for module specific status information. Read returns 0		R	0x00000000
0	RESETDONE	Internal Reset Monitoring		R	0
		0x0:	Internal module reset is on-going		
		0x1:	Reset completed		

Table 16-29. Register Call Summary for Register MCSPI_SYSSTATUS

McSPI Basic Programming Model
- Initialization of Modules: [0]
- McSPI Configuration and Operations Example: [1]

McSPI Use Cases and Tips
- Programming Flow: [2] [3]

McSPI Register Manual
- McSPI Register Summary: [4]

Table 16-30. MCSPI_IRQSTATUS

Address Offset	0x18		
Physical Address	0x4809 8018	**Instance**	MCSPI1
	0x4809 A018		MCSPI2
	0x480B 8018		MCSPI3
	0x480B A018		MCSPI4
Description	The interrupt status regroups all the status of the module internal events that can generate an interrupt		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														EOW	WKS	Reserved	RX3_FULL	TX3_UNDERFLOW	TX3_EMPTY	Reserved	RX2_FULL	TX2_UNDERFLOW	TX2_EMPTY	Reserved	RX1_FULL	TX1_UNDERFLOW	TX1_EMPTY	RX0_OVERFLOW	RX0_FULL	TX0_UNDERFLOW	TX0_EMPTY

Bits	Field Name	Description	Type	Reset
31:18	Reserved	Reads returns 0	RW	0x0000
17	EOW	End of word count event when a channel is enabled using the FIFO buffer and the channel had sent the number of SPI word defined by MCSPI_XFERLEVEL[31:16] WCNT bit field. Write 0x0: Event status bit unchanged. Read 0x0: Event false. Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
16	WKS	Wake-up event in slave mode when an active control signal is detected on the spim_csx line programmed in the MCSPI_CHxCONF[SPIENSLV] field (where x=0 only) Write 0x0: Event status bit unchanged Read 0x0: Event false Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
15	Reserved	Reads returns 0.	RW	0x0
14	RX3_FULL	MCSPI_RX3 register is full (only when channel 3 is enabled) Write 0x0: Event status bit unchanged Read 0x0: Event false Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
13	TX3_UNDERFLOW	MCSPI_TX3 register underflow (only when channel 3 is enabled)(1) Write 0x0: Event status bit unchanged Read 0x0: Event false Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
12	TX3_EMPTY	MCSPI_TX3 register is empty (only when channel 3 is enabled)(2) Write 0x0: Event status bit unchanged Read 0x0: Event false Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
11	Reserved	Read returns 0.	RW	0x0
10	RX2_FULL	MCSPI_RX2 register full (only when channel 2 is enabled) Write 0x0: Event status bit unchanged Read 0x0: Event false Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
9	TX2_UNDERFLOW	MCSPI_TX2 register underflow (only when channel 2 is enabled)(1) Write 0x0: Event status bit unchanged Read 0x0: Event false Write 0x1: Event status bit is reset. Read 0x1: Event is pending.	RW	0x0
8	TX2_EMPTY	MCSPI_TX2 register empty (only when channel 2 is enabled)(2) Write 0x0: Event status bit unchanged Read 0x0: Event false	RW	0x0

(1) The MCSPI_TXx register is empty (not updated by host or DMA with new data) before its time slot assignment.
Exception: No TX_UNDERFLOW event when no data has been loaded into the MCSPI_TXx register since channel has been enabled.

(2) Enabling the channel automatically rises this event.

Bits	Field Name	Description	Type	Reset
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
7	Reserved	Read returns 0.	RW	0x0
6	RX1_FULL	MCSPI_RX1 register full (only when channel 1 is enabled)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
5	TX1_UNDERFLOW	MCSPI_TX1 register underflow (only when channel 1 is enabled)(1)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
4	TX1_EMPTY	MCSPI_TX1 register empty (only when channel 1 is enabled)(3)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
3	RX0_OVERFLOW	MCSPI_RX0 register overflow (only in slave mode)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
2	RX0_FULL	MCSPI_RX0 register full (only when channel 0 is enabled)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
1	TX0_UNDERFLOW	MCSPI_TX0 register underflow (only when channel 0 is enabled)(4)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		
0	TX0_EMPTY	MCSPI_TX0 register empty (only when channel 0 is enabled)(3)	RW	0x0
		Write 0x0: Event status bit unchanged		
		Read 0x0: Event false		
		Write 0x1: Event status bit is reset.		
		Read 0x1: Event is pending.		

(3) Enabling the channel automatically rises this event.

(4) The MCSPI_TXx register is empty (not updated by host or DMA with new data) before its time slot assignment.
Exception: No TX_UNDERFLOW event when no data has been loaded into the MCSPI_TXx register since channel has been enabled.

Table 16-31. Register Call Summary for Register MCSPI_IRQSTATUS

McSPI Functional Description
- Master Transmit-and-Receive Mode (Full Duplex): [0]
- Master Transmit-Only Mode (Half Duplex): [1]
- Master Receive-Only Mode (Half Duplex): [2] [3] [4] [5]
- Single-Channel Master Mode: [6] [7]
- Buffer Almost Full: [8] [9]
- Buffer Almost Empty: [10] [11]
- End of Transfer Management: [12]
- Interrupts: [13]
- Interrupt Events in Master Mode: [14] [15] [16] [17] [18] [19]
- Interrupt Events in Slave Mode: [20] [21] [22] [23] [24] [25] [26]
- Interrupt-Driven Operation: [27] [28]
- Polling: [29] [30]
- Idle Mode: [31] [32] [33]

McSPI Basic Programming Model
- McSPI Configuration and Operations Example: [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47]

McSPI Use Cases and Tips
- Programming Flow: [48] [49] [50]

McSPI Register Manual
- McSPI Register Summary: [51]
- McSPI Register Description: [52]

Table 16-32. MCSPI_IRQENABLE

Address Offset	0x1C		
Physical Address	0x4809 801C	**Instance**	MCSPI1
	0x4809 A01C		MCSPI2
	0x480B 801C		MCSPI3
	0x480B A01C		MCSPI4
Description	This register allows to enable/disable the module internal sources of interrupt, on an event-by-event basis.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														EOWKE	WKE	Reserved	RX3_FULL_ ENABLE	TX3_UNDERFLOW_ ENABLE	TX3_EMPTY_ ENABLE	Reserved	RX2_FULL_ ENABLE	TX2_UNDERFLOW_ ENABLE	TX2_EMPTY_ ENABLE	Reserved	RX1_FULL_ ENABLE	TX1_UNDERFLOW_ ENABLE	TX1_EMPTY_ ENABLE	RX0_OVERFLOW_ ENABLE	RX0_FULL_ ENABLE	TX0_UNDERFLOW_ ENABLE	TX0_EMPTY_ ENABLE

Bits	Field Name	Description	Type	Reset
31:18	Reserved	Reads return 0	RW	0x0000
17	EOWKE	End of Word count Interrupt Enable. 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
16	WKE	Wake-up event interrupt enable in slave mode when an active control signal is detected on the spim_csx line programmed in the MCSPI_CHxCONF[SPIENSLV] field (where x=0 only)	RW	0x0

Bits	Field Name	Description	Type	Reset
		0x0: Interrupt disabled 0x1: Interrupt enabled		
15	Reserved	Read returns 0.	RW	0x0
14	RX3_FULL_ ENABLE	MCSPI_RX3 register full interrupt enable (channel 3) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
13	TX3_UNDERFLOW_ ENABLE	MCSPI_TX3 register underflow interrupt enable (channel 3) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
12	TX3_EMPTY_ ENABLE	MCSPI_TX3 register empty interrupt enable (channel 3) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
11	Reserved	Read returns 0.	RW	0x0
10	RX2_FULL_ ENABLE	MCSPI_RX2 register full interrupt enable (channel 2) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
9	TX2_UNDERFLOW_ ENABLE	MCSPI_TX2 register underflow interrupt enable (channel 2) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
8	TX2_EMPTY_ ENABLE	MCSPI_TX2 register empty interrupt enable (channel 2) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
7	Reserved	Read returns 0.	RW	0x0
6	RX1_FULL_ ENABLE	MCSPI_RX1 register full interrupt enable (channel 1) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
5	TX1_UNDERFLOW_ ENABLE	MCSPI_TX1 register underflow interrupt enable (channel 1) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
4	TX1_EMPTY_ ENABLE	MCSPI_TX1 register empty interrupt enable (channel 1) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
3	RX0_OVERFLOW_ ENABLE	MCSPI_RX0 register overflow interrupt enable (channel 0) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
2	RX0_FULL_ ENABLE	MCSPI_RX0 register full interrupt enable (channel 0) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
1	TX0_UNDERFLOW_ ENABLE	MCSPI_TX0 register underflow interrupt enable (channel 0) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0
0	TX0_EMPTY_ ENABLE	MCSPI_TX0 register empty interrupt enable (channel 0) 0x0: Interrupt disabled 0x1: Interrupt enabled	RW	0x0

Table 16-33. Register Call Summary for Register MCSPI_IRQENABLE

McSPI Functional Description • Interrupts: [0] • Interrupt-Driven Operation: [1] • Polling: [2] • Idle Mode: [3] [4]
McSPI Basic Programming Model • McSPI Configuration and Operations Example: [5]
McSPI Use Cases and Tips • Programming Flow: [6] [7]
McSPI Register Manual • McSPI Register Summary: [8]

Table 16-34. MCSPI_WAKEUPENABLE

Address Offset	0x20		
Physical Address	0x4809 8020	**Instance**	MCSPI1
	0x4809 A020		MCSPI2
	0x480B 8020		MCSPI3
	0x480B A020		MCSPI4
Description	The wake-up enable register allows to enable/disable the module internal sources of wake-up on event-by-event basis.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															WKEN
R-0																															RW-0

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reads return 0	RW	0x00000000
0	WKEN	Wake-up functionality in slave mode when an active control signal is detected on the spim_cs line programmed in the field MCSPI_CHxCONF[SPIENSLV] 0x0: The event is not allowed to wake-up the system, even if the global control bit MCSPI_SYSCONF[ENAWAKEUP] is set. 0x1: The event is allowed to wake-up the system if the global control bit MCSPI_SYSCONF[ENAWAKEUP] is set.	RW	0

Table 16-35. Register Call Summary for Register MCSPI_WAKEUPENABLE

McSPI Functional Description • Idle Mode: [0] [1] [2]
McSPI Register Manual • McSPI Register Summary: [3]

Table 16-36. MCSPI_SYST

Address Offset	0x24		
Physical Address	0x4809 8024	**Instance**	MCSPI1
	0x4809 A024		MCSPI2
	0x480B 8024		MCSPI3
	0x480B A024		MCSPI4
Description	This register is used to check the correctness of the system interconnect either internally to peripheral bus, or externally to device IO pads, when the module is configured in system test (SYSTEST) mode.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SSB	SPIENDIR	SPIDATDIR1	SPIDATDIR0	WAKD	SPICLK	SPIDAT_1	SPIDAT_0	SPIEN_3	PIEN_2	SPIEN_1	SPIEN_0

Bits	Field Name	Description	Type	Reset
31:12	Reserved	Reads returns 0	RW	0x00000
11	SSB	Set status bit 0x0: No action. Writing 0 does not clear already set status bits; This bit must be cleared prior attempting to clear a status bit of the MCSPI_ IRQSTATUS register. 0x1: Force to 1 all status bits of MCSPI_ IRQSTATUS register. Writing 1 into this bit sets to 1 all status bits contained in the MCSPI_IRQSTATUS register.	RW	0
10	SPIENDIR	Set the direction of the spim_cs lines and spim_clk line 0x0: Output (as in master mode) 0x1: Input (as in slave mode)	RW	0
9	SPIDATDIR1	Set the direction of the SPIDAT[1] (spim_simo) 0x0: Output 0x1: Input	RW	0
8	SPIDATDIR0	Set the direction of the SPIDAT[0] (spim_somi) 0x0: Output 0x1: Input	RW	0
7	WAKD	SWAKEUP output (signal data value of internal signal to system). The signal is driven high or low according to the value written into this register bit. 0x0: The pin is driven low. 0x1: The pin is driven high.	RW	0
6	SPICLK	spim_clk line (signal data value) If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the spim_clk line (high or low) and a write into this bit has no effect. If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the spim_clk line is driven high or low according to the value written into this register.	RW	0
5	SPIDAT_1	spim_somi line (signal data value) If MCSPI_SYST[SPIDATDIR1] = 0 (output mode direction), the spim_somi line is driven high or low according to the value written into this register.	RW	0

Bits	Field Name	Description	Type	Reset
		If MCSPI_SYST[SPIDATDIR1] = 1 (input mode direction), this bit returns the value on the spim_somi line (high or low) and a write into this bit has no effect.		
4	SPIDAT_0	spim_simo line (signal data value) If MCSPI_SYST[SPIDATDIR0] = 0 (output mode direction), the spim_simo line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIDATDIR0] = 1 (input mode direction), this bit returns the value on the spim_simo line (high or low) and a write into this bit has no effect.	RW	0
3	SPIEN_3	spim_cs3 line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the spim_cs3 line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the spim_cs3 line (high or low) and a write into this bit has no effect.	RW	0
2	SPIEN_2	spim_cs2 line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the spim_cs2 line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the spim_cs2 line (high or low) and a write into this bit has no effect.	RW	0
1	SPIEN_1	spim_cs1 line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the spim_cs1 line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the spim_cs1 line (high or low) and a write into this bit has no effect.	RW	0
0	SPIEN_0	spim_cs0 line (signal data value) If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the spim_cs0 line is driven high or low according to the value written into this register. If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the spim_cs0 line (high or low) and a write into this bit has no effect.	RW	0

Table 16-37. Register Call Summary for Register MCSPI_SYST

McSPI Register Manual

- McSPI Register Summary: [0]
- McSPI Register Description: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14]

Table 16-38. MCSPI_MODULCTRL

Address Offset	0x28		
Physical Address	0x4809 8028	**Instance**	MCSPI1
	0x4809 A028		MCSPI2
	0x480B 8028		MCSPI3
	0x480B A028		MCSPI4
Description	This register is dedicated to the configuration of the serial port interface.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																												SYSTEM_TEST	MS	Reserved	SINGLE

Bits	Field Name	Description	Type	Reset
31:4	Reserved	Reads returns 0	RW	0x0000000
3	SYSTEM_TEST	Enables the system test mode	RW	0
		0x0: Functional mode		
		0x1: System test mode (SYSTEST)		
2	MS	Master/ Slave	RW	1
		0x0: Master - The module generates the spim_clk and spim_cs for channel x		
		0x1: Slave - The module receive		
1	Reserved	(returns 0 after writing 0) (returns 1 after writing 1)	RW	0
0	SINGLE	Single forced channel/multichannel (master mode only)	RW	0
		0x0: One or more channels will be used in master mode with automatic chip-select generation.		
		0x1: Only one channel will be used in master mode and the chip-select is driven by the MCSPI_CHxCONF[20] FORCE bit. This bit must be set in force spim_cs mode.		

Table 16-39. Register Call Summary for Register MCSPI_MODULCTRL

McSPI Functional Description • Single-Channel Master Mode: [0] [1] [2] [3] • Chip-Select Timing Control: [4] • Slave Mode: [5]
McSPI Basic Programming Model • McSPI Configuration and Operations Example: [6] [7]
McSPI Use Cases and Tips • Programming Flow: [8] [9] [10] [11]
McSPI Register Manual • McSPI Register Summary: [12] • McSPI Register Description: [13]

Table 16-40. MCSPI_CHxCONF

Address Offset	0x2C + (0x14 * x)	**Index**	x= 0 to 3 for MCSPI1. x= 0 to 1 for MCSPI2 and MCSPI3. x= 0 for MCSPI4.
Physical Address	0x4809 802C + (0x14 * x)	**Instance**	MCSPI1
	0x4809 A02C + (0x14 * x)		MCSPI2
	0x480B 802C + (0x14 * x)		MCSPI3
	0x480B A02C + (0x14 * x)		MCSPI4
Description	This register is dedicated to the configuration of the channel x.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CLKG	FFER	FFEW	TCS		SBPOL	SBE	Reserved		FORCE	TURBO	IS	DPE1	DPE0	DMAR	DMAW	TRM		WL				EPOL	CLKD					POL	PHA

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Read returns 0s.	RW	0x00
29	CLKG	Clock divider granularity. This register defines the granularity of channel clock divider: power of two or one clock cycle granularity. When this bit is set, the MCSPI_CHxCTRL[15:8] EXTCLK bit field must be configured to reach a maximum of 4096 clock divider ratio. Then the clock divider ratio is a concatenation of MCSPI_CHxCONF[5:2] CLKD and EXTCLK values. 0x0: Clock granularity of power of two 0x1: One clock cycle ganularity	RW	0x0
28	FFER	FIFO enabled for Receive. Only one channel can have this bit field set. 0x0: The buffer is not used to Receive data 0x1: The buffer is used to Receive data	RW	0x0
27	FFEW	FIFO enabled for Transmit. Only one channel can have this bit field set. 0x0: The buffer is not used to Transmit data 0x1: The buffer is used to Transmit data	RW	0x0
26:25	TCS	Chip select time control Defines the number of interface clock cycles between CS toggling and first (or last) edge of SPI clock. 0x0: 0.5 clock cycle 0x1: 1.5 clock cycles 0x2: 2.5 clock cycles 0x3: 3.5 clock cycles	RW	0x0
24	SBPOL	Start bit polarity 0x0: Start bit polarity is held to 0 during SPI transfer. 0x1: Start bit polarity is held to 1 during SPI transfer.	RW	0x0
23	SBE	Start bit enable for SPI transfer 0x0: Default SPI transfer length as specified by WL bit field 0x1: Start bit D/CX added before SPI transfer. Polarity is defined by MCSPI_CHxCONF[SBPOL].	RW	0x0
22:21	Reserved	Write the reset value. Read returns the reset value.	RW	0x0
20	FORCE	Manual spim_csx assertion to keep spim_csx for channel x active between SPI words (single channel master mode only). The MCSPI_MODULCTRL[0] SINGLE bit must bit set to 1. 0x0: Writing 0 into this bit drives the spin_csx line low for channel x when MCSPI_CHxCONF [6] EPOL = 0, and drives it high when MCSPI_CHxCONF[6] EPOL = 1. 0x1: Writing 1 into this bit drives the spim_csx line high for channel x when MCSPI_CHxCONF[E6] EPOL = 0, and drives it low when MCSPI_CHxCONF[6] EPOL = 1.	RW	0x0
19	TURBO	Turbo mode 0x0: Turbo is deactivated (recommended for single SPI word transfer) 0x1: Turbo is activated to maximize the throughput for multi-SPI word transfers.	RW	0x0
18	IS	Input select 0x0: Data Line 0 (spim_somi) selected for reception 0x1: Data Line 1 (spim_simo) selected for reception	RW	0x1

Bits	Field Name	Description	Type	Reset
17	DPE1	Transmission enable for data line 1 (spim_simo) 0x0: Data Line 1 (spim_simo) selected for transmission 0x1: No transmission on data Line 1 (spim_simo)	RW	0x1
16	DPE0	Transmission enable for data line 0 (spim_somi) 0x0: Data Line 0 (spim_somi) selected for transmission 0x1: No transmission on data Line 0 (spim_somi)	RW	0x0
15	DMAR	DMA Read request The DMA Read request line is asserted when the channel is enabled and a new data is available in the receive register of the channel. The DMA Read request line is deasserted on read completion of the receive register of the channel. 0x0: DMA read request disabled 0x1: DMA read request enabled	RW	0x0
14	DMAW	DMA Write request. The DMA write request line is asserted when the channel is enabled and the MCSPI_TXx register of the channel is empty. The DMA write request line is deasserted on load completion of the MCSPI_TXx register of the channel. 0x0: DMA write request disabled 0x1: DMA write request enabled	RW	0x0
13:12	TRM	Transmit/receive modes 0x0: Transmit and receive mode 0x1: Receive-only mode 0x2: Transmit-only mode 0x3: Reserved	RW	0x0
11:7	WL	SPI word length 0x0: Reserved 0x1: Reserved 0x2: Reserved 0x3: The SPI word is 4-bit long 0x4: The SPI word is 5-bit long 0x5: The SPI word is 6-bit long 0x6: The SPI word is 7-bit long 0x7: The SPI word is 8-bit long 0x8: The SPI word is 9-bit long 0x9: The SPI word is 10-bit long 0xA: The SPI word is 11-bit long 0xB: The SPI word is 12-bit long 0xC: The SPI word is 13-bit long 0xD: The SPI word is 14-bit long 0xE: The SPI word is 15-bit long 0xF: The SPI word is 16-bit long 0x10: The SPI word is 17-bit long 0x11: The SPI word is 18-bit long 0x12: The SPI word is 19-bit long 0x13: The SPI word is 20-bit long 0x14: The SPI word is 21-bit long 0x15: The SPI word is 22-bit long 0x16: The SPI word is 23-bit long 0x17: The SPI word is 24-bit long	RW	0x00

Bits	Field Name	Description	Type	Reset
		0x18: The SPI word is 25-bit long		
		0x19: The SPI word is 26-bit long		
		0x1A: The SPI word is 27-bit long		
		0x1B: The SPI word is 28-bit long		
		0x1C: The SPI word is 29-bit long		
		0x1D: The SPI word is 30-bit long		
		0x1E: The SPI word is 31-bit long		
		0x1F: The SPI word is 32-bit long		
6	EPOL	spim_csx polarity for channel x	RW	0x0
		0x0: spim_csx is held high during the active state.		
		0x1: spim_csx is held low during the active state.		
5:2	CLKD	Frequency divider for spim_clk (for master device only) A programmable clock divider divides the SPI reference clock (CLKSPIREF) by a 4-bit value and results in a new spim_clk clock available to shift data in and out.	RW	0x0
		0x0: 1		
		0x1: 2		
		0x2: 4		
		0x3: 8		
		0x4: 16		
		0x5: 32		
		0x6: 64		
		0x7: 128		
		0x8: 256		
		0x9: 512		
		0xA: 1024		
		0xB: 2048		
		0xC: 4096		
		0xD: 8192		
		0xE: 16384		
		0xF: 32768		
1	POL	spim_clk polarity	RW	0x0
		0x0: spim_clk is held high during the active state.		
		0x1: spim_clk is held low during the active state.		
0	PHA	spim_clk phase	RW	0x0
		0x0: Data are latched on odd-numbered edges of spim_clk.		
		0x1: Data are latched on even-numbered edges of spim_clk.		

Table 16-41. Register Call Summary for Register MCSPI_CHxCONF

McSPI Functional Interface

- Multichannel SPI Protocol and Data Format: [0] [1] [2] [3] [4] [5] [6] [7]
- Transfer Format: [8]

Table 16-41. Register Call Summary for Register MCSPI_CHxCONF (continued)

McSPI Functional Description

- Master Transmit-and-Receive Mode (Full Duplex): [9]
- Master Transmit-Only Mode (Half Duplex): [10]
- Master Receive-Only Mode (Half Duplex): [11]
- Single-Channel Master Mode: [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
- Start Bit Mode: [24] [25] [26] [27]
- Chip-Select Timing Control: [28]
- Programmable SPI Clock (spim_clk): [29] [30] [31] [32] [33] [34]
- Dedicated Resources: [35] [36] [37] [38] [39]
- Slave Transmit-and-Receive Mode: [40]
- Slave Transmit-Only Mode: [41]
- Slave Receive-Only Mode: [42]
- FIFO Buffer Management: [43] [44] [45]
- Buffer Almost Full: [46] [47]
- Buffer Almost Empty: [48] [49]
- Interrupt Events in Master Mode: [50] [51]
- Interrupt Events in Slave Mode: [52] [53]
- DMA Requests: [54] [55]
- Idle Mode: [56]

McSPI Basic Programming Model

- Transfer Procedures without FIFO: [57]
- McSPI Configuration and Operations Example: [58] [59] [60] [61] [62] [63] [64] [65] [66] [67]
- Transfer Procedures with FIFO: [68]

McSPI Use Cases and Tips

- Programming Flow: [69] [70] [71] [72] [73] [74] [75] [76]

McSPI Register Manual

- McSPI Register Summary: [77]
- McSPI Register Description: [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89]

Table 16-42. MCSPI_CHxSTAT

Address Offset	0x30 + (0x14 * x)	**Index**	x= 0 to 3 for MCSPI1. x= 0 to 1 for MCSPI2 and MCSPI3. x= 0 for MCSPI4.
Physical Address	0x4809 8030 + (0x14 * x)	**Instance**	MCSPI1
	0x4809 A030 + (0x14 * x)		MCSPI2
	0x480B 8030 + (0x14 * x)		MCSPI3
	0x480B A030 + (0x14 * x)		MCSPI4
Description	This register provides status information about MCSPI_TXx and MCSPI_RXx registers of channel x.		
Type	R		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								RXFFF	RXFFE	TXFFF	TXFFE	EOT	TXS	RXS

Bits	Field Name	Description	Type	Reset
31:7	Reserved	Read returns 0s.	R	0x00000000
6	RXFFF	Channel x FIFO Receive Buffer Full Status	R	0x0
		Read 0x0: FIFO Receive Buffer is not full		
		Read 0x1: FIFO Receive Buffer is full		
5	RXFFE	Channel x FIFO Receive Buffer Empty Status	R	0x0

Bits	Field Name	Description	Type	Reset
		Read 0x0: FIFO Receive Buffer is not empty		
		Read 0x1: FIFO Receive Buffer is empty		
4	TXFFF	Channel x FIFO Transmit Buffer Full Status	R	0x0
		Read 0x0: FIFO Transmit Buffer is not full		
		Read 0x1: FIFO Transmit Buffer is full		
3	TXFFE	Channel x FIFO Transmit Buffer Empty Status	R	0x0
		Read 0x0: FIFO Transmit Buffer is not empty		
		Read 0x1: FIFO Transmit Buffer is empty		
2	EOT	Channel x end-of-transfer status. The definitions of beginning and end of transfer vary with master versus slave and the transfer format (transmit/receive mode, turbo mode). See dedicated chapters for details.	R	0x0
		Read 0x0: This flag is automatically cleared when the shift register is loaded with the data from the MCSPI_TXx register (beginning of transfer).		
		Read 0x1: This flag is automatically set to one at the end of an SPI transfer.		
1	TXS	Channel x MCSPI_TXx register status	R	0x0
		Read 0x0: Register is full.		
		Read 0x1: Register is empty.		
0	RXS	Channel x MCSPI_RXx register status	R	0x0
		Read 0x0: Register is empty.		
		Read 0x1: Register is full.		

Table 16-43. Register Call Summary for Register MCSPI_CHxSTAT

McSPI Functional Description
- Master Transmit-and-Receive Mode (Full Duplex): [0] [1] [2]
- Master Transmit-Only Mode (Half Duplex): [3]
- Master Receive-Only Mode (Half Duplex): [4]
- Single-Channel Master Mode: [5] [6] [7] [8] [9]
- Dedicated Resources: [10] [11]
- Slave Transmit-and-Receive Mode: [12]
- Slave Transmit-Only Mode: [13]
- Slave Receive-Only Mode: [14]
- End of Transfer Management: [15]

McSPI Use Cases and Tips
- Programming Flow: [16] [17] [18] [19] [20] [21] [22] [23]

McSPI Register Manual
- McSPI Register Summary: [24]

Table 16-44. MCSPI_CHxCTRL

Address Offset	0x34 + (0x14 * x)	**Index**	x= 0 to 3 for MCSPI1. x= 0 to 1 for MCSPI2 and MCSPI3. x= 0 for MCSPI4.
Physical Address	0x4809 8034 + (0x14 * x)	**Instance**	MCSPI1
	0x4809 A034 + (0x14 * x)		MCSPI2
	0x480B 8034 + (0x14 * x)		MCSPI3
	0x480B A034 + (0x14 * x)		MCSPI4
Description	This register is dedicated to enable the channel x		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																EXTCLK								Reserved							EN

Bits	Field Name	Description		Type	Reset
31:16	Reserved	Read returns 0s.		RW	0x0000
15:8	EXTCLK	Clock ratio extension: This register is used to concatenate with MCSPI_CHxCONF[5:2] CLKD bit field for clock ratio only when granularity is one clock cycle (MCSPI_CHxCONF[28] CLKG bit set to 1). Then the max value reached is 4096 clock divider ratio.		RW	0x00
		0x0:	Clock ratio is CLKD + 1		
		0x1:	Clock ratio is CLKD + 1 + 16		
		0xFF:	Clock ratio is CLKD + 1 + 4080		
7:1	Reserved	Read returns 0s.		RW	0x00
0	EN	Channel enable		RW	0x0
		0x0:	Channel x is not active.		
		0x1:	Channel x is active.		

Table 16-45. Register Call Summary for Register MCSPI_CHxCTRL

McSPI Functional Description
- Master Transmit-and-Receive Mode (Full Duplex): [0]
- Single-Channel Master Mode: [1] [2]
- Programmable SPI Clock (spim_clk): [3]
- Dedicated Resources: [4] [5]

McSPI Basic Programming Model
- McSPI Configuration and Operations Example: [6] [7]

McSPI Use Cases and Tips
- Programming Flow: [8] [9] [10] [11] [12] [13] [14] [15]

McSPI Register Manual
- McSPI Register Summary: [16]
- McSPI Register Description: [17]

Table 16-46. MCSPI_TXx

Address Offset	0x38 + (0x14 * x)	**Index**	x= 0 to 3 for MCSPI1. x= 0 to 1 for MCSPI2 and MCSPI3. x= 0 for MCSPI4.
Physical Address	0x4809 8038 + (0x14 * x)	**Instance**	MCSPI1
	0x4809 A038 + (0x14 * x)		MCSPI2
	0x480B 8038 + (0x14 * x)		MCSPI3
	0x480B A038 + (0x14 * x)		MCSPI4
Description	This register contains a single SPI word to transmit on the serial link, whatever SPI word length is.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDATA																															

Bits	Field Name	Description	Type	Reset
31:0	TDATA	Channel 0 Data to transmit	RW	0x00000000

Table 16-47. Register Call Summary for Register MCSPI_TXx

McSPI Functional Description

- Master Transmit-and-Receive Mode (Full Duplex): [0] [1] [2] [3]
- Master Transmit-Only Mode (Half Duplex): [4]
- Master Receive-Only Mode (Half Duplex): [5] [6] [7] [8] [9]
- Single-Channel Master Mode: [10]
- Dedicated Resources: [11] [12] [13] [14] [15]
- Slave Transmit-and-Receive Mode: [16] [17]
- Slave Receive-Only Mode: [18] [19] [20]
- Interrupt Events in Master Mode: [21] [22] [23] [24] [25] [26] [27]
- Interrupt Events in Slave Mode: [28] [29] [30] [31] [32] [33]
- Interrupt-Driven Operation: [34]
- Polling: [35]
- DMA Requests: [36] [37]

McSPI Basic Programming Model

- McSPI Configuration and Operations Example: [38]
- Common Transfer Procedure: [39] [40]

McSPI Use Cases and Tips

- Programming Flow: [41] [42] [43] [44] [45] [46] [47]

McSPI Register Manual

- McSPI Register Summary: [48]
- McSPI Register Description: [49] [50] [51] [52] [53]

Table 16-48. MCSPI_RXx

Address Offset	0x3C + (0x14 * x)	**Index**	x= 0 to 3 for MCSPI1. x= 0 to 1 for MCSPI2 and MCSPI3. x= 0 for MCSPI4.
Physical Address	0x4809 803C + (0x14 * x)	**Instance**	MCSPI1
	0x4809 A03C + (0x14 * x)		MCSPI2
	0x480B 803C + (0x14 * x)		MCSPI3
	0x480B A03C + (0x14 * x)		MCSPI4
Description	This register contains a single SPI word received through the serial link, what ever SPI word length is.		
Type	R		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA																															

Bits	Field Name	Description	Type	Reset
31:0	RDATA	Channel 0 Received Data	R	0x00000000

Table 16-49. Register Call Summary for Register MCSPI_RXx

McSPI Functional Description
- Master Transmit-and-Receive Mode (Full Duplex): [0] [1]
- Master Transmit-Only Mode (Half Duplex): [2] [3] [4]
- Master Receive-Only Mode (Half Duplex): [5] [6]
- Single-Channel Master Mode: [7] [8] [9] [10]
- Dedicated Resources: [11] [12] [13]
- Slave Transmit-and-Receive Mode: [14]
- Slave Transmit-Only Mode: [15] [16]
- End of Transfer Management: [17]
- Interrupt Events in Master Mode: [18] [19] [20] [21]
- Interrupt Events in Slave Mode: [22] [23] [24] [25] [26] [27]
- Interrupt-Driven Operation: [28]
- Polling: [29]
- DMA Requests: [30]
- Idle Mode: [31]

McSPI Basic Programming Model
- McSPI Configuration and Operations Example: [32]

McSPI Use Cases and Tips
- Programming Flow: [33] [34] [35] [36] [37] [38]

McSPI Register Manual
- McSPI Register Summary: [39]
- McSPI Register Description: [40] [41]

Table 16-50. MCSPI_XFERLEVEL

Address Offset	0x7C		
Physical Address	0x4809 807C	**Instance**	MCSPI1
	0x4809 A07C		MCSPI2
	0x480B 807C		MCSPI3
	0x480B A07C		MCSPI4
Description	This register provides transfer levels needed while using FIFO buffer during transfer.		
Type	RW		
Write Latency	Immediate		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WCNT																Reserved		AFL						Reserved		AEL					

Bits	Field Name	Description	Type	Reset
31:16	WCNT	Spi word counter: This register holds the programmable value of number of SPI word to be transferred on channel which is using the FIFO buffer. When transfer had started, a read back in this register returns the current SPI word transfer index. 0x0: Counter not used 0x1: One spi word 0xFFFE : 65534 spi word 0xFFFF : 65535 spi word	RW	0x0000
15:14	Reserved	Read returns 0s.	RW	0x0

Bits	Field Name	Description	Type	Reset
13:8	AFL	Buffer Almost Full: This register holds the programmable almost full level value used to determine almost full buffer condition. If the user wants an interrupt or a DMA read request to be issued during a receive operation when the data buffer holds at least I bytes, then this bit field must be set to I-1. 0x0: One byte 0x1: 2 bytes 0x3E: 63 bytes 0x3F: 64 bytes	RW	0x00
7:6	Reserved	Read returns 0s.	RW	0x0
5:0	AEL	Buffer Almost Empty: This register holds the programmable almost empty level value used to determine almost empty buffer condition. If the user wants an interrupt or a DMA write request to be issued during a transmit operation when the data buffer is able to receive I bytes, then this bit field must be set to I-1. 0x0: One byte 0x1: 2 bytes 0x3E: 63 bytes 0x3F: 64 bytes	RW	0x00

HDQ/1-Wire

This chapter describes the features and functions of the HDQ/1-Wire module.

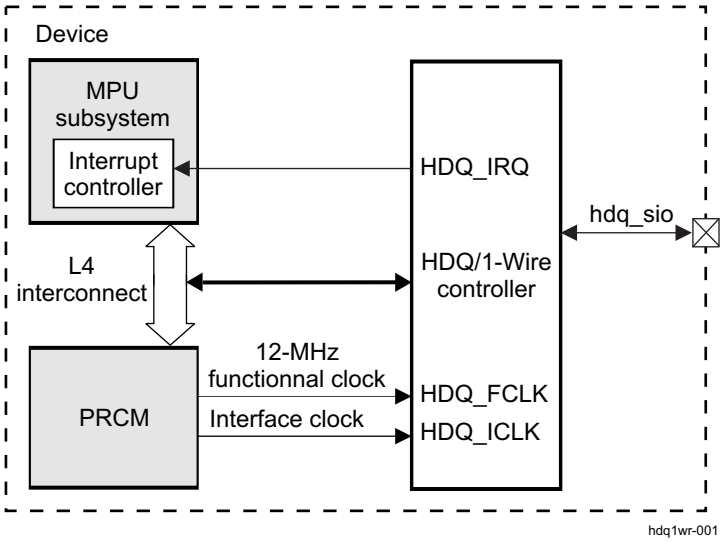
Topic **Page**

17.1 HDQ/1-Wire Overview

The HDQ/1-Wire module implements the hardware protocol of the master functions of the Benchmarq HDQ and the Dallas Semiconductor 1-Wire® protocols. These protocols use a single wire for communication between the master (HDQ/1-Wire controller) and the slave (HDQ/1-Wire external compliant device).

Figure 17-1 shows the HDQ/1-Wire controller module.

Figure 17-1. HDQ/1-Wire Highlight



The HDQ and 1-Wire module has a generic L4 interface and is intended to be used in an interrupt-driven fashion. The one-pin interface is implemented as an open-drain output at the device level.

The HDQ operates from a fixed 12-MHz functional clock provided by the PRCM module.

Only the MPU subsystem uses the HDQ/1-Wire module.

The main features of the HDQ/1-Wire module support the following:

- Benchmarq HDQ protocol
- Dallas Semiconductor 1-Wire protocol
- Power-down mode

The HDQ/1-Wire module provides a communication rate of 5K bits/s over an address space of 128 bytes.

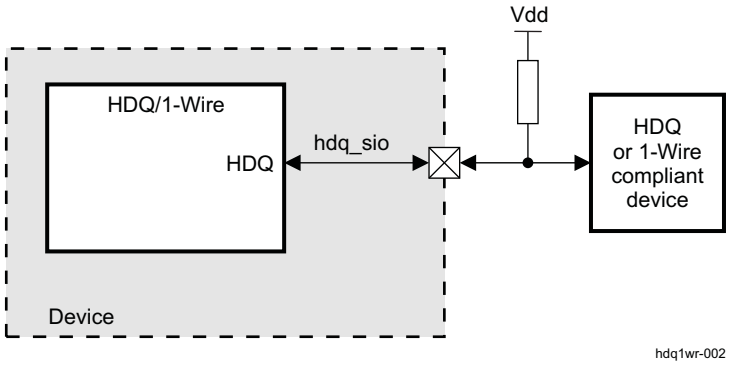
A typical application of the HDQ/1-Wire module is the communication with battery monitor (gas gauge) integrated circuits.

17.2 HDQ/1-Wire Environment

17.2.1 HDQ/1-Wire Functional Interface

Figure 17-2 shows a typical application using the HDQ/1-Wire connection.

Figure 17-2. HDQ/1-Wire Typical Application System Overview



An external pullup is required, because the two protocols use a return-to-1 mechanism (that is, after any command, the line is pulled to a logical high level).

The HDQ/1-Wire module operates according to a command structure that is programmed into transmit command registers (as described in Section 17.5).

The 1-Wire mode runs at slower speeds than the capabilities of the mode.

Table 17-1 describes the external pullup signal from an HDQ or 1-Wire compliant device.

Table 17-1. I/O Description

Signal Name	I/O	Description	Value at Reset
hdq_sio	Bidir	Serial data input/output	1

17.2.2 HDQ and 1-Wire (SDQ) Protocols

17.2.2.1 HDQ Protocol Initialization (Default)

In HDQ mode, the firmware does not require the host to create an initialization pulse to the slave. However, the slave can be reset by using an initialization pulse (also referred to as a break pulse). The initialization pulse is generated by setting the INITIALIZATION bit (HDQ.HDQ_CTRL_STATUS[2]). The slave does not respond with a presence pulse as it does in the 1-Wire protocol.

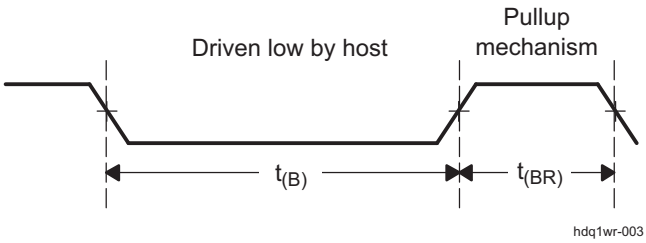
The HDQ is a command-based protocol in which the host sends a command byte to the slave. The command directs the slave either to store the next eight bits of data received to a register specified by the command byte (write operation) or to output the eight bits of data from a register specified by the command byte (read operation). The master implementation is a simple byte engine. The sending of the ID, command/address, and data is controlled by firmware. The master engine provides only a single HDQ.HDQ_TX_DATA register.

The command and data bytes consist of a stream of eight bits with a maximum transmission rate of 5K bits/s. The least significant bit (LSB) of a command or data byte is transmitted first. If a communication time-out occurs between the host and the slave (for example, if the host waits longer than the specified time for the slave to respond, or if this is the first access command), then the host must send an initialization pulse (BREAK pulse) before sending the command again.

The slave detects a break when the HDQ pin is driven to a logic-low state for a specified break time t(B) or greater. The HDQ pin then returns to its normal ready-high logic state for a specified break-recovery time t(BR). The slave is then ready for a command from the host processor. Figure 17-3 shows this behavior.

An interrupt condition indicates either a TX-complete, an RX-complete, or a time-out condition. Reading the interrupt status register clears all interrupt conditions. Only one interrupt signal is sent to the microcontroller, and only one overall mask bit can enable or disable the interrupt. The interrupt conditions cannot be individually masked.

Figure 17-3. HDQ Break-Pulse Timing Diagram

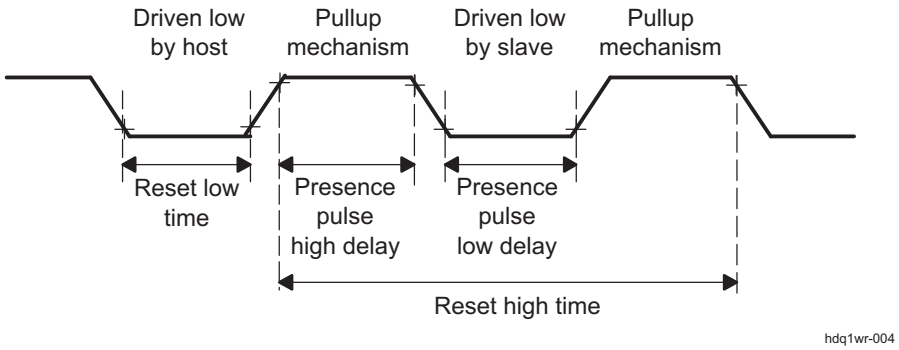


17.2.2.2 1-Wire (SDQ) Protocol Initialization

In 1-Wire (SDQ) protocol, the host first sends an initialization pulse (by pulling the line to a logic-low state) and then waits for the slave to respond with a presence pulse before enabling any communication sequence.

As for the initialization pulse, the presence pulse is a low-going edge on the line initiated by the slave. The timing diagram in Figure 17-4 shows the 1-Wire (SDQ) reset sequence.

Figure 17-4. 1-Wire (SDQ) Reset Timing Diagram



The host drives the line to a logic-low state for a minimum of reset low time. Once the slave detects this pulse, it must drive the line to a logic-low state within the presence pulse high delay for a minimum period of presence pulse low time.

If the slave does not respond within this interval of time, a time-out event occurs and no transaction can be initiated. The host must initiate the reset sequence again before sending any command to the slave.

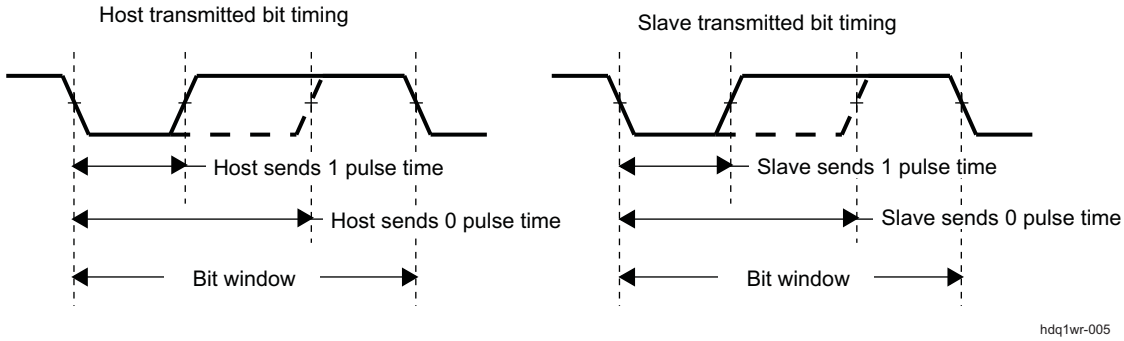
On the other hand, if the slave sends back its presence pulse within the specified interval of time, the communication can be enabled after the reset high time.

17.2.2.3 Communication Sequence (HDQ and 1-Wire Protocols)

This section describes the part common to both protocols.

After a successful break pulse (HDQ mode) or initialization sequence (1-Wire protocol), the host and slave are ready for bit transmission. Each bit to transmit (either from the host to the slave or from the slave to the host) is preceded by a low-going edge on the line, as shown in Figure 17-5.

Figure 17-5. HDQ/1-Wire Transmitted Bit Timing



The return-to-1 data-bit frame consists of three distinct sections. The first section starts the transmission when either the slave or the host takes the line to a logic-low state. The next section is the actual data transmission in which the data must be valid during a specified period of time after the negative edge that starts the communication. The final section stops the transmission by returning the HDQ/1-Wire line to a logic-high state. Communication with an HDQ/1-Wire slave always occurs with the LSB being transmitted first.

The command byte of the HDQ/1-Wire protocols consists of eight contiguous valid command bits. The command byte contains two fields: R/W command and address. The R/W bit of the command byte determines whether the command is a read or a write, and the address field containing bits AD6-AD0 indicates the address to be read or written. Table 17-2 lists the command byte values.

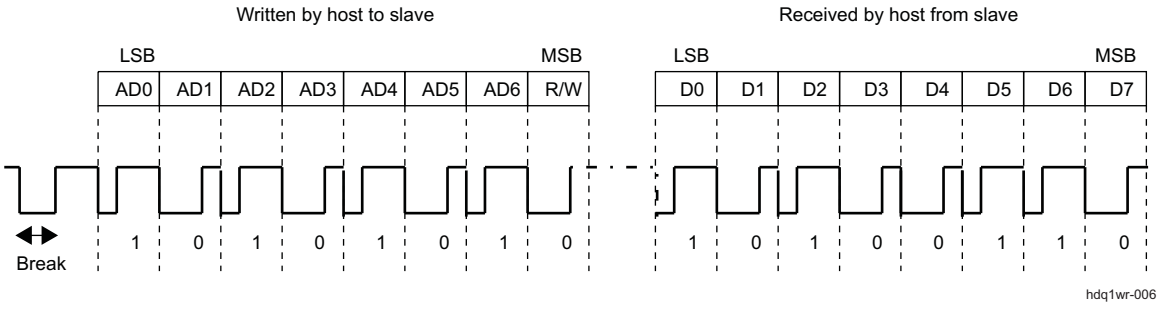
Table 17-2. HDQ/1-Wire Command Byte

7	6	5	4	3	2	1	0
R/W	AD6	AD5	AD4	AD3	AD2	AD1	AD0

R/W Indicates whether the command byte is a read or a write. A 1 indicates a write command; the following eight bits must be written to the register specified by the address field of the command byte. A 0 indicates that the command is a read. On a read command, the slave outputs the requested register contents.

AD6-AD0 Represent the seven bits labeled AD6-AD0 containing the address portion of the register to be accessed.
The communication sequence example in Figure 17-6 shows a read command at address 0x55; the received data is 0x65.

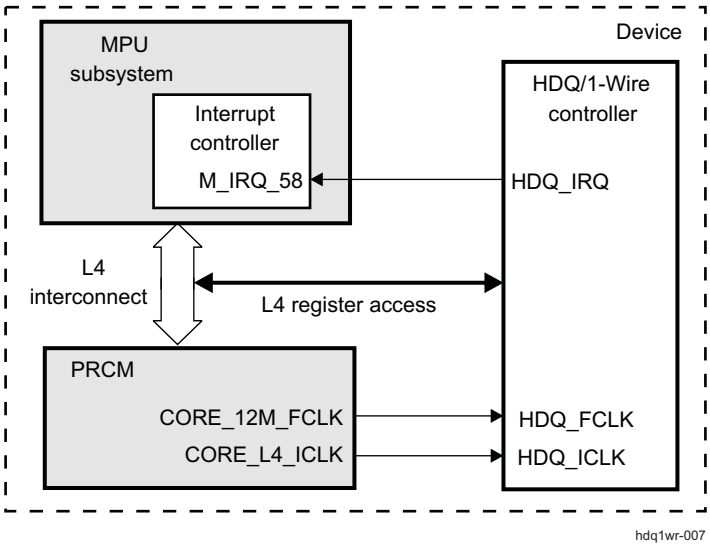
Figure 17-6. HDQ/1-Wire Communication Sequence



17.3 HDQ/1-Wire Integration

Figure 17-7 shows the HDQ/1-Wire module integration in the device.

Figure 17-7. HDQ/1-Wire Integration



17.3.1 Clocking, Reset, and Power Management Scheme

17.3.1.1 HDQ/1-Wire Clocks

There are two clock domains in the HDQ/1-Wire module: the functional clock domain and the interface clock domain. When these clocks are set in the PRCM module, the following rule must be observed: HDQ_ICLK ≥ HDQ_FCLK.

- HDQ_FCLK belongs to the functional clock domain. It is a fixed 12-MHz clock provided by the PRCM. It is used to clock the internal module logic.

  The HDQ_FCLK source is the CORE_12M_FCLK PRCM output, which is derived from the CORE_48M_FCLK clock signal (a 1/4 ratio is applied). The CORE_12M_FCLK clock is issued from the peripherals DPLL4. For more information about the clock, see , *Power, Reset, and Clock Management*.

  When the HDQ/1-Wire module no longer requires the HDQ_FCLK, the software can disable it at the PRCM level by setting the EN_HDQ bit (PRCM.CM_FCLKEN1_CORE[22]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it either.

  For details about the PRCM register settings and DPLL4 configuration, see , *Power, Reset, and Clock Management*.

- HDQ_ICLK is the interface clock. It runs at L4 interconnect clock speed and is used to trigger access to the HDQ/1-Wire L4 interface. Its source is the PRCM CORE_L4_ICLK signal. It typically runs at L3/2 frequency.

  When the HDQ/1-Wire module no longer requires the HDQ_ICLK, the software can disable it at the PRCM level by setting the EN_HDQ bit (PRCM.CM_ICLKEN1_CORE[22]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it either.

  It is also possible to activate an autoidle mode for this clock (AUTO_HDQ bit PRCM.CM_AUTOIDLE1_CORE[22] set to 1). In this case, HDQ_ICLK follows the CORE clock domain behavior. For more information, see , *Power, Reset, and Clock Management*.

17.3.1.2 HDQ/1-Wire Reset Scheme

Global reset of the module is done either by activating the CORE_RST signal in the CORE_RST domain (for more information, see , *Power, Reset, and Clock Management*) or by setting to 1 the SOFTRESET bit HDQ.HDQ_SYSCONFIG[1]. Setting this bit enables an active software reset functionality equivalent to a hardware reset.

17.3.1.3 HDQ/1-Wire Power Domain

The HDQ/1-Wire belongs to the CORE power domain. For more information about the CORE power domain, see , *Power, Reset, and Clock Management*.

17.3.2 Hardware Requests

The HDQ/1-Wire can generate one interrupt:

- HDQ_IRQ: This is an interrupt to the MPU subsystem interrupt controller. It is mapped on M_IRQ_58.

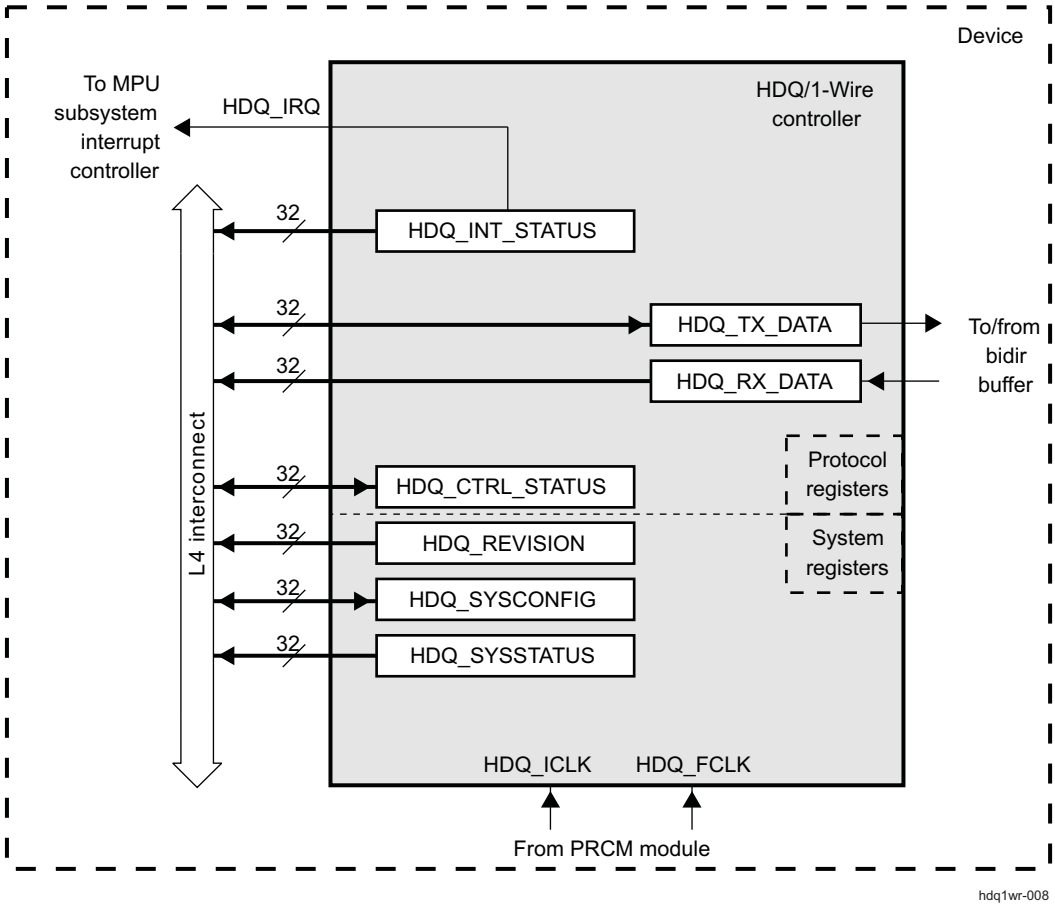
17.4 HDQ/1-Wire Functional Description

The HDQ/1-Wire module works with both the HDQ and 1-Wire protocols. The protocols use a single wire to establish communication between the master and the slave. Both protocols use a return-to-1 mechanism; that is, after any command is driven, the line is pulled to a high level. This mechanism requires an external pullup.

17.4.1 HDQ/1-Wire Block Diagram

Figure 17-8 shows the HDQ/1-Wire block diagram.

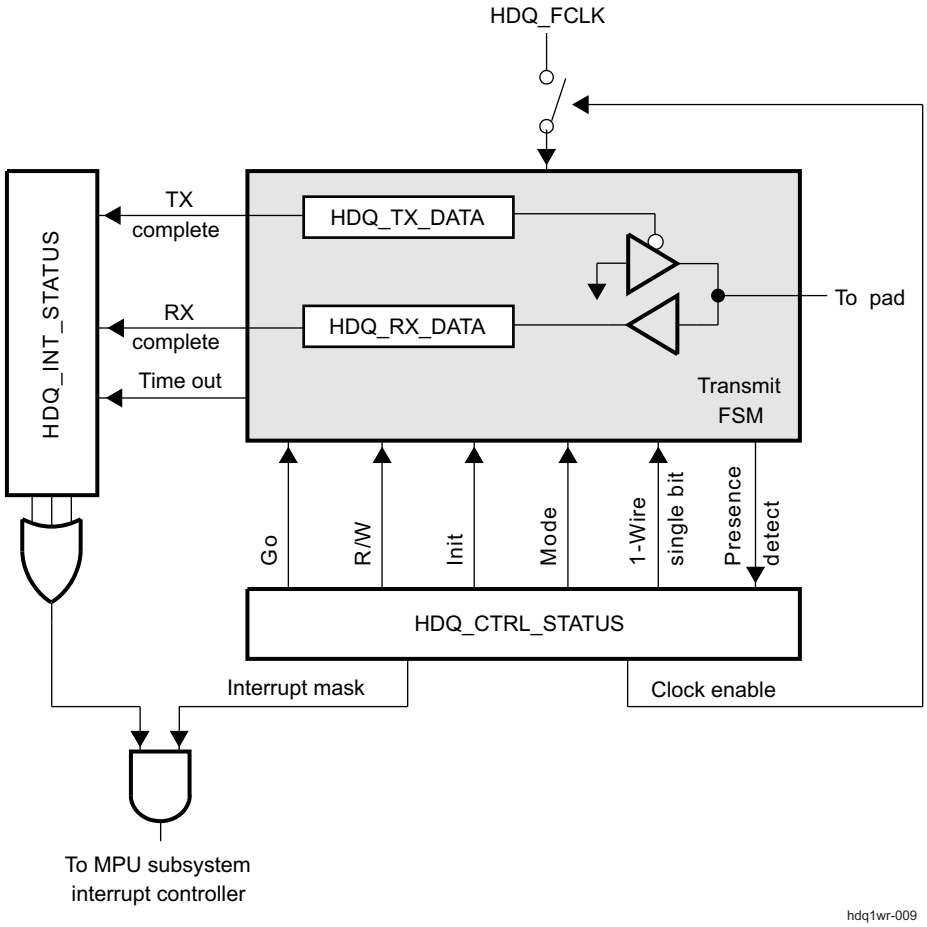
Figure 17-8. HDQ/1-Wire Block Diagram



The MODE bit HDQ.HDQ_CTRL_STATUS[0] allows selection between the HDQ and 1-Wire protocols. This bit is assumed static for design purposes. The configuration is in HDQ mode by default.

Figure 17-9 shows the protocol-dedicated register scheme.

Figure 17-9. Protocol Registers Description



The receive and transmit operations are performed with respect to the HDQ protocol timing. The module is implemented once in the OMAP device and is clocked with a single clock whether it runs HDQ or 1-Wire protocol. Although the data exchange is slower than the capabilities of the 1-Wire mode, this mode still meets the timing requirements and practical considerations. That is, the 1-Wire protocol runs at the HDQ protocol speed, which is slower (5K bits/s). The timing parameters and protocol are different in the two modes.

17.4.2 HDQ Mode (Default)

17.4.2.1 HDQ Mode Features

The HDQ mode supports the following:

- Benchmark HDQ protocol
- Power-down mode

17.4.2.2 Description

In the HDQ mode, there is no need for the host to create an initialization pulse to the slave. However, the host can reset the slave by using an initialization pulse (also known as a break pulse). Setting the INITIALIZATION bit HDQ.HDQ_CTRL_STATUS[2] creates this pulse by pulling the line down. When the slave receives the pulse, it is ready for communication but does not respond with a presence pulse.

In a typical write operation, two bytes are sent to the slave. The first byte corresponds to the command/address byte, and the second byte corresponds to the data to be written.

In a typical read operation, the host sends a command/address byte and the slave returns a byte of data.

The master is implemented to send and receive bytes. Sending the command/address and data is controlled by the firmware. The master provides only a single data TX register.

The HDQ protocol is a return-to-1 protocol. Consequently, after a byte is sent to the slave (either command/address + data for a write, or just command/address for a read), the host pulls the line up. The line is set to the high-impedance state in the device and an external pullup brings it to a logical high level.

In the case of a read operation, the slave also drives the line to a logic-low state before sending the requested data.

If the host initiates a read and does not receive data within a specified interval of time (that is, the slave does not drive the line low within this interval), the TIMEOUT bit HDQ.HDQ_INT_STATUS[0] is set, thereby indicating a read failure. The TIMEOUT bit remains set until the host reads the interrupt status register (HDQ.HDQ_INT_STATUS).

An interrupt condition indicates either a TX-complete, an RX-complete, or a time-out on a transaction. The corresponding bit is set in the interrupt status register (HDQ.HDQ_INT_STATUS). This register is cleared as soon as it is read.

Only one interrupt signal is sent to the MPU, and only an overall mask can enable or disable the interrupts. These interrupts cannot be individually masked.

17.4.2.3 Single-Bit Mode

In HDQ mode, the single-bit mode (1_WIRE_SINGLE_BIT bit HDQ.HDQ_CTRL_STATUS[7] set to 1) has no effect because the HDQ protocol supports only byte transfers.

17.4.2.4 Interrupt Conditions

The HDQ/1-Wire module provides the following interrupt status:

1. Transmission complete:

   A write operation of one byte was completed. Successful or failed completion is not indicated, because there is no acknowledgment from the slave in HDQ protocol. This interrupt condition is cleared by reading the interrupt status register (HDQ.HDQ_INT_STATUS).

2. Read complete:

   In HDQ mode, the interrupt status indicates that a byte has been successfully read. This interrupt condition is cleared by reading the interrupt status register (HDQ.HDQ_INT_STATUS).

3. Presence detect/time-out:

   In HDQ mode, the interrupt status indicates that after a read command initiated by the host, the slave did not pull the line low within the specified time. This interrupt condition is cleared by reading the interrupt status register (HDQ.HDQ_INT_STATUS).

   In HDQ mode, a time-out condition is also used to indicate the successful completion of a break pulse. That is, if the master has sent the break pulse, it is indicated with a time-out instead of a TX-complete.

Only one interrupt is generated to the MPU based on any of these interrupt conditions. A read operation on the interrupt status register clears all the interrupt status bits that were previously set.

17.4.3 1-Wire Mode

17.4.3.1 1-Wire Mode Features

The 1-Wire mode supports the following:

- Dallas Semiconductor 1-Wire protocol
- Power-down mode
- Single-bit mode

17.4.3.2 Description

The 1-Wire mode requires an initialization pulse to be sent to the slave(s) connected on the interface. If a slave is present, it responds with a presence pulse.

The initialization pulse is sent after INITIALIZATION bit HDQ.HDQ_CTRL_STATUS[2] is set. The firmware sends the initialization pulse depending on the value of this bit.

The slave presence on the line is detected by a presence bit in the control and status register. When the slave receives the initialization pulse, it sends back its presence pulse by pulling down the line. The module detects this low-going edge and sets the PRESENCEDETECT bit HDQ.HDQ_CTRL_STATUS[3].

In a similar way, if a presence pulse is not received from the slave after an initialization pulse is sent, the PRESENCEDETECT bit remains cleared.

Whether or not a presence pulse is detected after an initialization pulse is sent, the TIMEOUT bit HDQ.HDQ_INT_STATUS[0] is set and an interrupt condition is generated.

In 1-Wire mode, the generated interrupt condition means the maximum time allowed for receiving the response has elapsed and the software must check the PRESENCEDETECT bit to determine whether or not there was a presence pulse.

The INITIALIZATION bit is cleared at the end of the initialization pulse at the same time as the TIMEOUT bit is set. The TIMEOUT bit is cleared when the interrupt status register (HDQ.HDQ_INT_STATUS) is read.

For read operations, 1-Wire is a bit-by-bit protocol, which means the slave must be clocked by the host for each bit of the byte to read.

The line is pulled up at the end of the command/address byte. On the first read, the host creates a low-going edge to initiate a bit read. The line is then pulled up (pulled to the high-impedance state by the host and set to a high logical level by the external pullup) and the slave either drives the line low to transmit a 0 or does not drive the line to transmit a 1. This sequence is repeated for each bit to read.

The first bit the host receives is the LSB, and the last bit is the most significant bit (MSB) in the receive data register (HDQ.HDQ_RX_DATA).

An interrupt condition indicates either a TX-complete, an RX-complete, or a time-out condition (that is, the time allowed for the slave to indicate its presence has elapsed). A read operation on the interrupt status register clears the interrupt conditions previously set. As in the HDQ mode, only one interrupt signal is sent to the MPU. Only an overall mask bit can enable or disable the interrupt (the interrupt conditions cannot be masked individually).

17.4.3.3 1-Wire Single-Bit Mode Operation

A single-bit mode can be entered by setting the appropriate bit in the control and status register (1_WIRE_SINGLE_BIT bit HDQ.HDQ_CTRL_STATUS[7]). In this mode, only one bit of data at a time is transferred between the master and the slave. After the bit is transferred, an interrupt is generated (that is, there is an RX-complete for a read operation and a TX-complete for a write operation). Bit 0 of the RX register (HDQ.HDQ_RX_DATA) is updated each time a bit is received from the slave; bit 0 of the TX register (HDQ.HDQ_TX_DATA) contains the bit to be sent.

17.4.3.4 Interrupt Conditions

The HDQ/1-Wire module provides the following interrupt status:

1. Transmission complete:

   A write operation of one byte was completed. Successful or failed completion is not indicated, because there is no acknowledgment from the slave in 1-Wire protocol. This interrupt condition is cleared by reading the interrupt status register (HDQ.HDQ_INT_STATUS).

2. Read complete:

   In 1-Wire mode, the interrupt status indicates that a byte has been successfully read. This interrupt condition is cleared by reading the interrupt status register (HDQ.HDQ_INT_STATUS).

3. Presence detect/time-out:

   In 1-Wire mode, the interrupt status indicates that it is now valid to check the PRESENCEDETECT bit. This interrupt condition is cleared by reading the interrupt status register (HDQ.HDQ_INT_STATUS).

Only one interrupt is generated to the MPU based on any of these interrupt conditions. A read operation on the interrupt status register clears all interrupt status bits that were previously set.

17.4.3.5 Status Flags

The presence-condition-detected status flag is contained in the PRESENCEDETECT bit HDQ.HDQ_CTRL_STATUS[3]. This is valid only in 1-Wire mode. The flag is updated when TIMEOUT bit HDQ.HDQ_INT_STATUS[0] is set. Therefore, its correct value shows only after the interrupt is generated. The firmware must wait for the time-out condition; otherwise, the flag keeps its previous value and is undefined.

17.4.4 Module Power Saving

17.4.4.1 Autoidle Mode

The HDQ/1-Wire module provides an autoidle function in its interconnect clock domain.

The interconnect clock autoidle power-saving mode is enabled or disabled through the AUTOIDLE bit HDQ.HDQ_SYSCONFIG[0]. When this mode is enabled and there is no activity on the interconnect interface, the interconnect clock (HDQ_ICLK) is disabled inside the module, thereby reducing power consumption. When there is new activity on the interconnect interface, the interconnect clock is restarted with no latency penalty. This mode is disabled by default after a reset.

This mode is recommended to reduce power consumption.

17.4.4.2 Power-Down Mode

The HDQ/1-Wire module also provides a power-saving function in its functional clock domain.

Setting the CLOCKENABLE bit in the control and status register (CLOCKENABLE bit HDQ.HDQ_CTRL_STATUS[5]) to 0 shuts off the functional clock (HDQ_FCLK) to the state-machine. The state-machine is reset when the functional clock is disabled; if any transaction is ongoing, it is aborted into the reset state.

The register values are not affected by disabling the functional clock.

Do not access the module registers after the software has put the module in power-down mode except to write to the CLOCKENABLE bit to take the module out of power-down mode.

17.4.5 System Power Management and Wakeup

As part of the system-wide power-management scheme, the HDQ/1-Wire module can go into idle state at the request of the PRCM module (for more information, see , *Power, Reset, and Clock Management*). However, the HDQ/1-Wire module does not support handshake protocol with the PRCM. The HDQ/1-Wire module can go into idle mode only as part of the L4 interconnect clock domain (both CORE_L4_ICLK and FUNC_12M_CLK belong to the L4 interconnect clock domain).

When the AUTO_HDQ bit PRCM.CM_AUTOIDLE1_CORE[22] is set, the HDQ/1-Wire module behavior follows the L4 interconnect clock domain behavior. If the whole domain is put into idle, the HDQ/1-Wire is also put into idle.

Software must ensure correct clock management.

CAUTION

There is no hardware mechanism to prevent cutting off the HDQ/1-Wire clocks while the module is performing a transfer. The result would be a loss of data being transferred.

17.5 HDQ/1-Wire Basic Programming Model

The HDQ/1-Wire module can be considered a simple byte engine because it only implements the hardware interface layer for both HDQ and 1-Wire protocols. The correct sequencing is controlled by the firmware, which is described in this section.

17.5.1 Module Initialization Sequence

17.5.1.1 Mode Selection

MODE bit HDQ.HDQ_CTRL_STATUS[0] allows selection between the HDQ and 1-Wire protocols. When set to 0, the protocol is HDQ; when set to 1, the protocol is 1-Wire. The bit is assumed static for design purposes. The configuration is in HDQ mode by default.

Although this bit can be modified at any point, it is strongly recommended that it be modified only as part of the boot-up configuration.

17.5.1.2 Reset/Initialization

No slave presence test is required in HDQ mode; however, the slave can be reset by setting the INITIALIZATION bit HDQ.HDQ_CTRL_STATUS[2] and the GO bit HDQ.HDQ_CTRL_STATUS[4]. The line is then pulled down (break pulse) and the bit returns to 0 after the pulse is sent. Upon completion, a time-out interrupt is also generated. The slave does not respond to this pulse.

In 1-Wire mode, the slave returns a presence pulse when it receives the initialization pulse. The protocol for initialization is as follows:

1. Set the INITIALIZATION bit HDQ.HDQ_CTRL_STATUS[2] to 1 and the GO bit HDQ.HDQ_CTRL_STATUS[4] to 1 to send the pulse. When the pulse is sent, the bit is cleared in the register.
2. Wait for the presence detect flag (TIMEOUT bit HDQ.HDQ_INT_STATUS[0]) to generate an interrupt. This flag is set when the response time allowed to the slave has elapsed, whether it has sent a pulse or not.
3. Read the HDQ.HDQ_CTRL_STATUS register to check whether the presence pulse has been received before starting any transfer.

17.5.2 HDQ Protocol Basic Programming Model

17.5.2.1 Write Operation

The write operation sequence is as follows:

1. Write the command/address or data value to the TX write register (HDQ.HDQ_TX_DATA).

NOTE: Steps 2 and 3 can be performed simultaneously.

2. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to indicate a write.
3. Set GO bit HDQ.HDQ_CTRL_STATUS[4] to start the transmission.
   (a) The hardware sends the byte from the TX write register.
   (b) In a write operation, the TIMEOUT bit is always cleared, because there is no acknowledge mechanism from the slave.
   (c) When the write operation is completed, the TX-complete flag is set in the interrupt status register (TXCOMPLETE bit HDQ.HDQ_INT_STATUS[2]). If interrupts are masked (that is, the corresponding bit has been previously set in the control and status register), no interrupt signal is generated.
   (d) The GO bit HDQ.HDQ_CTRL_STATUS[4] is cleared at the end of a write operation.
4. The software must read the interrupt status register to clear the interrupt.
5. Repeat step 1 through step 4 for each successive byte to write.

17.5.2.2 Read Operation

The read operation sequence is as follows:

1. Write the command value to the TX write register (HDQ.HDQ_TX_DATA).
2. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to 0 and GO bit HDQ.HDQ_CTRL_STATUS[4] to 1 and wait for the TX-complete interrupt.

NOTE: Steps 3 and 4 can be performed simultaneously.

3. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to 1 to indicate a read.
4. Write 1 to GO bit HDQ.HDQ_CTRL_STATUS[4] to initiate the read.
   (a) The hardware detects a low-going edge on the line (generated by the slave) and receives 8 bits of data in the RX receive buffer register (HDQ.HDQ_RX_DATA). The first bit received is the LSB and the last bit is the MSB. The master performs this step as soon as the slave sends the data, irrespective of the state of GO bit HDQ.HDQ_CTRL_STATUS[4]. However, an RX-complete interrupt is generated only when the software writes the GO bit.
   (b) If a time-out occurs, the TIMEOUT bit HDQ.HDQ_CTRL_STATUS[0] is set.
   (c) Completion of the operation is indicated by setting the RX-complete flag in the interrupt status register (RXCOMPLETE bit HDQ.HDQ_INT_STATUS[1]. If interrupts are masked (that is, the corresponding bit was previously set in the control and status register), no interrupt signal is generated.
   (d) At the end of the read operation, the GO bit HDQ.HDQ_CTRL_STATUS[4] is cleared. It is also cleared if a time-out is detected
5. The software must read the interrupt status register (HDQ.HDQ_INT_STATUS) to determine whether an RX was successfully completed or a time-out occurred.
6. The software reads the RX receive buffer register (HDQ.HDQ_RX_DATA) to retrieve the read data from the slave.
7. Repeat step 1 through step 6 for each successive byte.

NOTE: In HDQ mode, the address/command is written only once to the slave. However, after the first byte is received, an RX-complete interrupt is set. Therefore, the software must initiate the read of the second byte by setting the GO bit in the control and status register. The first byte received is shadowed and provided to the software while the hardware is fetching the second byte of data.

17.5.3 1-Wire Mode (SDQ) Basic Programming Model

17.5.3.1 Write Operation

The write operation sequence is as follows:

1. Write the ID, command, or data value to the TX write register (HDQ.HDQ_TX_DATA).

NOTE: Steps 2 and 3 can be performed simultaneously.

2. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to 0 to indicate a write operation.
3. Set GO bit HDQ.HDQ_CTRL_STATUS[4] to 1 to start the transmission.
   (a) The hardware sends the byte stored in the TX write register.
   (b) In a write operation, the TIMEOUT bit is always cleared.
   (c) When the operation is completed, the TX-complete flag is set in the interrupt status register (TXCOMPLETE bit HDQ_INT_STATUS [2]). No interrupt signal is generated if interrupts are masked (that is, the corresponding bit was previously set in the control and status register).
   (d) The GO bit of the control and status register is cleared at the end of a write operation.

4. The software must read the interrupt status register (HDQ.HDQ_INT_STATUS) to clear the interrupt.
5. Repeat step 1 through step 4 for each successive byte to write.

17.5.3.2 Read Operation

The read operation sequence is as follows:

1. Write the address to the TX write register (HDQ.HDQ_TX_DATA).
2. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to 0 and the GO bit HDQ.HDQ_CTRL_STATUS[4] to 1 and wait for the TX-complete interrupt flag.
3. Write the command value to the TX write register (HDQ.HDQ_TX_DATA).
4. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to 0 and GO bit HDQ.HDQ_CTRL_STATUS[4] to 1 and wait for the TX-complete interrupt flag.

NOTE: Steps 5 and 6 can be performed simultaneously.

5. Set DIR bit HDQ.HDQ_CTRL_STATUS[1] to 1 to indicate a read.
6. Set GO bit HDQ.HDQ_CTRL_STATUS[4] to 1 to start the transmission.
   (a) The hardware (master) generates a low-going edge and clocks 8 bits of data into the RX receive register (HDQ.HDQ_RX_DATA). The first bit received is the LSB and the last bit is the MSB.
   (b) TIMEOUT bit HDQ.HDQ_INT_STATUS[0] is always cleared in a read operation.
   (c) When the operation is complete, the RX-complete flag is set in the interrupt status register (RXCOMPLETE bit HDQ.HDQ_INT_STATUS[1]). No interrupt signal is generated if the interrupts are masked (that is, the corresponding bit was previously set in the control and status register).
   (d) GO bit HDQ.HDQ_CTRL_STATUS[4] is cleared at the end of the read. It is also cleared if a time-out occurs.
7. The software must read the interrupt status register to determine whether an RX was successfully completed or a time-out occurred.
8. The software reads the RX receive buffer register (HDQ.HDQ_RX_DATA) to retrieve the read data from the slave.
9. Repeat step 1 through step 8 for each successive byte.

17.5.3.3 1-Wire Bit Mode Operation

Select the single-bit mode by setting the 1_WIRE_SINGLE_BIT bit HDQ.HDQ_CTRL_STATUS[7] to 1. In this mode, only one bit of data at a time is transferred between the master and the slave. After the bit is transferred, the corresponding interrupt flag is set (that is, there is an RX-complete (RXCOMPLETE bit HDQ.HDQ_INT_STATUS[1]) for a read operation and a TX-complete (TXCOMPLETE bit HDQ.HDQ_INT_STATUS[2]) for a write operation). Bit 0 of the RX register (HDQ.HDQ_RX_DATA) is updated each time a bit is received; bit 0 of the TX register (HDQ.HDQ_TX_DATA) contains the bit of data to be sent.

17.5.4 Power Management

The software has independent control of the two clock domains (interconnect clock: HDQ_ICLK and functional clock: HDQ_FCLK). Because there is no acknowledge mechanism from the HDQ/1-Wire module to an idle request, the software must ensure that a clock is not shut off while a transfer is being processed (the data would be lost).

If the autoidle function (AUTOIDLE bit HDQ.HDQ_SYSCONFIG[0] set to 1) provides a safe transfer (the module wakes up the HDQ_ICLK as soon as a transfer is initiated), the power-down mode and the PRCM idle requests (through the whole L4 clock domain idle request) must be handled carefully.

The following sections describe the steps to follow to use the power-down and idle modes.

17.5.4.1 Module Power-Down Mode

1. Before shutting off the HDQ_FCLK, wait for an RX-complete or a TX-complete interrupt.

- In a read operation, the transfer is completed when the RX-complete flag (RXCOMPLETE bit HDQ.HDQ_INT_STATUS [1]) generates an interrupt.
- In a write operation, the transfer is completed when the TX-complete flag (TXCOMPLETE bit HDQ.HDQ_INT_STATUS [2]) generates an interrupt. The software must check whether the interrupt was generated after the address/command byte was sent or after the data byte was sent. The clock must not be shut off after the command/ address byte is sent; otherwise, the data is not written to the slave.

HDQ.HDQ_INT_STATUS must be read to clear the interrupt condition.

2. Set the CLOCKENABLE bit HDQ.HDQ_CTRL_STATUS[5] to 0 to disable the clock.

   Do not access the module registers after the software has put the module into power-down mode except to write to the clock-enable bit to take the module out of power-down mode.

17.5.4.2 System Idle Mode

This section describes the steps to follow at the module level before enabling the idle mode at the system level (for more information about the system power management scheme, see , *Power, Reset, and Clock Management*.

As part of the L4 interconnect clock domain, the HDQ/1-Wire clocks can be cut at the PRCM level. HDQ_FCLK can be cut if the EN_HDQ bit PRCM.CM_FCLKEN1_CORE [22] is set to 0 and no other modules require CORE_12M_FCLK. The software must verify that no transfer is in progress.

In a read operation:

1. Wait for an RX-complete interrupt.

   In a read operation, the transfer is completed when the RX-complete flag (RXCOMPLETE bit HDQ.HDQ_INT_STATUS [1]) generates an interrupt.
2. Read the HDQ.HDQ_INT_STATUS to clear the read-complete interrupt flag.
3. Read the HDQ.HDQ_RX_DATA to retrieve the read data.
4. The HDQ_ICLK can be shut off by entering the system idle mode.

In a write operation:

1. Wait for a TX-complete interrupt.

   In a write operation, the transfer is completed when the TX-complete flag (TXCOMPLETE bit HDQ.HDQ_INT_STATUS[2]) generates an interrupt. The software must check whether the interrupt was generated after the address/command byte was sent or after the data byte was sent. The clock must not be shut off after the command/address byte is sent; otherwise, the data is not written to the slave.
2. Read the HDQ.HDQ_INT_STATUS to clear the TX-complete interrupt flag.
3. The HDQ_ICLK can be shut off by entering the system idle mode.

Concerning HDQ_ICLK, two situations can occur:

- The clock is no longer required and EN_HDQ bit PRCM.CM_ ICLKEN1_CORE[22] is set by software. In this case, the clock is cut off, provided no other modules require it. Before setting the EN_HDQ bit, the software must follow the steps described in this section.
- AUTO_HDQ bit PRCM.CM_AUTOIDLE1_CORE[22] is set. In this case, the software must verify that all HDQ/1-Wire transfers are complete before enabling the L4 interconnect clock domain idle mode. Otherwise, the HDQ/1-Wire has no way to prevent the clock from being cut, because no hardware mechanism exists. The steps listed in this section must be verified before putting the L4 clock domain into idle state.

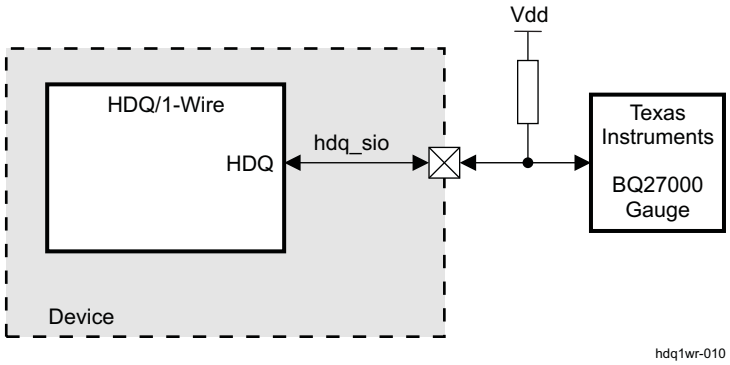
17.6 HDQ/1-Wire Use Cases and Tips

17.6.1 How to Configure the HDQ/1-Wire when Connected with a BQ27000 Gauge

17.6.1.1 Environment

Figure 17-10 details the OMAP device connections with the BQ27000 gauge.

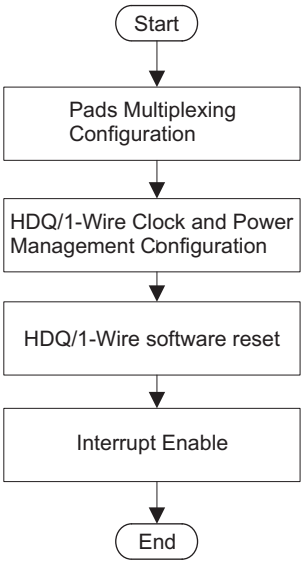
Figure 17-10. Environment



17.6.1.2 Programming Flow

This section details the programming flow of the HDQ/1-Wire. Figure 17-11 shows the main steps of this configuration. The BQ27000 gauge uses the HDQ mode.

Figure 17-11. HDQ/1-Wire Configuration in HDQ Mode



17.6.1.3 Pad Configuration and HDQ/1-Wire Clock and Power Management

Table 17-3 shows the pad multiplexing and the clock and power management configuration to select for the HDQ/1-Wire module.

Table 17-3. Registers Print for HDQ/1-Wire Configuration

Register Name	Address	Value	Value description
SCM.CONTROL_PADCONF_I2C3_SDA	0x 4800 21C4	0x0118 0100	Configure hdq_sio pad in mode 0

Table 17-3. Registers Print for HDQ/1-Wire Configuration (continued)

Register Name	Address	Value	Value description
PRCM.CM_FCLKEN1_CORE	0x4800 4A00	0x0020 0000	Enable HDQ/1-Wire Functional clock
PRCM.CM_ICLKEN1_CORE	0x4800 4A10	0x0020 0000	Enable HDQ/1-Wire Interface clock
HDQ_CTRL_STATUS	0x480B 200C	0x0000 0020	Enable clocks and select the HDQ mode
HDQ_SYSCONFIG	0x480B 2014	0x0000 0000	Module clock is free-running (Disable autoidle mode)

17.6.1.4 HDQ/1-Wire Software Reset

Perform a software reset as described in Figure 17-12.

Figure 17-12. Software Reset Flowchart

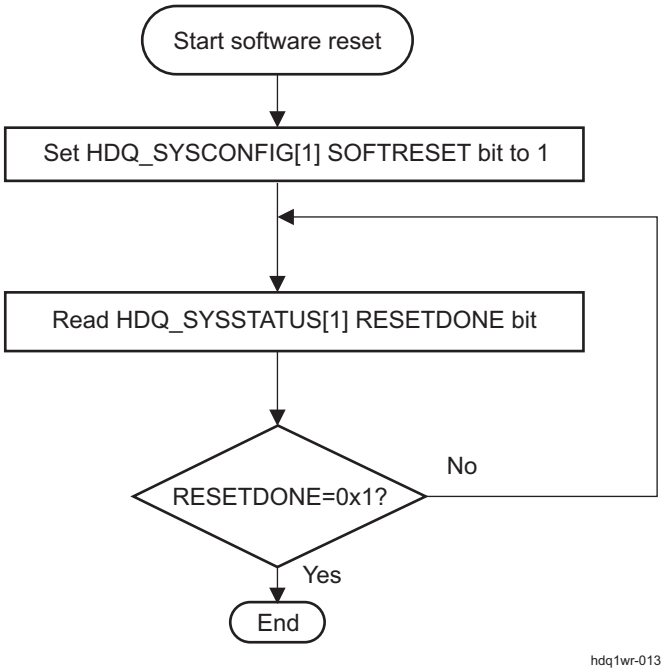


Table 17-4 describes the registers to be configured for the HDQ/1-Wire software reset step.

Table 17-4. Registers Print for HDQ/1-Wire Software Reset

Register Name	Address	Value	Value description
HDQ_SYSCONFIG	0x480B 2014	0x0000 0002	Initiate a software reset. The HDQ_SYSCONFIG[1] SOFTRESET is automatically reset by hardware.
HDQ_SYSSTATUS	0x480B 2018	0x0000 0001	The HDQ_SYSSTATUS[0] RESETDONE is set to 1 when the reset sequence is done.

17.6.1.5 Interrupts Enable

Table 17-5 describes the registers to be configured for the interrupts enable step and the use case values.

Table 17-5. Registers Print for HDQ/1-Wire Interrupts Enable

Register Name	Address	Value	Value description
HDQ_CTRL_STATUS	0x480B 200C	0x0000 0060	Enable Interrupts

17.6.1.6 Read and Write Operations

The Read and Write operations in HDQ mode are described in Section 17.5.2. Please, refer to this section to see how HDQ/1-Wire registers are configured for these operations.

Some write operations are needed to configure the BQ27000 gauge: for example, it is necessary to write a COMMAND KEY (0xA9 or 0x56) in the Device Control Register of the gauge. For more information, see the TI BQ27000 gauge specification.

17.7 HDQ/1-Wire Register Manual

17.7.1 HDQ/1-Wire Instance Summary

Table 17-6 lists the HDQ/1-Wire instances.

Table 17-6. Instance Summary

Module Name	Base Address	Size
HDQ/1-Wire	0x480B 2000	4K bytes

CAUTION

All reserved bits must be written with 0. There is no synchronization between the register clock domain and the state-machine domain. Therefore, the following rules must be observed when accessing the module registers:

- A read from the interrupt status register or the receive buffer register is not allowed unless the processor has been interrupted by the module.
- After the release of the GO bit in the control and status register, no access to the TX data register or the control and status register is allowed until the processor has been interrupted by the module.
- Polling of the interrupt status register by software to determine whether an interrupt was generated is not allowed.
- No access to the module registers should be done after the software puts the module in power-down mode (by setting bit 5 of the control and status register to 0) except to re-enable the clock.

CAUTION

The HDQ/1-Wire registers are limited to 32-bit data accesses. 16-bit and 8-bit are not allowed and can corrupt register content.

17.7.2 HDQ/1-Wire Register Mapping Summary

Table 17-7 lists the HDQ/1-Wire registers.

Table 17-7. HDQ/1-Wire Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	HDQ/1-Wire Physical Address
HDQ_REVISION	R	32	0x000	0x480B2000
HDQ_TX_DATA	RW	32	0x004	0x480B2004
HDQ_RX_DATA	R	32	0x008	0x480B2008
HDQ_CTRL_STATUS	RW	32	0x00C	0x480B200C
HDQ_INT_STATUS	R	32	0x010	0x480B2010
HDQ_SYSCONFIG	RW	32	0x014	0x480B2014
HDQ_SYSSTATUS	R	32	0x018	0x480B2018

17.7.3 HDQ/1-Wire Register Description

Table 17-8 through Table 17-20 describe the individual bits of the HDQ/1-Wire registers.

Table 17-8. HDQ_REVISION

Address Offset	0x000		
Physical Address	0x480B 2000	**Instance**	HDQ/1-Wire
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								REVISION							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0s.	R	0x000000
7:0	REVISION	IP revision The 4 LSBs indicate a minor revision. The 4 MSBs indicate a major revision. Ex: 0x21: Revision 2.1	R	See(1).

(1) TI internal data

Table 17-9. Register Call Summary for Register HDQ_REVISION

HDQ/1-Wire Register Manual • HDQ/1-Wire Register Mapping Summary: [0]

Table 17-10. HDQ_TX_DATA

Address Offset	0x004		
Physical Address	0x480B 2004	**Instance**	HDQ/1-Wire
Description	This register contains the data to be transmitted.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TX_DATA							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0s.	R	0x000000
7:0	TX_DATA	Transmit data (used in both HDQ and 1-Wire modes)	RW	0x00

Table 17-11. Register Call Summary for Register HDQ_TX_DATA

HDQ/1-Wire Environment • HDQ Protocol Initialization (Default): [0]
HDQ/1-Wire Functional Description • 1-Wire Single-Bit Mode Operation: [1]
HDQ/1-Wire Basic Programming Model • Write Operation: [2] • Read Operation: [3] • Write Operation: [4] • Read Operation: [5] [6] • 1-Wire Bit Mode Operation: [7]
HDQ/1-Wire Register Manual • HDQ/1-Wire Register Mapping Summary: [8]

Table 17-12. HDQ_RX_DATA

Address Offset	0x008		
Physical Address	0x480B 2008	**Instance**	HDQ/1-Wire
Description	This register contains the data to be received.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								RX_DATA							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0s.	R	0x000000
7:0	RX_DATA	Receive data (used in both HDQ and 1-Wire modes)	R	0x00

Table 17-13. Register Call Summary for Register HDQ_RX_DATA

HDQ/1-Wire Functional Description
- Description: [0]
- 1-Wire Single-Bit Mode Operation: [1]

HDQ/1-Wire Basic Programming Model
- Read Operation: [2] [3]
- Read Operation: [4] [5]
- 1-Wire Bit Mode Operation: [6]
- System Idle Mode: [7]

HDQ/1-Wire Register Manual
- HDQ/1-Wire Register Mapping Summary: [8]

Table 17-14. HDQ_CTRL_STATUS

Address Offset	0x00C		
Physical Address	0x480B 200C	**Instance**	HDQ/1-Wire
Description	This register provides status information about the module.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								1_WIRE_SINGLE_BIT	INTERRUPTMASK	CLOCKENABLE	GO	PRESENCEDETECT	INITIALIZATION	DIR	MODE

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reads return 0s.	R	0x000000
7	1_WIRE_SINGLE_BIT	Single-bit mode for 1-Wire 0x0: Disabled 0x1: Enabled	RW	0
6	INTERRUPTMASK	Interrupt masking bit 0x0: Disable interrupts 0x1: Enable interrupts	RW	0
5	CLOCKENABLE	Power down mode bit	RW	0

Bits	Field Name	Description	Type	Reset
		0x0: Disable clocks 0x1: Enable clocks		
4	GO	Go bit Write 1 to send the appropriate commands. Bit returns to 0 after the command is complete.	RW	0
3	PRESENCEDETECT	Presence detect received, 1-Wire mode only 0x0: Not detected 0x1: Detected	R	0
2	INITIALIZATION	Write 1 to send initialization pulse. Bit returns to 0 after pulse is sent.	RW	0
1	DIR	DIR bit, determines if next command is read or write 0x0: Write 0x1: Read	RW	0
0	MODE	Mode selection bit 0x0: HDQ mode 0x1: 1-Wire mode	RW	0

Table 17-15. Register Call Summary for Register HDQ_CTRL_STATUS

HDQ/1-Wire Environment
- HDQ Protocol Initialization (Default): [0]

HDQ/1-Wire Functional Description
- HDQ/1-Wire Block Diagram: [1]
- Description: [2]
- Single-Bit Mode: [3]
- Description: [4] [5]
- 1-Wire Single-Bit Mode Operation: [6]
- Status Flags: [7]
- Power-Down Mode: [8]

HDQ/1-Wire Basic Programming Model
- Mode Selection: [9]
- Reset/Initialization: [10] [11] [12] [13] [14]
- Write Operation: [15] [16] [17]
- Read Operation: [18] [19] [20] [21] [22] [23] [24]
- Write Operation: [25] [26]
- Read Operation: [27] [28] [29] [30] [31] [32] [33]
- 1-Wire Bit Mode Operation: [34]
- Module Power-Down Mode: [35]

HDQ/1-Wire Use Cases and Tips
- Pad Configuration and HDQ/1-Wire clock and power management: [36]
- Interrupts Enable: [37]

HDQ/1-Wire Register Manual
- HDQ/1-Wire Register Mapping Summary: [38]

Table 17-16. HDQ_INT_STATUS

Address Offset	0x010		
Physical Address	0x480B 2010	**Instance**	HDQ/1-Wire
Description	This register controls interrupt status.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													TXCOMPLETE	RXCOMPLETE	TIMEOUT

Bits	Field Name	Description	Type	Reset
31:3	Reserved	Reads return 0s.	R	0x00000000
2	TXCOMPLETE	TX-complete interrupt flag Set to 1 if cause of interrupt. Set to 0 when register read.	R	0
1	RXCOMPLETE	Read-complete interrupt flag Set to 1 if cause of interrupt. Set to 0 when register read.	R	0
0	TIMEOUT	Presence detect/timeout interrupt flag In 1-Wire mode, set to 1 if slave's presence detected. In HDQ mode, set to 1 if timeout on read occurs. Set to 0 when register read.	R	0

Table 17-17. Register Call Summary for Register HDQ_INT_STATUS

HDQ/1-Wire Functional Description
- Description: [0] [1] [2]
- Interrupt Conditions: [3] [4] [5]
- Description: [6] [7]
- Interrupt Conditions: [8] [9] [10]
- Status Flags: [11]

HDQ/1-Wire Basic Programming Model
- Reset/Initialization: [12]
- Write Operation: [13]
- Read Operation: [14] [15]
- Write Operation: [16] [17]
- Read Operation: [18] [19]
- 1-Wire Bit Mode Operation: [20] [21]
- Module Power-Down Mode: [22] [23] [24]
- System Idle Mode: [25] [26] [27] [28]

HDQ/1-Wire Register Manual
- HDQ/1-Wire Register Mapping Summary: [29]

Table 17-18. HDQ_SYSCONFIG

Address Offset	0x014		
Physical Address	0x480B 2014	**Instance**	HDQ/1-Wire
Description	This register controls various bits.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Reads return 0s.	R	0x00000000
1	SOFTRESET	Start soft reset sequence. 0x0: Disabled 0x1: Enabled	RW	0
0	AUTOIDLE	Interconnect idle 0x0: Module clock is free-running. 0x1: Module is in power saving mode: Clock is running only when module is accessed or inside logic is in function to process events.	RW	0

Table 17-19. Register Call Summary for Register HDQ_SYSCONFIG

HDQ/1-Wire Integration • HDQ/1-Wire Reset Scheme: [0]
HDQ/1-Wire Functional Description • Autoidle Mode: [1]
HDQ/1-Wire Basic Programming Model • Power Management: [2]
HDQ/1-Wire Use Cases and Tips • Pad Configuration and HDQ/1-Wire clock and power management: [3] • HDQ/1-Wire Software Reset: [4] [5]
HDQ/1-Wire Register Manual • HDQ/1-Wire Register Mapping Summary: [6]

Table 17-20. HDQ_SYSSTATUS

Address Offset	0x018		
Physical Address	0x480B 2018	**Instance**	HDQ/1-Wire
Description	This register monitors the reset sequence.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reads return 0s.	R	0x00000000
0	RESETDONE	Reset monitoring 0x0: The module is currently performing its reset. When the module is in power-down mode, set to 0 to indicate this fact. 0x1: The module has finished its reset.	R	0x-

Table 17-21. Register Call Summary for Register HDQ_SYSSTATUS

HDQ/1-Wire Use Cases and Tips • HDQ/1-Wire Software Reset: [0] [1]
HDQ/1-Wire Register Manual • HDQ/1-Wire Register Mapping Summary: [2]

Multichannel Buffered Serial Port

This chapter introduces the multichannel buffered serial port (McBSP).

Topic **Page**

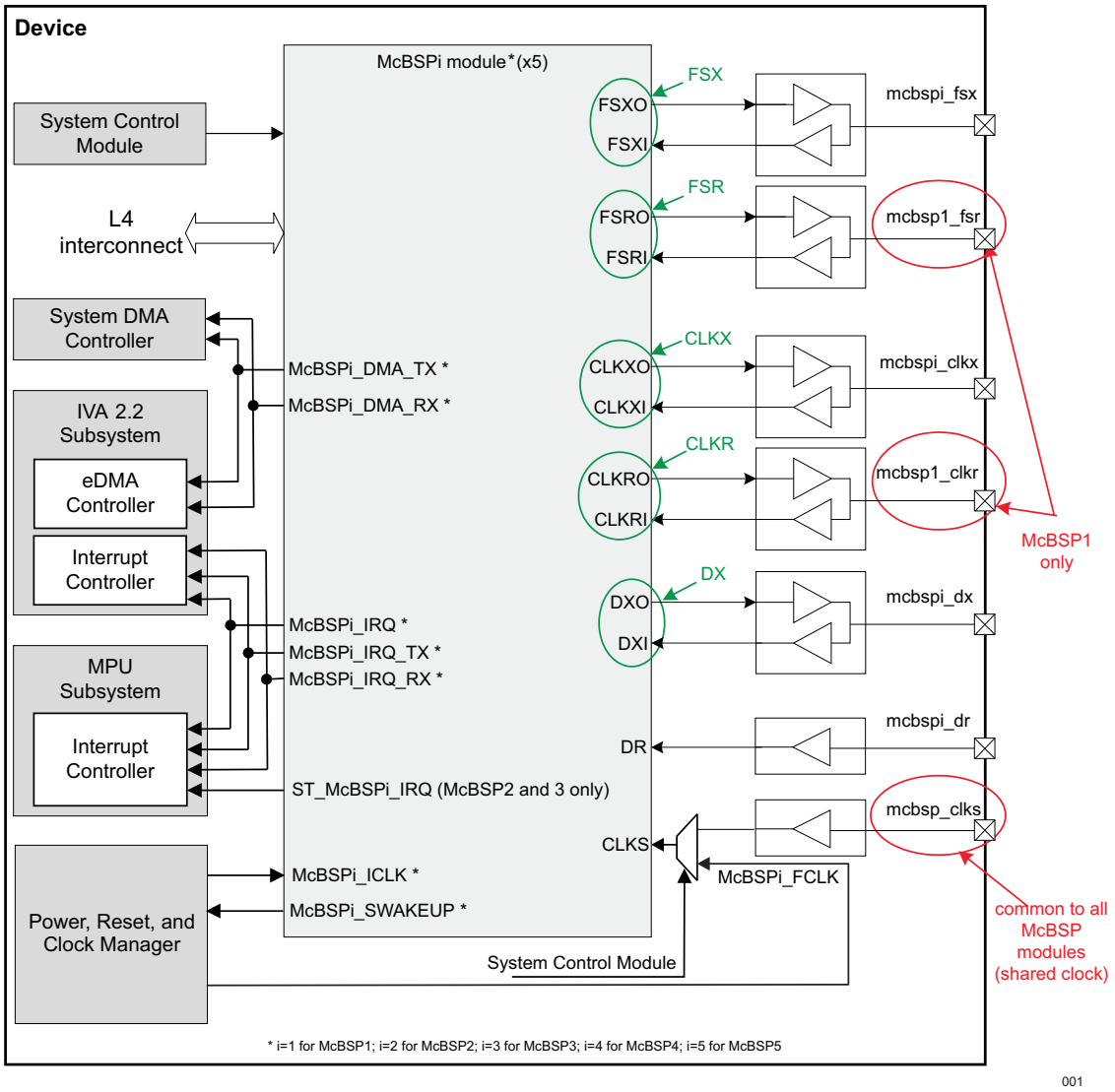
18.1 McBSP Overview

The multichannel buffered serial port provides a full-duplex direct serial interface between the device and other devices in a system such as other application chips (digital base band), audio and voice codec (power management device), etc. Because of its high level of versatility, it can accommodate to a wide range of peripherals and clocked frame oriented protocols (for details, see Section 18.1.1).

The device provides five instances of the McBSP module.

Figure 18-1 shows the McBSP overview in the device.

Figure 18-1. McBSP Highlight



18.1.1 McBSP Features

The main features of the McBSP modules are:

- L4 interconnect slave interface supports:
  - 32-bit data bus width
  - 32-bit access supported
  - 16- /8-bit access supported only by data registers
  - 10-bit address bus width

  - Burst mode not supported
  - Write nonposted transaction mode supported
- 128 × 32-bit words (512 bytes) for each buffer for transmit/receive operations (McBSP1, 3, 4, 5)
- 5K bytes (1024 x 32 bits for audio buffer + 256 × 32 bits for buffer) for each buffer for transmit/receive audio operations (McBSP2 only)
- Interrupts configurable in legacy mode (2 requests) or PRCM compliant (1 request)
- Transmit and receive DMA requests triggered with programmable FIFO thresholds
- SIDETONE core support: Audio loopback capability (McBSP2 and 3 only)
- Multidrop support
- Serial interface description
  - 6 pin configuration (McBSP 1 only)
  - 4 pin configuration (McBSP2, 3, 4, 5)
  - Full-duplex communication
  - Multichannel selection modes
    - Support to enable or block transfers in each of the channels
    - 128 channels for transmission and for reception
  - Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices:
    - Inter-IC sound (I2S) compliant devices
    - Pulse code modulation (PCM) devices
    - Time division multiplexed (TDM) bus devices

CAUTION

McBSP modules do not offer support for μ-law and A-law companding, two partitions mode dynamic reassignment, AC'97, and SPI protocol.

  - A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits
  - Bit reordering (send/receive least significant bit [LSB])
- Clock and frame-synchronization generation support:
  - Independent clocking and framing for reception and for transmission up to 48 MHz
  - Support for external generation of clock signals and frame-synchronization (frame-sync) signals
  - A programmable sample rate generator for internal generation and control of clock signals and frame-sync signals
  - Programmable polarity for frame-sync pulses and for clock signals

NOTE:

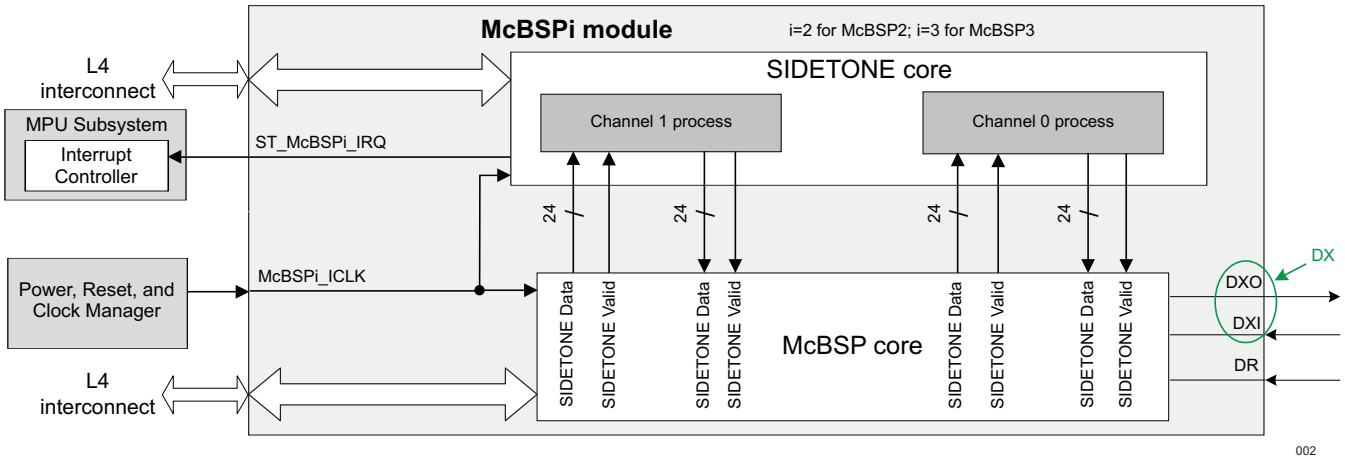
- McBSP modules do not support features such as re-transmit or re-receive of an erroneous frame or word.
- McBSP modules support dual phase frames to provide I2S fully compliant capabilities. But, this dual phase mode is limited at one channel (or word) for each phase instead of 128 channels max for single phase mode.

18.1.2 SIDETONE Core

The purpose of this section is to present the SIDETONE core implemented in McBSP2 and 3 modules. It is required that two of the audio input channels to be looped back, filtered, and mixed to the two corresponding audio output channels.

Data to be processed comes on two separate paths (one for each channel) through a simple interface. After filtering the data, gain is applied and outputted through a similar interface. For more details, see Figure 18-2).

Figure 18-2. SIDETONE Core Architecture



The SIDETONE core offers the following features:

- Filter coefficients are shared between the two channels (the filter coefficient is assumed to represent values in the range –1…+1)
- Gains are independent for the two channels (the gain coefficients are in the range –2…+2)
- The filter output is multiplied with the gain, and the result is rounded to output channel word width.
- The FIR filter length is 128 samples.
- Filtered loop back signals are added to the corresponding output signals, and saturation is applied if the result exceeds arithmetic range.
- Filter coefficients and gains are programmable:
  - Coefficients are changed while audio is stopped.
  - Gains are changed at any time (also during audio operation with SIDETONE enabled).

18.2 McBSP Environment

This section describes the intended functions for McBSP module from an environment point of view (that is, external connections). It presents the McBSP connectivity options, lists the possible interfaces, and details the protocol and data format used in each case.

18.2.1 McBSP Functions

The device provides five instances of the McBSP module, called McBSP1, McBSP2, McBSP3, McBSP4, and McBSP5.

List of recommended usage (non exhaustive) per McBSP modules in the device:

- McBSP1: Digital baseband (DBB) Data
- McBSP2: Audio data with audio buffer and SIDETONE feature
- McBSP3: Bluetooth voice data with SIDETONE feature
- McBSP4: DBB voice data
- McBSP5: Midi data

Table 18-1 describes the functions and the corresponding application fields.

Table 18-1. Functions Description

Function	Application field	Recommended McBSP module	Description
Control and Data	Digital Base Band (DBB) Data	McBSP1	Serial interface to transfer data
Audio Data	Audio Data with Audio Buffer	McBSP2	Audio interface to transfer audio data with Inter-IC Sound codec (I2S)
	Audio Data with Audio Buffer and SIDETONE feature		
	Midi Data	McBSP5	
Voice Data	Bluetooth Voice Data	McBSP3	Voice interface to transfer voice data with Pulse Code Modulation codec (PCM)
	Bluetooth Voice Data with SIDETONE feature		
	DBB Voice Data	McBSP4	

18.2.2 McBSP Signals Descriptions

The five McBSP modules consist of a data-flow path and a control path connected to external devices by a serial interface with 6 pins configuration (McBSP 1 only) or 4 pins configuration (McBSP2, 3, 4, 5).

For a McBSP module with 6 pins configuration, an internal loop back capability between Transmitter and Receiver clock signals, and both frame synchronization signals enables using the McBSP module with 4 pins configuration. The related internal multiplexers are controlled through the System Control Module on the device (see Section 18.3).

Table 18-2. Input/Output Description

Pin Name[1]	I/O[2]	Description	Internal Signal Name	Reset Value	Control and Data	Audio Data	Voice Data
mcbsp_clks	I	External clock (shared by all McBSP modules)	CLKS	-	✓	✓	✓
mcbspi_dr	I	Receive serial data	DR	-	✓	✓	✓
mcbspi_dx	I/O[3]	Transmit serial data	DX	0	✓	✓	✓
mcbspi_clkx	I/O [3]	Transmit clock[4]	CLKX	0	✓	✓	✓
mcbspi_fsx	I/O [3]	Transmit frame synchronization [5]	FSX	0	✓	✓	✓
mcbsp1_clkr	I/O [3]	Receive clock	CLKR	1	✓		
mcbsp1_fsr	I/O [3]	Receive frame synchronization	FSR	0	✓		

(1) I = 1 to 5
(2) I = Input; O = Output
(3) For details of the input/output selection, see Section 18.2.3.1.
(4) This signal is also used as CLKR when it is configured as output (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).
(5) This signal is also used as FSR when it is configured as output (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).

- The mcbsp_clks pin can be used to inject an external clock. This clock is used to generate control signals depending on the module internal configuration (see Section 18.4.3). The CLKS signal of the McBSP modules is linked to an external signal through the mcbsp_clks pin, but the CLKS signal can also be linked to an internal clock provided by PRCM of the device. For more information, see Section 18.3.
- Data are transmitted to external devices interfacing with McBSP modules via the mcbspi_dx pin. Data from those devices are received on the mcbspi_dr pin.

NOTE: The mcbspi_dx pin is an input/output signal to use the McBSP module in half-duplex mode.

- Control information is communicated via the following pins: mcbspi_clkx (transmit clock), mcbsp1_clkr (receive clock), mcbspi_fsx (transmit frame-sync), and mcbsp1_fsr (receive frame-sync).

CAUTION

External pins mcbsp1_clkr and mcbsp1_fsr are connected to pads only for McBSP1 module; other McBSP modules don't have these connections. For these modules, CLKR and FSR signals sources are mcbspi_clkx and mcbspi_fsx pins, respectively. Consequently, there is a light restriction on other McBSP modules when used in full-duplex mode. Both reception and transmission use the same clock signal and the same frame synchronization signal.

18.2.3 McBSP Functions Description

18.2.3.1 McBSP Modes

For all McBSP functions, McBSP modules can operate in master or slave mode. The difference between these modes is the definition of the source of McBSP clocks and McBSP frames synchronization:

- Master mode: McBSP module provides them to the external device
- Slave mode: McBSP module receives them from the external device

The choice between the two modes depends of technical data of the external device and the type of interface (protocols and data formats). For one McBSP module, there are four possible functions:

1. Transmit and receive master mode
2. Transmit and receive slave mode
3. Transmit master mode and receive slave mode
4. Transmit slave mode and receive master mode

NOTE: If the McBSP module has a serial interface with 4 pins configuration (McBSP2, 3, 4, 5), only modes 1 or 2 are possible.

Figure 18-3 shows the connection between the McBSP1 module (6 pins configuration) and an external device in transmit master mode and receive slave mode.

Figure 18-3. Mode Overview of McBSP1 Module

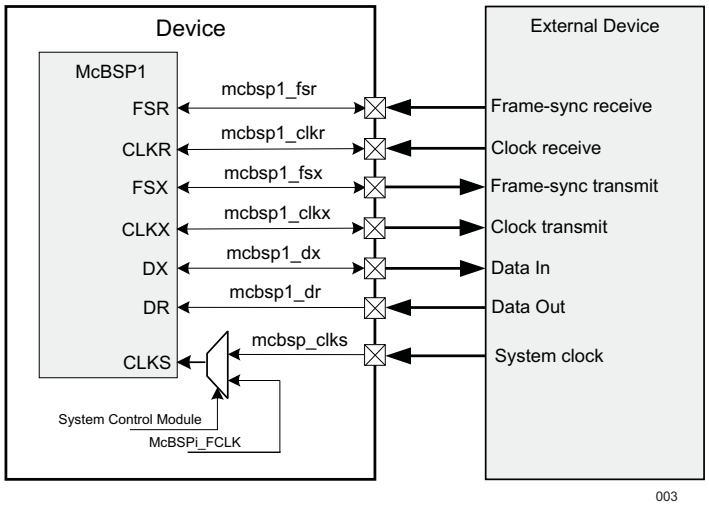
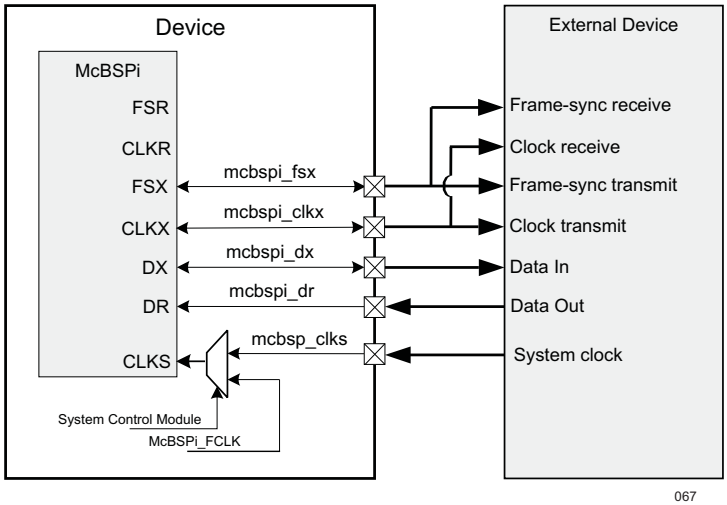


Figure 18-4 shows the connection between the McBSPi module, with I = 2, 3, 4 or 5 (4 pins configuration) and an external device in transmit and receive master mode.

Figure 18-4. Mode Overview of McBSPi Module

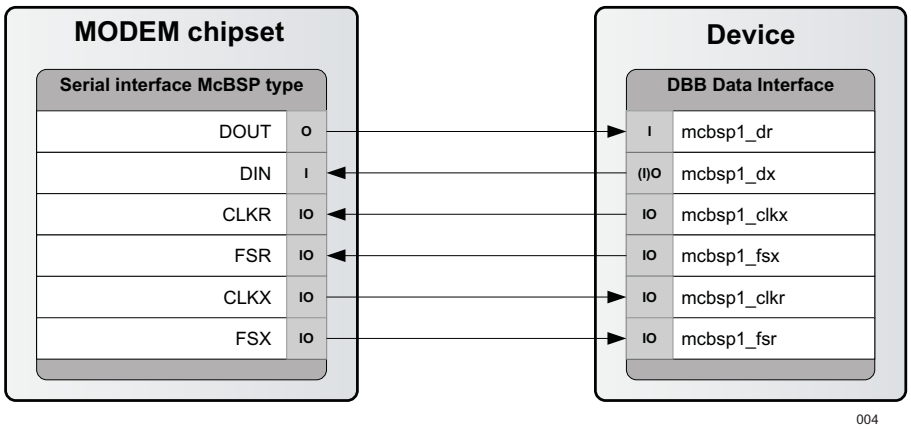


18.2.3.2 McBSP Functions

18.2.3.2.1 McBSP Function 1: Control and Data

In full-duplex mode (reception and transmission use independent clock signals and frame synchronization signals), the McBSP module can be used to exchange control and data with an external chipset, allowing the device to be interfaced with a modem device. Figure 18-5 shows typical connections between device and modem chipset to illustrate DBB data application.

Figure 18-5. DBB Data Application



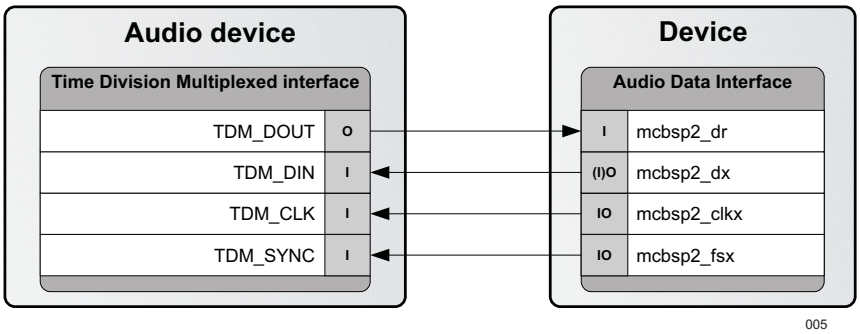
In Figure 18-5, the McBSP1 module is configured in transmit master mode and in receive slave mode.

18.2.3.2.2 McBSP Function 2: Audio Data

The McBSP module is connected to audio devices through the I2S interface. The I2S link serial interface is a TDM slot based serial interface that is used to transfer audio data. Those audio devices can be either AICs or other serially connected A/D and D/A devices.

Figure 18-6 shows typical connections between device and a typical device of analog audio interface (power managment device) to illustrate Audio Data application. The typical device contains several audio analog inputs and outputs, as well as digital microphone inputs.

Figure 18-6. Audio Data Application



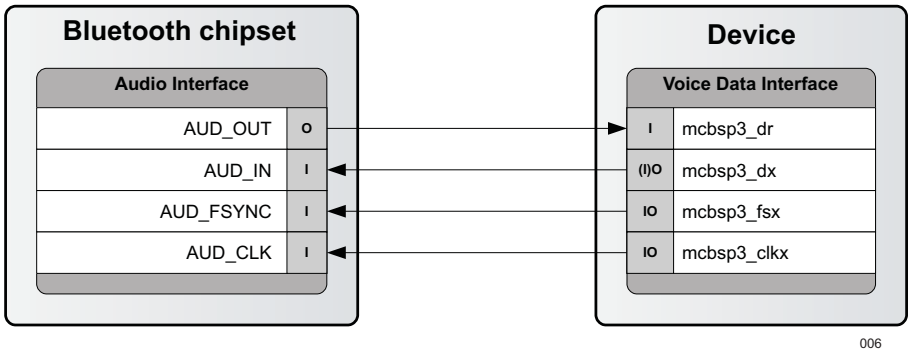
In Figure 18-6, the McBSP2 module is configured in transmit and receive master mode.

18.2.3.2.3 McBSP Function 3: Voice Data

The McBSP module is connected to a voice device through the PCM interface. The PCM link serial interface is a TDM slot based serial interface that is used to transfer voice data. The voice devices can be either modem chipsets, Bluetooth chipsets or others devices with voice data interface.

Figure 18-7 shows typical connections between device and Bluetooth chipset (TI BRF6300 or TI BRF6350) to illustrate voice data application.

Figure 18-7. Voice Data Application



In Figure 18-7, the McBSP3 module is configured in transmit and receive master mode.

18.2.4 McBSP Protocols and Data Formats

The McBSP module can use one of the three protocols with associated data formats:

- Serial protocol to exchange serial data
- Audio protocol to exchange the audio samples
- Voice protocol to exchange the voice samples

The McBSP modules offer the flexibility to modify the following parameters to adapt to the device features as described in the following subsections.

18.2.4.1 Words, Frames, and Phases Definitions

18.2.4.1.1 Words or Channels

The data bits are transferred (transmission or reception) in a group called a serial word or channel. The number of bits in a word (length) is programmable via bits field (McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 field and McBSPi.MCBSPLP_RCR2_REG[7:5] RWDLEN2 field, McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 field and McBSPi.MCBSPLP_XCR2_REG[7:5] XWDLEN2 field) and can be 8, 12, 16, 20, 24 or 32 bits (see Section 18.4.2.3). The McBSP module uses clock signals to control the time for each bit transfer. Data are sampled/driven on rising or falling edge of clock signals. This clock polarity is programmable via bits field of pin-control register (McBSPi.MCBSPLP_PCR_REG).

For more information, see Section 18.4.2.4.

18.2.4.1.2 Frames

One or more words (max 128) are transferred in a group called a frame. The McBSP module can transmit / receive a maximum of 128 words per frame, programmable via bits field of transmit and receive control registers (McBSPi.MCBSPLP_XCR1_REG/McBSPi.MCBSPLP_XCR2_REG and McBSPi.MCBSPLP_RCR1_REG/McBSPi.MCBSPLP_RCR2_REG). For more details, see Section 18.4.2.3.

All the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP module uses frame-synchronization signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization signal, the McBSP module begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP module receives/transmits the next frame, and so on. Frame-synchronization pulse is active high or low. This pulse polarity is programmable via bits field of pin-control register (McBSPi.MCBSPLP_PCR_REG).

Each frame transfer can be delayed by 0, 1, or 2 clock cycles, depending on the value of bits for transmit and receive control registers (McBSPi.MCBSPLP_XCR2_REG and McBSPi.MCBSPLP_RCR2_REG). For more information, see Section 18.4.4.6.3 and Section 18.4.4.3.3.

18.2.4.1.3 Phases

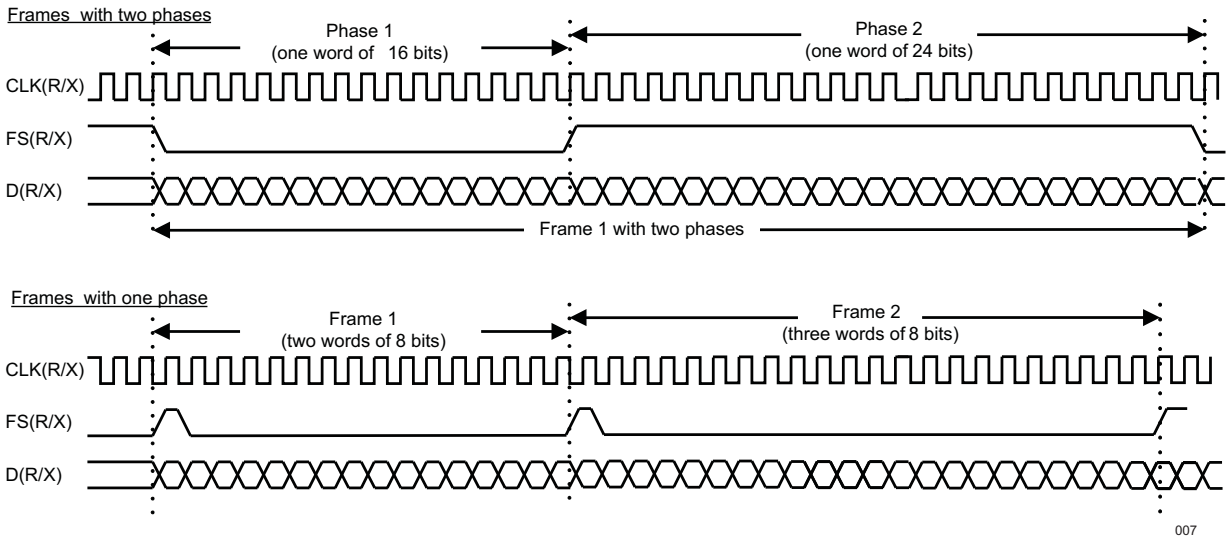
The McBSP module allows configuring each frame to contain one or two phases. The McBSP module supports dual phase frames to provide I2S fully compliant capabilities. These two phases represent left and right channels of audio stereo signals.

The limitation on dual phase frame is that the number of words per phase must be set to one for both first and second phase. But, the number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers.

For example, software may define a frame composed of a first phase with one 12-bit word and a second phase with one 16-bit word. This configuration allows the software to compose frames for custom applications. For more details, see Section 18.4.2.4.

Figure 18-8 shows signal activity for two possible reception/transmission scenarios.

Figure 18-8. McBSP Reception/Transmission Signal Activity



18.2.4.2 Serial Protocol and Data Formats

18.2.4.2.1 Protocol

The serial protocol is used to send and receive control data without specific formats. This allows McBSP module to accommodate all serial devices and their protocols.

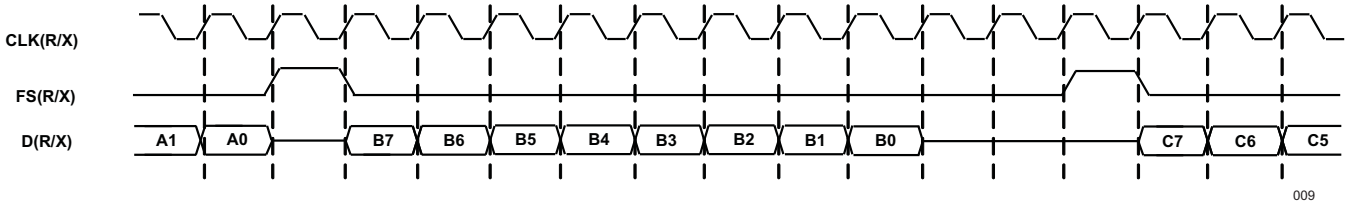
18.2.4.2.2 Data Format

Figure 18-9 shows typical operation of the McBSP clock and frame sync signals. Serial clocks CLKR and CLKX define the boundaries between bits for receive and transmit, respectively. Similarly, frame-sync signals FSR and FSX define the beginning of an element and/or frame transfer. The McBSP module allows the configuration of the following parameters for data and frame synchronization:

- Polarity of FSR, FSX, CLKX, and CLKR
- The number of words per frame
- The number of bits per word
- Whether subsequent frame synchronization restarts the serial data stream or is ignored
- The data delay from frame synchronization to first data bit which can be 0- , 1- , or 2-bit delays

The configuration is independent for receive and transmit parts. For more details and configuration examples, see Section 18.4 and Section 18.5.

Figure 18-9. Serial Data Formats



18.2.4.3 Audio Protocol and Data Formats

18.2.4.3.1 Protocol

The I2S protocol is used to send and receive audio data from 8 KHz up to 48 KHz sampling rate (frame-sync frequency), with 16 bits or 32 bits per words (Supported frequencies are 8, 11.025, 12, 16, 22.05, 24, 32, 44.1 and 48 KHz).

The frame-synchronization signal defines the frame length in the I2S protocol. Each frame consists of a fixed number of words. In dual-phase frame, the frame-synchronization signal is low for the left phase time slot and is high for the right phase time slot. In addition, the frame-synchronization signal is synchronous to the falling edge of the clock signal.

18.2.4.3.2 Data Formats

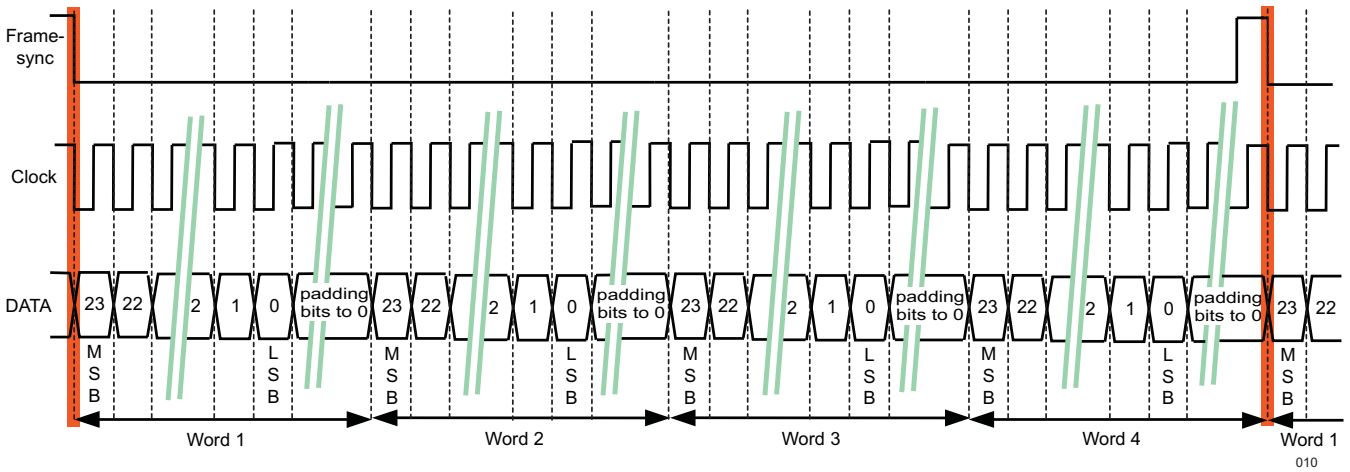
The I2S protocol supports TDM, I2S, left justified, and right justified data formats.

Bits of each word (sample) are clocked using clock signal. For each word, MSB is first. LSBs are padded to 0 when the data length (8, 12, 16, 20, or 24 bits) is less than the sample word width (16 or 32 bits).

18.2.4.3.2.1 TDM Data Format

Figure 18-10 shows that each frame of TDM data format is composed of four words (or channels).

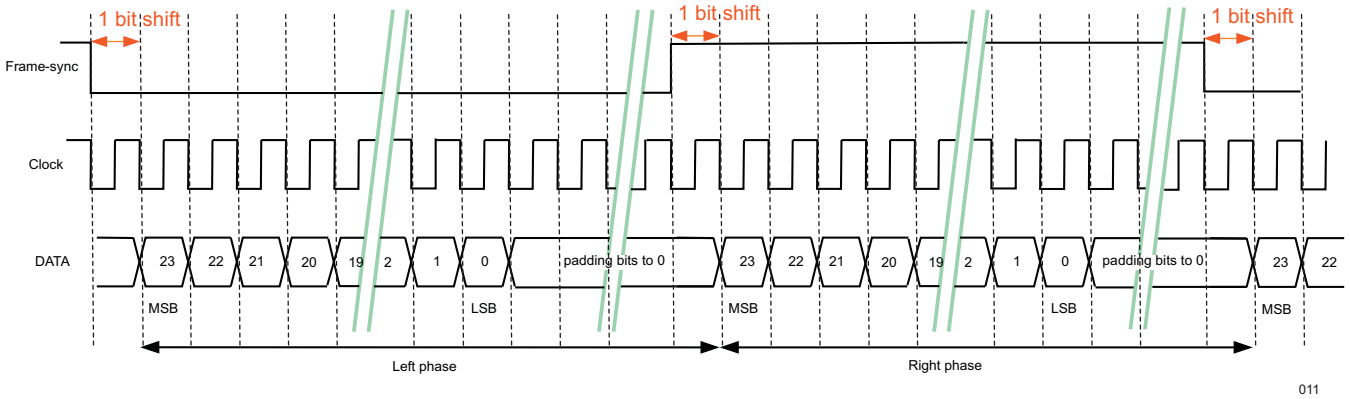
Figure 18-10. TDM Data Format; Word Width: 32 Bits; Data Length: 24 Bits



18.2.4.3.2.2 I2S Data Format

Figure 18-11 shows an example with 24 bits data (MSB first) and 8 padding bits at '0'.

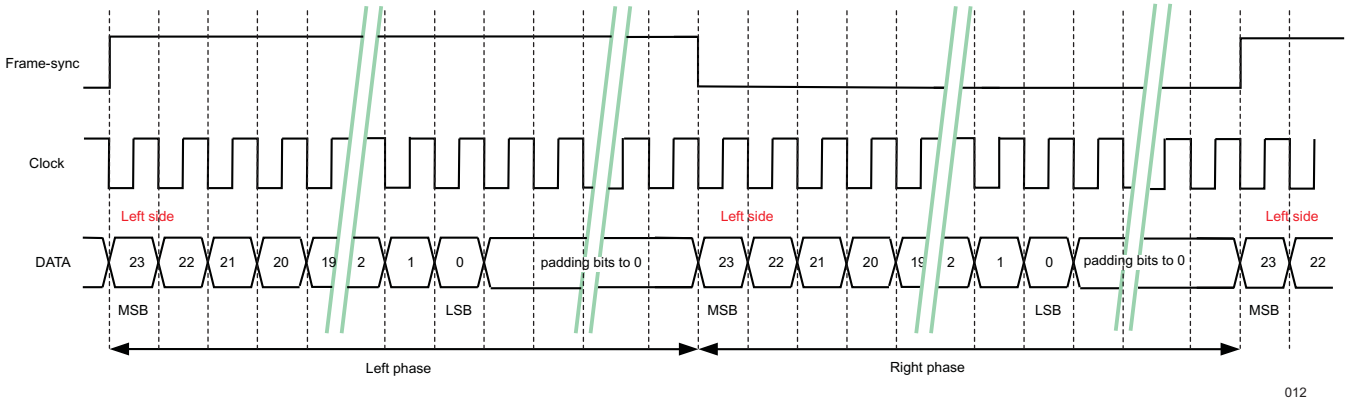
Figure 18-11. I2S Data Format; Word Width: 32 Bits; Data Length: 24 Bits



18.2.4.3.2.3 Left Justified Data Format

Figure 18-12 shows an example with 24 bits data (MSB first) and 8 padding bits at '0'.

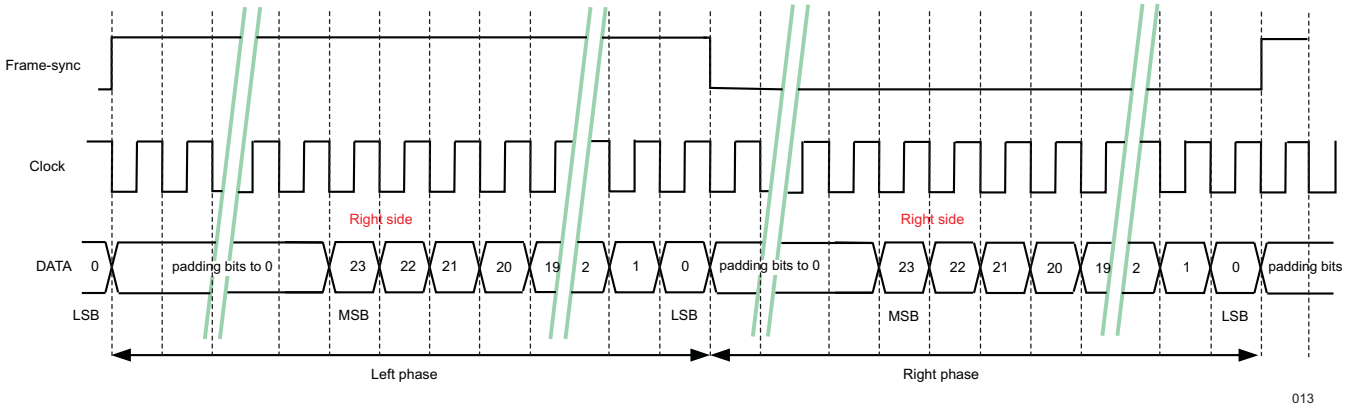
Figure 18-12. Left Justified Data Format; Word Width: 32 Bits; Data Length: 24 Bits



18.2.4.3.2.4 Right Justified Data Format

Figure 18-13 shows an example with 8 padding bits at '0' and 24 bits data (MSB first).

Figure 18-13. Right Justified Data Format; Word Width: 32 Bits; Data Length: 24 Bits



18.2.4.4 Voice Protocol and Data Formats

18.2.4.4.1 Protocol

The PCM protocol is intended to transfer voice data at 8 kHz (default narrowband mode) or 16 kHz (wideband mode) sample rates (frame-sync frequency). PCM protocol can act as a slave or master, and is used by the Bluetooth interface and the modem generic interface. The frame-synchronization defines the frame length in the PCM protocol. Bits are clocked using PCM clock signal, with MSB first.

18.2.4.4.2 Data Formats

Two modes are available for the PCM protocol: mode 1 and mode 2. For these both modes, it has two types of operations: Mono or stereo channels. The difference between PCM mode 1 and PCM mode 2 is in the way they use either the rising or the falling edge of clock signal, and the frame-synchronization polarity.

- PCM Mode 1: Input data is latched on the falling edge of the clock, and the transmitted data starts on the rising edge of the clock. Frame-synchronization pulse is active high.
- PCM Mode 2: Input data is latched on the falling edge of the clock, and the transmitted data starts on the falling edge of the clock. Frame-synchronization pulse is active low.

Figure 18-14 and Figure 18-15 shows an example of PCM protocol, mode 1 and mode 2, respectively, for a frame composed one word (width: 32 bits) with 16 bits data.

Figure 18-14. PCM Protocol - Mode 1 Data Format

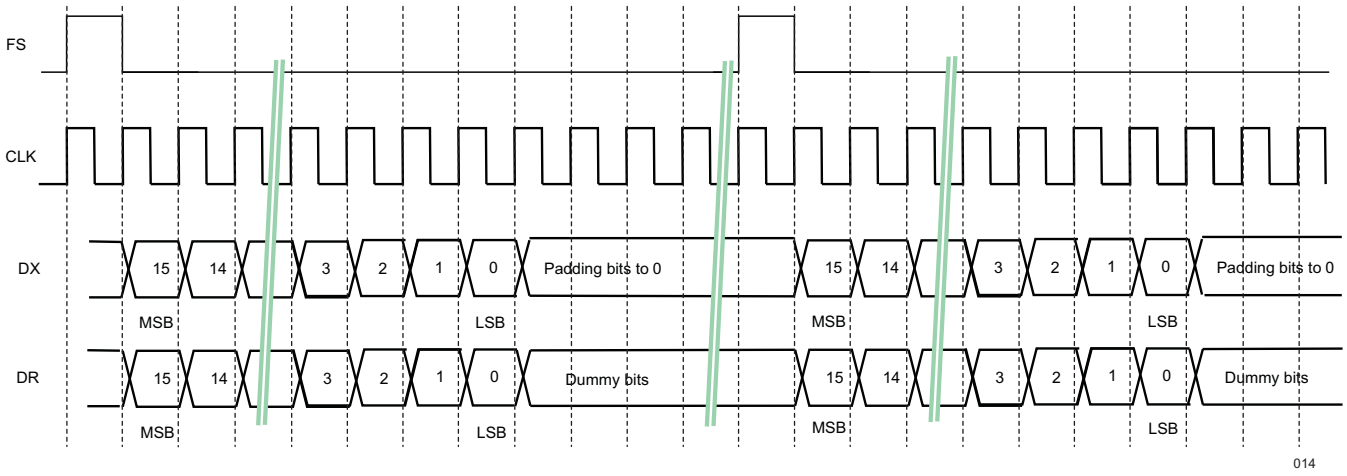
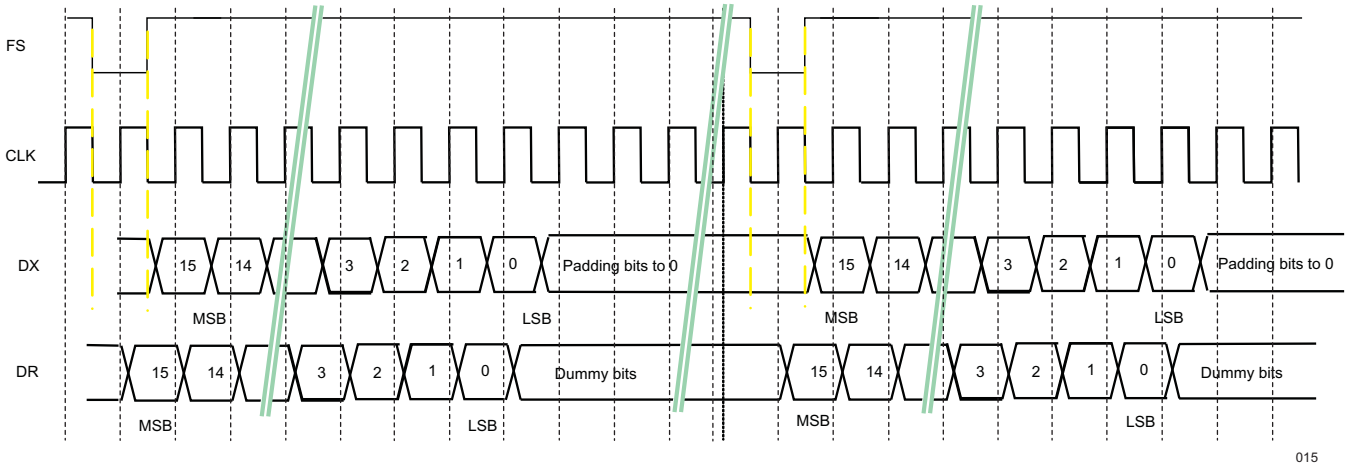


Figure 18-15. PCM Protocol - Mode 2 Data Format



18.3 McBSP Integration

This section describes the McBSP modules integration inside the device and all the details about signal source controls, clocks, resets, power management, and hardware requests.

McBSP modules are divided into 2 families: McBSP modules that are gated in CORE domain (McBSP1 and 5), and McBSP modules that are gated in PER domain (McBSP2, 3, and 4).

For more details on CORE and PER domain, see , *Power, Reset and Clock Management*.

Figure 18-16 through Figure 18-20 highlight the integration of the McBSP modules in the device including interrupt handlers, DMA requests, clock generators, and interconnections.

Figure 18-16. McBSP1 Integration

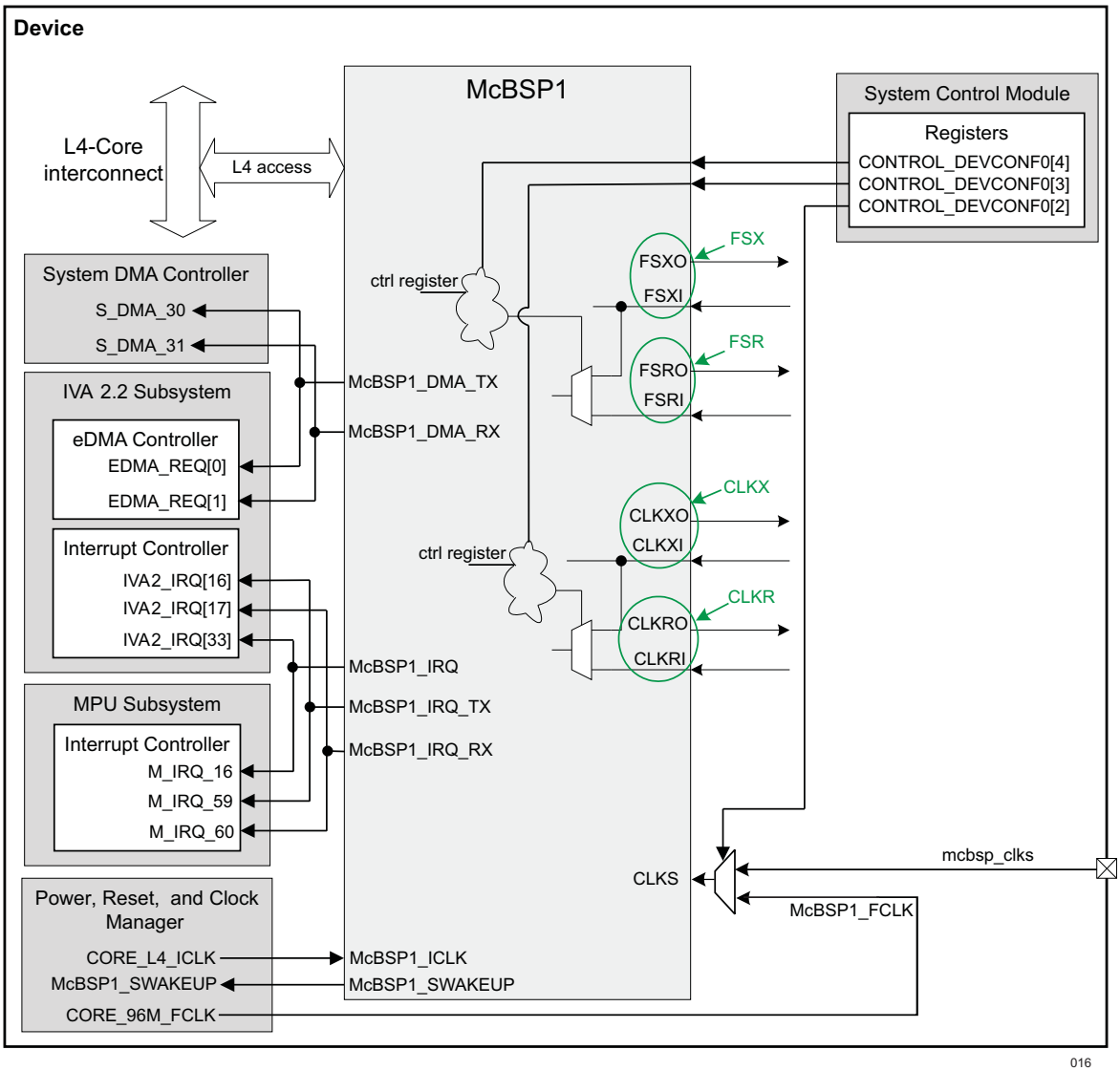


Figure 18-17. McBSP2 Integration

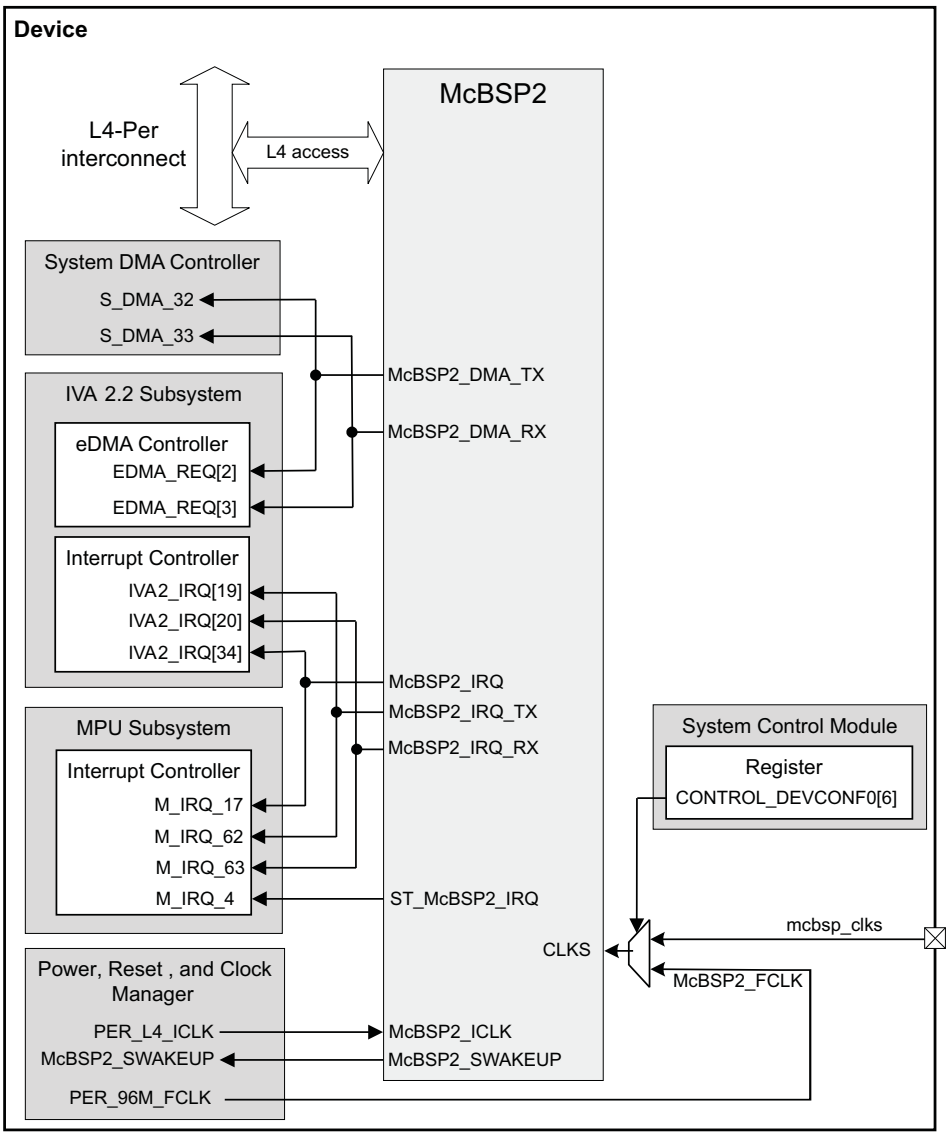


Figure 18-18. McBSP3 Integration

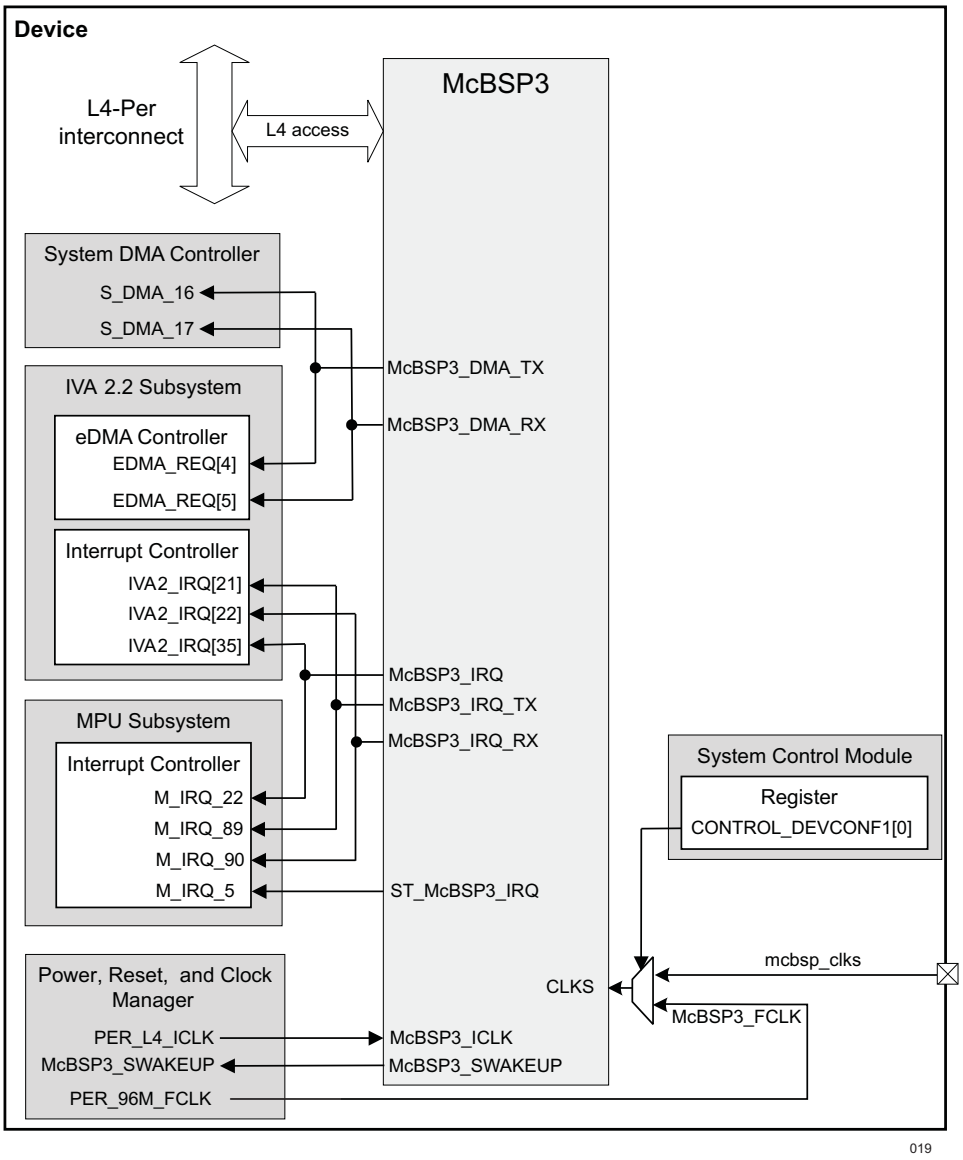


Figure 18-19. McBSP4 Integration

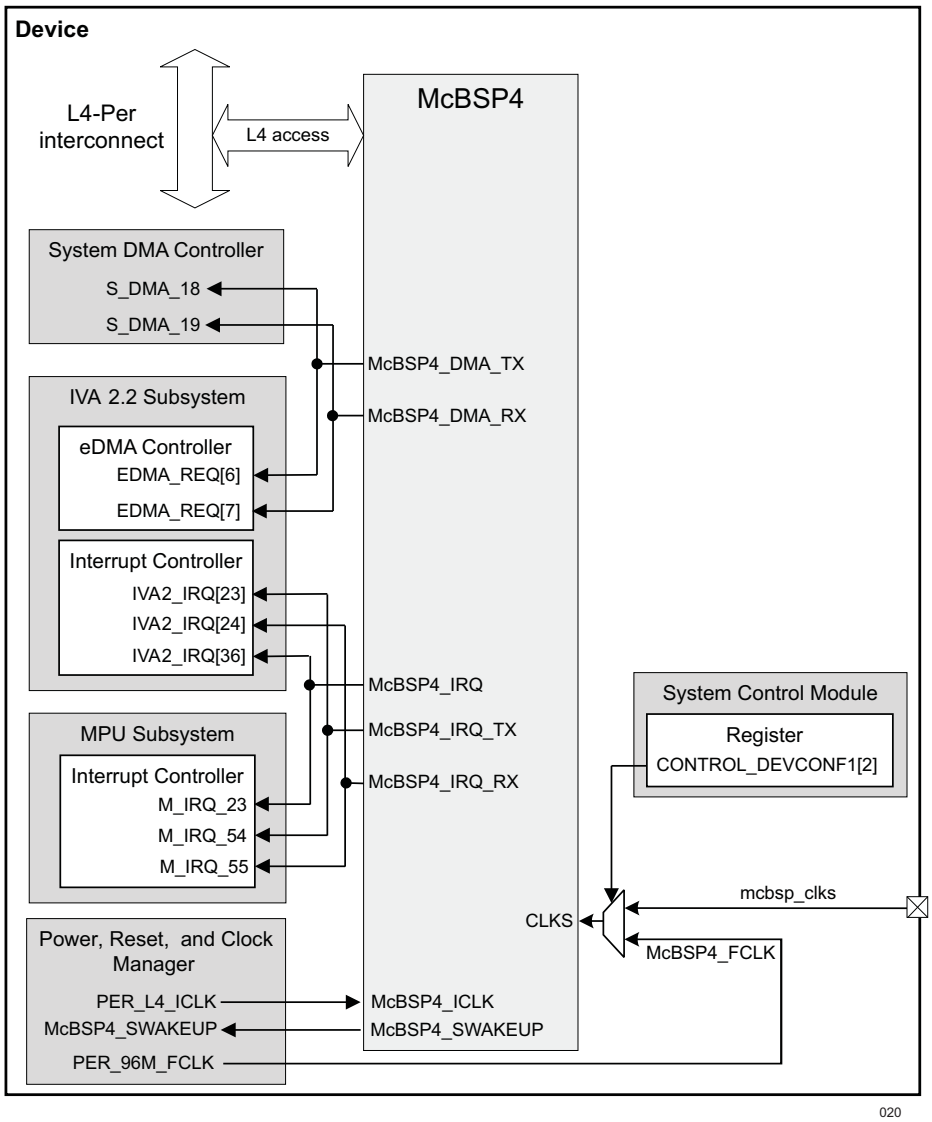
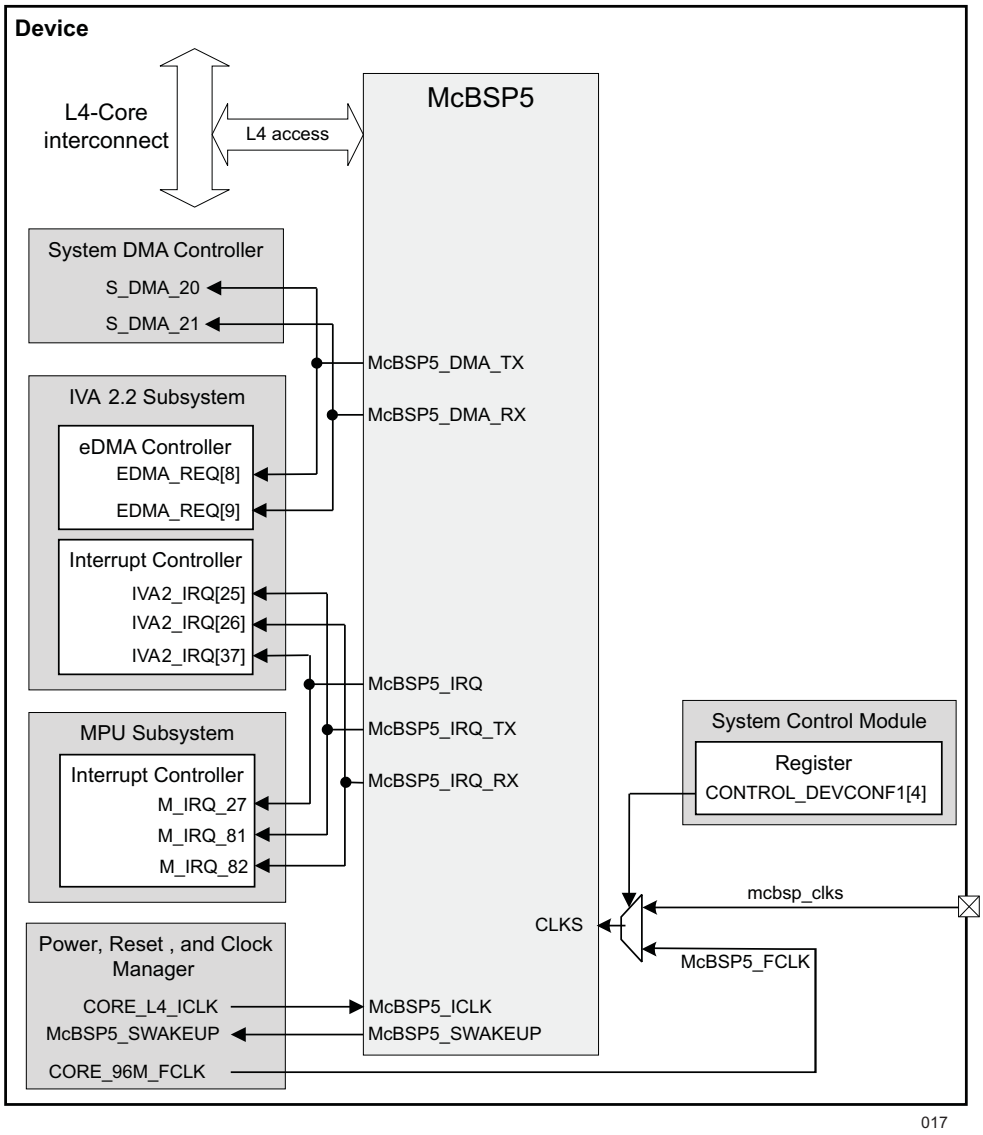


Figure 18-20. McBSP5 Integration



18.3.1 Signal Source Control

The FSR, CLKR and CLKS signals sources are defined by the system control module. The control registers of the system control module are used to select these signals sources.

18.3.1.1 McBSP1 Module (6 Pins Configuration)

The MCBSP1_CLKS bit of the CONTROL.CONTROL_DEVCONF0[2] register is used to select the McBSP1 module CLKS signal source:

- When set to ‘0’, the CLKS source is from the CORE_96M_FCLK
- When set to ‘1’, the CLKS source is from the mcbsp_clks pin

The MCBSP1_CLKR bit of the CONTROL_DEVCONF0[3] register is used to select the McBSP1 module CLKR signal source:

- When set to ‘0’, the CLKR source is from the CLKR input signal
- When set to ‘1’, the CLKR source is from the CLKX input signal

The MCBSP1_FSR bit of the CONTROL_DEVCONF0[4] register is used to select the McBSP1 module FSR signal source:

- When set to ‘0’, the FSR source is from the FSR input signal
- When set to ‘1’, the FSR source is from the FSX input signal

18.3.1.2 McBSP2, 3, 4, and 5 modules (4 pins configuration)

For these McBSPi modules, there are no external mcbspi_clkr and mcbspi_fxr pins (I=2, 3, 4, and 5).

Consequently, the system control module controls only the internal connection of CLKS input signals. The settings are explained in Section 18.3.2.2.2, Section 18.3.2.2.3, Section 18.3.2.2.4, and Section 18.3.2.2.5.

18.3.2 Clocking, Reset, and Power Management Scheme

18.3.2.1 Power Domain

McBSP1 and McBSP5 modules belong to the CORE domain, whereas the McBSP2, McBSP3, and McBSP4 modules belong to the PER (peripheral) domain. This separation of McBSP modules in two power domain allows major part of the device to be switched off while keeping active McBSP2, 3, and 4.

For more information about these power domains, see , *Power Reset and Clock Management*.

18.3.2.2 Clocks

There are two clock domains in the McBSP module:

- Functional clock domain
- Interface clock domain

Table 18-3. Clocking Signals Input to McBSP Module

Type	Name	Source	Description
Interface	McBSPi_ICLK	PER_L4_ICLK (McBSP2, 3, 4) (1)	The L4 interface clock is used for the module internal L4 interconnect slave interface and all depending parts of the Interface clock domain.
		CORE_L4_ICLK (McBSP1, 5) (1)	
Functional	CLKS	PER_96M_FCLK (McBSP2, 3, 4) (1)	McBSP module is running using either a functional clock generated internally (master mode) or supplied from its serial interface (slave mode) for the internal logic. Internal registers select the source of the functional clock and the divider ratio to apply. Functional clock after division in any mode is limited to maximum frequency divided by 2.
		CORE_96M_FCLK (McBSP1, 5) (1)	
		mcbsp_clks (external source common to all McBSP modules)	
Functional	CLKX	mcbspi_clkx (external source)	
Functional	CLKR (McBSP1 only)	mcbsp1_clkr (external source)	

(1) For more information about these sources, see , *Power Reset and Clock Management*.

18.3.2.2.1 McBSP1 Clocks

The McBSP1 module is clocked by a functional clock (CLKS, CLKX, or CLKR) and an interface clock (McBSP1_ICLK).

- The functional clock is used to generate control signals depending on the module internal configuration (see Section 18.4). For McBSP1 module, the functional clock comes from the CLKS signal, CLKX signal, or CLKR signal. The choice between these three clocks is defined by the SCLKME bit of the MCBSP1.MCBSPLP_PCR_REG[7] register and the CLKSM bit of the MCBSP1.MCBSPLP_SRGR2_REG[13] register.

  The CLKS signal of the McBSP1 module is linked to an internal clock (CORE_96M_FCLK) provided by PRCM, whereas the CLKS signal can also be linked to an external signal through the mcbsp_clks pin of the device boundary. The MCBSP1_CLKS bit of the CONTROL.CONTROL_DEVCONF0[2] register is used to select the McBSP1 module CLKS signal source:

  – 0: The CLKS source is from the CORE_96M_FCLK.

– 1: The CLKS source is from the mcbsp_clks pin.

For more information on this register , see Chapter 6, *System Control Module*.

NOTE: When the McBSP1 module does not require the PRCM functional clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP1 bit (PRCM.CM_FCLKEN1_CORE[9]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more details, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off CORE_96M_FCLK clock are met the PRCM automatically launches a **Hardware** handshake protocol to ensure McBSP1 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP1. For more details, see , *Power Reset and Clock Management*.

The CLKX and CLKR signals are connected either by mcbsp1_clkx or mcbsp1_clkr pads. These signals are used like functional clocks by the intermediary of the sample rate generator (SRG).

- The McBSP1_ICLK runs at the L4 core interconnect clock speed. It is used to trigger access to the McBSP1 L4 interface and McBSP1 configuration interface via the MPU/IVA2.2 shared bus. It can also be an input clock for the McBSP sample-rate generator (clock divider), depending on the module configuration (see Section 18.4.3). Its source is the CORE_L4_ICLK signal.

NOTE: When the McBSP1 module does not require the interface clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP1 bit (PRCM.CM_ICLKEN1_CORE[9]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more information, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off CORE_L4_ICLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP1 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP1. For more details, see , *Power Reset and Clock Management*.

It is also possible to activate an autoidle mode for this clock (PRCM.CM_AUTOIDLE1_CORE[9] register AUTO_MCBSP1 bit set to 1). In this case, McBSP1_ICLK follows the CORE_L4 clock domain behavior on the device. For more information, see , *Power Reset and Clock Management*.

18.3.2.2.2 McBSP2 Clocks

The McBSP2 module is clocked by a functional clock (CLKS, CLKX or CLKR) and an interface clock (McBSP2_ICLK).

- The functional clock is used to generate control signals depending on the module internal configuration (see Section 18.4). For McBSP2 module, the functional clock comes from the CLKS signal CLKX signal, or CLKR signal. The choice between these three clocks is defined by the SCLKME bit of the MCBSP2.MCBSPLP_PCR_REG[7] register and the CLKSM bit of the MCBSP2.MCBSPLP_SRGR2_REG[13] register.

  The CLKS signal of the McBSP2 module is linked to an internal clock (PER_96M_FCLK) provided by PRCM, whereas the CLKS signal can also be linked to an external signal through the mcbsp_clks pin of the device boundary. The MCBSP2_CLKS bit of the CONTROL.CONTROL_DEVCONF0[6] register is used to select the McBSP2 module CLKS signal source:

  – 0: The CLKS source is from the PER_96M_FCLK.

  – 1: The CLKS source is from the mcbsp_clks pin.

  For more information, see Chapter 6, *System Control Module*.

NOTE: When the McBSP2 module does not require the functional clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP2 bit (PRCM.CM_FCLKEN_PER[0]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more details, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off PER_96M_FCLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP2 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP2. For more details, see , *Power Reset and Clock Management*.

Only, the CLKX signal is connected by mcbsp2_clkx pads. The CLKR signal is connected to the CLKX signal. These signals are used like functional clocks by the intermediary of the SRG.

- The McBSP2_ICLK runs at the L4 core interconnect clock speed. It is used to trigger access to the McBSP2 L4 interface and McBSP2 configuration interface via the MPU/IVA2.2 shared bus. It can also be an input clock for the McBSP sample-rate generator (clock divider), depending on the module configuration (see Section 18.4.3). Its source is either the PER_L4_ICLK signal.

NOTE: When the McBSP2 module does not require the interface clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP2 bit (PRCM.CM_ICLKEN_PER[0]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more information, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off PER_L4_ICLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP2 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP2. For more details, see , *Power Reset and Clock Management*.

It is also possible to activate an autoidle mode for this clock (PRCM.CM_AUTOIDLE_PER[0] register AUTO_MCBSP2 bit set to 1). In this case, McBSP2_ICLK follows the CORE_L4 clock domain behavior on the device. For more information, see , *Power Reset and Clock Management*.

18.3.2.2.3 McBSP3 Clocks

The McBSP3 module is clocked by a functional clock (CLKS, CLKX or CLKR) and an interface clock (McBSP3_ICLK).

- The functional clock is used to generate control signals depending on the module internal configuration (see Section 18.4). For McBSP3 module, the functional clock comes from the CLKS signal CLKX signal, or CLKR signal. The choice between these three clocks is defined by the SCLKME bit of the MCBSP3.MCBSPLP_PCR_REG[7] register and the CLKSM bit of the MCBSP3.MCBSPLP_SRGR2_REG[13] register.

  The CLKS signal of the McBSP3 module is linked to an internal clock (PER_96M_FCLK) provided by PRCM, whereas the CLKS signal can also be linked to an external signal through the mcbsp_clks pin of the device boundary. The MCBSP3_CLKS bit of the CONTROL.CONTROL_DEVCONF1[0] register is used to select the McBSP3 module CLKS signal source:

  – 0: The CLKS source is from the PER_96M_FCLK.
  – 1: The CLKS source is from the mcbsp_clks pin.

  For more information, see Chapter 6, *System Control Module*.

NOTE: When the McBSP3 module does not require the functional clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP3 bit (PRCM.CM_FCLKEN_PER[1]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more details, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off PER_96M_FCLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP3 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP3. For more details, see , *Power Reset and Clock Management*.

Only, the CLKX signal is connected by mcbsp3_clkx pads. The CLKR signal is connected to the CLKX signal. These signals are used like functional clocks by the intermediary of SRG.

- The McBSP3_ICLK runs at the L4 core interconnect clock speed. It is used to trigger access to the McBSP3 L4 interface and McBSP3 configuration interface via the MPU/IVA2.2 shared bus. It can also be an input clock for the McBSP sample-rate generator (clock divider), depending on the module configuration (see Section 18.4.3). Its source is either the PER_L4_ICLK signal.

NOTE: When the McBSP3 module does not require the interface clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP3 bit (PRCM.CM_ICLKEN_PER[1]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more information, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off PER_L4_ICLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP3 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP3. For more details, see , *Power Reset and Clock Management*.

It is also possible to activate an autoidle mode for this clock (PRCM.CM_AUTOIDLE_PER[1] register AUTO_MCBSP3 bit set to 1). In this case, McBSP3_ICLK follows the PER_L4 clock domain behavior on the device. For more information, see , *Power Reset and Clock Management*.

18.3.2.2.4 McBSP4 Clocks

The McBSP4 module is clocked by a functional clock (CLKS or CLKX) and an interface clock (McBSP4_ICLK).

- The functional clock is used to generate control signals depending on the module internal configuration (see Section 18.4). For McBSP4 module, the functional clock comes from the CLKS signal, the CLKX signal, or the CLKR signal. The choice between these three clocks is defined by the SCLKME bit of the MCBSP4.MCBSPLP_PCR_REG[7] register and the CLKSM bit of the MCBSP4.MCBSPLP_SRGR2_REG[13] register.

  The CLKS signal of the McBSP4 module is linked to an internal clock (PER_96M_FCLK) provided by PRCM, whereas the CLKS signal can also be linked to an external signal through the mcbsp_clks pin of the device boundary. The MCBSP4_CLKS bit of the CONTROL.CONTROL_DEVCONF1[2] register is used to select the McBSP4 module CLKS signal source:

  – 0: The CLKS source is from the PER_96M_FCLK.
  – 1: The CLKS source is from the mcbsp_clks pin.

  For more information, see Chapter 6, *System Control Module*.

NOTE: When the McBSP4 module does not require the functional clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP4 bit (PRCM.CM_FCLKEN_PER[2]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more details, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off PER_96M_FCLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP4 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP4. For more details, see , *Power Reset and Clock Management*.

Only the CLKX signal is connected by mcbsp4_clkx pads. The CLKR signal is connected to the CLKX signal. These signals are used like functional clocks by the intermediary of SRG.

- The McBSP4_ICLK runs at the L4 core interconnect clock speed. It is used to trigger access to the McBSP4 L4 interface and McBSP4 configuration interface via the MPU/IVA2.2 shared bus. It can also be an input clock for the McBSP sample-rate generator (clock divider), depending on the module configuration (see Section 18.4.3). Its source is either the PER_L4_ICLK signal.

NOTE: When the McBSP4 module does not require the interface clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP4 bit (PRCM.CM_ICLKEN_PER[2]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more information, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off PER_L4_ICLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP4 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP4. For more details, see , *Power Reset and Clock Management*.

It is also possible to activate an autoidle mode for this clock (PRCM.CM_AUTOIDLE_PER[2] register AUTO_MCBSP4 bit set to 1). In this case, McBSP4_ICLK follows the PER_L4 clock domain behavior on the device. For more information, see , *Power Reset and Clock Management*.

18.3.2.2.5 McBSP5 Clocks

The McBSP5 module is clocked by a functional clock (CLKS or CLKX) and an interface clock (McBSP5_ICLK).

- The functional clock is used to generate control signals depending on the module internal configuration (see Section 18.4). For McBSP5 module, the functional clock comes from the CLKS signal, the CLKX signal, or the CLKR signal. The choice between these three clocks is defined by the SCLKME bit of the MCBSP5.MCBSPLP_PCR_REG[7] register and the CLKSM bit of the MCBSP5.MCBSPLP_SRGR2_REG[13] register.

  The CLKS signal of the McBSP5 module is linked to an internal clock (CORE_96M_FCLK) provided by PRCM. The CLKS signal can also be linked to an external signal through the mcbsp_clks pin of the device boundary. The MCBSP5_CLKS bit of the CONTROL.CONTROL_DEVCONF1[4] register is used to select the McBSP5 module CLKS signal source:

  – 0: The CLKS source is from the CORE_96M_FCLK.
  – 1: The CLKS source is from the mcbsp_clks pin.

  For more information, see Chapter 6, *System Control Module*.

NOTE: When the McBSP5 module does not require the functional clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP5 bit (PRCM.CM_FCLKEN1_CORE[10]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more details, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off CORE_96M_FCLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP5 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP5. For more details, see , *Power Reset and Clock Management*.

Only, the CLKX signal is connected by mcbsp5_clkx pads. The CLKR signal is connected to the CLKX signal. These signals are used like functional clocks by the intermediary of SRG.

- The McBSP5_ICLK runs at the L4 core interconnect clock speed. It is used to trigger access to the McBSP5 L4 interface and McBSP5 configuration interface via the MPU/IVA2.2 shared bus. It can also be an input clock for the McBSP sample-rate generator (clock divider), depending on the module configuration (see Section 18.4.3). Its source is either the CORE_L4_ICLK signal.

NOTE: When the McBSP5 module does not require the interface clock anymore, the software can disable it at the PRCM level by setting the EN_MCBSP5 bit (PRCM.CM_ICLKEN1_CORE[10]) in the PRCM registers. The clock is effectively cut, provided the other modules that receive it do not require it. For more information, see , *Power Reset and Clock Management*.

At PRCM level, when all the conditions to shut-off CORE_L4_ICLK clock are met the PRCM automatically launches a hardware handshake protocol to ensure McBSP5 is ready to have this clock switched off. Namely, the PRCM asserts an idle request to McBSP5. For more details, see , *Power Reset and Clock Management*.

It is also possible to activate an autoidle mode for this clock (PRCM.CM_AUTOIDLE1_CORE[10] register AUTO_MCBSP5 bit set to 1). In this case, McBSP5_ICLK follows the CORE_L4 clock domain behavior on the device. For more information, see , *Power Reset and Clock Management*.

18.3.2.2.6 SIDETONE Clock

The SIDETONE feature, in the McBSP2 and McBSP3 modules, is clocked only by an interface clock (McBSP2_ICLK or McBSP3_ICLK).

CAUTION

See Section 18.3.2.2.2 and Section 18.3.2.2.3 for information on McBSP2_ICLK and McBSP3_ICLK clocks.

When the SIDETONE feature does not require the clock anymore, the software can disable it at the SIDETONE level by setting the McBSPi.ST_SYSCONFIG_REG[0] AUTOIDLE bit in SIDETONE registers. To conserve power, when SIDETONE feature is not active or there is no activity on SIDETONE feature, the McBSPi_ICLK clock supports an automatic gating that is enabled or disabled by setting the McBSPi.ST_SYSCONFIG_REG[0] AUTOIDLE bit.

- When this bit is asserted (set to 1), the McBSPi_ICLK clock auto-gating is enabled and this clock is disabled internally to the SIDETONE feature, thus reducing power consumption, but not to the McBSP module that contains this feature.
  After reset, the automatic clock gating is enabled; thus, this bit must be disabled by software for activated SIDETONE feature.
- When this bit is set to 0, the McBSPi_ICLK clock auto-gating is disabled and this clock is enabled. The SIDETONE feature can be used normally.

18.3.2.3 Hardware and Software Reset

McBSP1 and McBSP5 modules belong to the CORE domain and their reset signal is the CORE_RST signal from the PRCM module, whereas McBSP2, 3 and 4 modules belong to the PER domain and their reset signal is the PER_RST signal from the PRCM module.

For more details about these signals, see , *Power Reset and Clock Management*.

Table 18-4. Software Reset Signals to All McBSP Modules

Type	Bit Field	Register Source	Activation	Description
Software	SOFTRESET	MCBSPi.MCBSPLP_SYSCONFIG_REG[1]	Active high	McBSP global software reset
	RRST	MCBSPi.MCBSPLP_SPCR1_REG[0]	Active low	This resets and disables the receiver, including the RB.
	XRST	MCBSPi.MCBSPLP_SPCR2_REG[0]		This resets and disables the transmitter, including the XB.
	GRST	MCBSPi.MCBSPLP_SPCR2_REG[6]		SRG is reset
	FRST	MCBSPi.MCBSPLP_SPCR2_REG[7]		Frame-sync logic is reset. Frame-sync generated signal is not generated by the SRG

For more details about these signals, see Section 18.6. See Section 18.5.1.1, for a complete description of the McBSP initialization procedure.

18.3.2.4 Power Management

18.3.2.4.1 McBSP Operating States

Two operating states are defined for all the McBSP modules:

- ACTIVE state: The module is running synchronously on the interface and functional clock, interrupts and DMA requests can be generated according to the configuration (register, master or slave mode, etc) and the external signals.
- IDLE state: As part of the system power management, the PRCM module can request the McBSP modules to enter IDLE state. Depending on the configured acknowledgment mode: Force Idle, No Idle and Smart Idle modes, a McBSP module will effectively enter IDLE state or not. As soon as a McBSP module enters IDLE state, it doesn't have anymore activities except those unrelated to clock activity (for example wake-up features) and its clocks are likely to be switched off at PRCM level.

  When the McBSP goes into IDLE state, the McBSP internal state-machine clock switches from interface clock (L4_ICLK) to external serial clock (because OMAP is supposed to shut down all internal clocks). Then the McBSP can continue to process during IDLE state with the external clock provided by the external component/AUDIOBUFFER.

  The McBSP can exit IDLE state only if the external serial clock is active. After exiting IDLE state, the McBSP state-machine clock is provided by the OMAP interface clock (L4_ICLK).

NOTE: Idle request and idle acknowledge are only internal signals, with no means to observe or to control. The signals generation and control is purely hardware (managed automatically by the PRCM and the McBSP module depending on the SIDLEMODE settings).

18.3.2.4.2 McBSP Acknowledgment Modes

During initialization or configuration of the McBSP module, the software must configure how the McBSP module will answer an IDLE solicitation from the PRCM module (that is, the way idle acknowledge will be asserted following an idle request assertion).

Each McBSP module can be configured via the MCBSPi.MCBSPLP_SYSCONFIG_REG[4:3] SIDLEMODE field as one of the following acknowledgment modes:

- Force Idle mode (SIDLEMODE bit = 0x0): An idle request is acknowledged unconditionally, regardless of the internal state of the module. The McBSP module immediately enters Idle state (no activity),

interface and PRCM functional clock can be stopped, no interrupts and DMA requests can be generated. In this mode, the McBSP module freezes all the internal activity when the PRCM clocks are switched off by the PRCM module, leading to a potential loss of data.

CAUTION

In Force Idle mode, the wake-up feature is inhibited.

CAUTION

If the McBSP functional part, transmitter or receiver, is running within this period of time (the functional clock source is not the PRCM functional clock), the internal state of the McBSP module will not be idle (FSM states, processes, etc.), and when the McBSP module exits from the Force Idle state unexpected behavior may happen in both receiver and transmitter. To avoid this, both receive and transmit parts, must be disabled by software prior to idle request assertion (all functional clock external sources must be disabled).

- No Idle mode (SIDLEMODE bit = 0x1): An idle request is never acknowledged, meaning it will prevent the PRCM from switching off its related clocks and from putting in a lower power state than the power domain it belongs to. The McBSP module never enters Idle state (is active).
- Smart Idle mode (SIDLEMODE bit = 0x2): Acknowledgment to an idle request is given based on the internal activity of the McBSP module. The McBSP module is in a waiting state, interface / functional clocks can be stopped, no interrupt can be generated, a wake-up signal can be generated according to the configuration (See Section 18.3.2.4.4) and external signals.

NOTE: The value MCBSPi.MCBSPLP_SYSCONFIG_REG[4:3] SIDLEMODE field = 0x3 must not be used.

When configured in Smart Idle mode, McBSP module also offers an additional granularity on McBSPi_FCLK and McBSPi_ICLK gating. McBSPi.MCBSPLP_SYSCONFIG_REG[9:8] CLOCKACTIVITY bit field is used to determine which clock will be shut down (McBSPi_FCLK, McBSPi_ICLK, none of them or both of them).

CLOCKACTIVITY setting is used in the McBSP module to determine on which part of the module the conditions to acknowledge the PRCM idle request will be tested. As an example, if McBSPi_FCLK is said not to be shut down upon a PRCM idle request, this means McBSP module will only consider McBSPi_ICLK and the associated pending activities before acknowledging the request.

NOTE: Some of McBSP features are associated to McBSPi_ICLK and others to McBSPi_FCLK. Using the CLOCKACTIVITY along with the Smart Idle mode ensures that the features associated with the clock that will remain active are always enabled, even if McBSP has acknowledged an idle request. For more information, see Section 18.3.2.4.4.

Table 18-5 lists the value of the bit field and indicates if the interface (McBSPi_ICLK) and PRCM functional (McBSPi_FCLK) clocks can be switched-off or not when an idle request is received by McBSP module.

Table 18-5. State of Clocks When the Module is in Idle State

CLOCKACTIVITY Value	Interface Clock (McBSPi_ICLK)	PRCM Functional Clock (CORE_96M_FCLK or PER_96M_FCLK)
0b00	OFF	OFF
0b01	OFF	ON
0b10	ON	OFF
0b11	ON	ON

NOTE: OFF means this clock can be switched-off.

ON means this clock must be maintained during wake up period.

CAUTION

The PRCM module does not have any hardware mean to read CLOCKACTIVITY settings. It is thus software responsibility to ensure a consistent programming between McBSPi.MCBSPLP_SYSCONFIG_REG[9:8] CLOCKACTIVITY bit field and PRCM 96M_FCLK and L4_ICLK control bits (for more information about clocks, see Section 18.3.2.2, *Clocks*). If McBSP module is disabled in both CM_FCLKEN and CM_ICLKEN PRCM registers while CLOCKACTIVITY is set to 11, nothing prevents the PRCM module from asserting its idle request which will be acknowledged regardless of the features associated with the McBSP clocks. This may lead to unpredictable behaviors.

The software can disable all clocks at the McBSP module level by setting the IDLE_EN bit in McBSPi.MCBSPLP_PCR_REG[14] registers. The IDLE_EN bit allows stopping all the clocks in the McBSP module (legacy):

- When set to ‘0’, the McBSP module is running
- When set to ‘1’, the clocks in the McBSP module are shut off when both IDLE_EN =1 and his power domain is in idle mode.

18.3.2.4.3 Wake-Up Capability

When configured in Smart Idle mode, the sources for wake-up generation are a subset of the interrupt sources. The wake up sources are enabled by setting the McBSPi.MCBSPLP_SYSCONFIG_REG[2] ENAWAKEUP bit (wake up feature control):

- Set to ‘0’, wake up capability is disabled
- Set to ‘1’, wake up capability is enabled

The McBSPi_SWAKEUP signal is the McBSP module asynchronous wake-up signal sent to the PRCM module when a wake-up generation is requested.

The wake up configurations are defined by setting the corresponding bits in the McBSPi.MCBSPLP_WAKEUPEN_REG register.

18.3.2.4.3.1 Receive Wake-up

There are 4 receive possible wake up configurations:

- McBSPi.MCBSPLP_WAKEUPEN_REG[3] RRDYEN bit: The McBSP module asserts the McBSPi_SWAKEUP request when the receive buffer reaches the high threshold value (RTHRESHOLD value + 1) of the McBSPi.MCBSPLP_THRSH1_REG register. If the McBSPi.MCBSPLP_IRQENABLE_REG[3] RRDYEN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting from idle mode (interrupt will be asserted once the McBSPi.MCBSPLP_IRQSTATUS_REG[3] RRDY bit changes from ‘0’ to ‘1’, indicating that received data is ready to be read).
- McBSPi.MCBSPLP_WAKEUPEN_REG[2] REOFEN bit: The McBSP module asserts the McBSPi_SWAKEUP request at the end of the frame. If the McBSPi.MCBSPLP_IRQENABLE_REG[2] REOFEN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting idle mode.
- McBSPi.MCBSPLP_WAKEUPEN_REG[1] RFSREN bit: The McBSP module sends a McBSPi_SWAKEUP request to the PRCM module when a receive frame-sync pulse is detected while the McBSP module is in idle mode. If the McBSPi.MCBSPLP_IRQENABLE_REG[1] RFSREN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting idle mode.

- McBSPi.MCBSPLP_WAKEUPEN_REG[0] RSYNCERREN bit: The McBSP module asserts the McBSPi_SWAKEUP request when an unexpected receive frame-sync pulse is detected. If the McBSPi.MCBSPLP_IRQENABLE_REG[0] RSYNCERREN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request o the MPU or IVA 2.2 subsystems when exiting from idle mode (interrupt is asserted once the McBSPi.MCBSPLP_IRQSTATUS_REG[0] RSYNCERR bit changes from ‘0’ to ‘1’, indicating that a receive error occurred).

18.3.2.4.3.2 Transmit Wakeup

For transmit, there are also 5 possible wake-up configuration scenarios:

- McBSPi.MCBSPLP_WAKEUPEN_REG[14] XEMPTYEOFEN bit: The McBSP module asserts the McBSPi_SWAKEUP request when a complete frame was transmitted and the transmit buffer is empty. If the McBSPi.MCBSPLP_IRQENABLE_REG[14] XEMPTYEOFEN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting from idle mode.
- McBSPi.MCBSPLP_WAKEUPEN_REG[10] XRDYEN bit: The McBSP module asserts the McBSPi_SWAKEUP request when the transmit buffer reaches the high threshold value (XTHRESHOLD value + 1) of the McBSPi.MCBSPLP_THRSH2_REG register. If the McBSPi.MCBSPLP_IRQENABLE_REG[10] XRDYEN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting from idle mode (interrupt is asserted once the McBSPi.MCBSPLP_IRQSTATUS_REG[10] XRDY bit changes from ‘0’ to ‘1’, indicating that transmit buffer data is ready to accept new data).
- McBSPi.MCBSPLP_WAKEUPEN_REG[9] XEOFEN bit: The McBSP module asserts the McBSPi_SWAKEUP request at the end of the frame. If the McBSPi.MCBSPLP_IRQENABLE_REG[9] XEOFEN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting from idle mode.
- McBSPi.MCBSPLP_WAKEUPEN_REG[8] XFSXEN bit: The McBSP module sends a McBSPi_SWAKEUP request when a transmit frame-sync pulse is detected while the module is in idle mode. If the McBSPi.MCBSPLP_IRQENABLE_REG[8] XFSXEN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting from idle mode.
- McBSPi.MCBSPLP_WAKEUPEN_REG[7] XSYNCERREN bit: The McBSP module asserts the McBSPi_SWAKEUP request when an unexpected transmits frame-sync pulse is detected. If the McBSPi.MCBSPLP_IRQENABLE_REG[7] XSYNCERREN bit is set to 1, the McBSP module sends an interrupt (McBSPi_IRQ) request to the MPU or IVA 2.2 subsystems when exiting from idle mode (interrupt will be asserted once the McBSPi.MCBSPLP_IRQSTATUS_REG[7] XSYNCERR bit changes from ‘0’ to ‘1’, indicating that a transmit error occurred).

18.3.2.4.3.3 Notes

When mcbsp1_fsr/mcbspi_fsx pins is configured as an output, the FSR/FSX wake-up generation makes no sense (the module cannot be in Smart Idle mode).

Detection of RSYNCERR/XSYNCERR during idle mode can be used only when mcbsp1_fsr/mcbspi_fsx pins is configured as an input and the remote system knows to assert such an error to trigger the wake up of the McBSP module.

The module does not implement interrupt request (IRQ) assertion when configured as (GPIO). Pins that can be used to accept input signals and/or send output signals but are not linked to specific uses); also a wake up capability in this mode is not available.

18.3.2.4.4 Analysis of the Receiver Smart Idle Behavior

The analysis of the power mode behavior is shown in Table 18-6 :

In this table, the CLKRM bit is in the McBSPi.MCBSPLP_PCR_REG register on position 8, CLKXM bit in the McBSPi.MCBSPLP_PCR_REG[9] register, and CLOCKACTIVITY bit in the McBSPi.MCBSPLP_SYSCONFIG_REG[9:8] register.

The value X signifies that the bit value is not significant.

Table 18-6. McBSP Smart Idle Mode Configuration Behavior

CLKRM Bit	CLKXM Bit	McBSP Mode	Source of Functional Clock	CLOCKACTIVITY Bit	Behavior
0	0	Slave	outside	0bXX	The module acknowledges the idle request as soon as there is no pending DMA, interrupt request or transmit buffer threshold synchronization (only when wake-up event is set on transmit threshold reached), regardless of the CLOCKACTIVITY settings or receive and transmit activity.
0	1	Transmit Master	McBSPi_ICLK	0b0X	The McBSP will not acknowledge the idle request unless: • The transmit part is disabled (XDISABLE) or under software reset (XRST) and the receive part is not using the transmit loop-back clock (CLKR is not connected to the CLKX input pin). • Both transmit and receive parts are disabled (XDISABLE/RDISABLE) or under software reset (XRST/RRST) The idle acknowledge is asserted as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached) and the pending transmit and/or receive frames were completed in case of transmit and/or receive disable.
			CLKS	0bX0	
			McBSPi_ICLK	0b1X	The module acknowledges the idle request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached).
			CLKS	0bX1	
			CLKR (outside)	0bXX	The module acknowledges the idle request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached), regardless of the CLOCKACTIVITY settings.
1	0	Receive master	McBSPi_ICLK	0b0X	The McBSP will not acknowledge the idle request unless: The receive part is disabled (RDISABLE) or under software reset (RRST) The Idle acknowledge is asserted as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached) and the pending transmit and/or receive frames where completed in case of transmit and/or receive disable.
			CLKS	0bX0	
			McBSPi_ICLK	0b1X	The module acknowledges the idle request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached).
			CLKS	0bX1	
			CLKX	0bXX	When CLKX is used as source (functional clock is provided from outside) then the module acknowledges the idle request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached), regardless of the CLOCKACTIVITY settings.

Table 18-6. McBSP Smart Idle Mode Configuration Behavior (continued)

CLKRM Bit	CLKXM Bit	McBSP Mode	Source of Functional Clock	CLOCKACTIVITY Bit	Behavior
1	1	Transmit and Receive Master	McBSPi_ICLK	0b0X	The McBSP will not acknowledge the idle request unless: Both transmit and receive parts are disabled (XDISABLE/RDISABLE) or under software reset (XRST/RRST) The idle acknowledge is asserted as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached) and the pending transmit and/or receive frames were completed in case of transmit and/or receive disable. Note that no wake-up event is available in this mode since the entire McBSP and remote device activity is frozen.
			CLKS	0bX0	
			McBSPi_ICLK	0b1X	The module acknowledges the idle request as soon as there is no pending DMA, interrupt request or transmit/receive buffer threshold synchronization (only when wake-up event is set on transmit/receive threshold reached).
			CLKS	0bX1	

NOTE: The RFSREN/XFSXEN mode is suitable for wake-up generation when both clocks (PRCM functional and interface) are switched off and mcbsp1_fsr/mcbspi_fsx pins is configured as input. The frame-sync pulse is asynchronously detected during idle.

The RSYNCERREN/XSYNCERREN mode can be used to wake-up the McBSP module only by a remote module implementing such a feature, to trigger a wake up. This mode requires functional clock to be active.

18.3.3 Hardware Requests

18.3.3.1 DMA Requests

The DMA requests are shared between the IVA2.2 subsystem DMA controller (eDMA) and system DMA controller (sDMA). Each of the five McBSP modules can generate two DMA events:

- McBSPi_DMA_TX : McBSPi module transmit request
- McBSPi_DMA_RX : McBSPi module receive request

The following table summarizes the DMA events with the mapping on both DMA controllers.

Table 18-7. McBSP DMA Requests

Request Name	Mapping	Destination	Description
McBSP1_DMA_TX	EDMA_REQ[0] S_DMA_30	IVA2.2 subsystem DMA controller system DMA controller	write (or transmit) request
McBSP1_DMA_RX	EDMA_REQ[1] S_DMA_31	IVA2.2 subsystem DMA controller system DMA controller	read (or receive) request
McBSP2_DMA_TX	EDMA_REQ[2] S_DMA_32	IVA2.2 subsystem DMA controller system DMA controller	write (or transmit) request
McBSP2_DMA_RX	EDMA_REQ[3] S_DMA_33	IVA2.2 subsystem DMA controller system DMA controller	read (or receive) request
McBSP3_DMA_TX	EDMA_REQ[4] S_DMA_16	IVA2.2 subsystem DMA controller system DMA controller	write (or transmit) request
McBSP3_DMA_RX	EDMA_REQ[5] S_DMA_17	IVA2.2 subsystem DMA controller system DMA controller	read (or receive) request
McBSP4_DMA_TX	EDMA_REQ[6] S_DMA_18	IVA2.2 subsystem DMA controller system DMA controller	write (or transmit) request
McBSP4_DMA_RX	EDMA_REQ[7] S_DMA_19	IVA2.2 subsystem DMA controller system DMA controller	read (or receive) request
McBSP5_DMA_TX	EDMA_REQ[8] S_DMA_20	IVA2.2 subsystem DMA controller system DMA controller	write (or transmit) request
McBSP5_DMA_RX	EDMA_REQ[9] S_DMA_21	IVA2.2 subsystem DMA controller system DMA controller	read (or receive) request

18.3.3.2 Interrupt Requests

18.3.3.2.1 McBSP Interrupt Requests

Each of the five McBSP modules can generate three interrupts, shared between the MPU subsystem and IVA2.2 subsystem interrupt controllers:

- McBSPi_IRQ: McBSPi module common interrupt request
- McBSPi_IRQ_TX: McBSPi module transmit interrupt request
- McBSPi_IRQ_RX: McBSPi module receive interrupt request

The following tables list the interrupt requests with the mapping.

Table 18-8. McBSP Common Interrupt Requests

Request Name	Mapping	Destination
McBSP1_IRQ	IVA2_IRQ[33] M_IRQ_16	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP2_IRQ	IVA2_IRQ[34] M_IRQ_17	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP3_IRQ	IVA2_IRQ[35] M_IRQ_22	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP4_IRQ	IVA2_IRQ[36] M_IRQ_23	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller

Table 18-8. McBSP Common Interrupt Requests (continued)

Request Name	Mapping	Destination
McBSP5_IRQ	IVA2_IRQ[37] M_IRQ_27	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller

Table 18-9. McBSP Transmit Interrupt Requests

Request Name	Mapping	Destination
McBSP1_IRQ_TX	IVA2_IRQ[16] M_IRQ_59	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP2_IRQ_TX	IVA2_IRQ[19] M_IRQ_62	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP3_IRQ_TX	IVA2_IRQ[21] M_IRQ_89	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP4_IRQ_TX	IVA2_IRQ[23] M_IRQ_54	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP5_IRQ_TX	IVA2_IRQ[25] M_IRQ_81	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller

Table 18-10. McBSP Receive Interrupt Requests

Request Name	Mapping	Destination
McBSP1_IRQ_RX	IVA2_IRQ[17] M_IRQ_60	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP2_IRQ_RX	IVA2_IRQ[20] M_IRQ_63	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP3_IRQ_RX	IVA2_IRQ[22] M_IRQ_90	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP4_IRQ_RX	IVA2_IRQ[24] M_IRQ_55	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller
McBSP5_IRQ_RX	IVA2_IRQ[26] M_IRQ_82	IVA2.2 subsystem interrupt controller MPU subsystem interrupt controller

An event can generate an interrupt request when the corresponding mask bit in the McBSPi.MCBSPLP_IRQENABLE_REG register is set to '1'.

Table 18-11 and Table 18-12 summarize the events causing the generation of an interrupt request.

Table 18-11. McBSP Transmit Interrupt Events

Event Name	Status Bit	Mask Bit	Description
Transmit buffer empty at end of frame	McBSPi.MCBSPLP_IRQSTATUS_REG[14] XEMPTYEOF	McBSPi.MCBSPLP_IRQENABLE_REG [14] XEMPTYEOFEN	This event happens when a complete frame was transmitted and the transmit buffer is empty .
Overflow	McBSPi.MCBSPLP_IRQSTATUS_REG[12] XOVFLSTAT	McBSPi.MCBSPLP_IRQENABLE_REG [12] XOVFLEN	This event happens when transmit data buffer is full and a new data is written in this buffer. The new data written is discarded.
Underflow	McBSPi.MCBSPLP_IRQSTATUS_REG[11] XUNDFLSTAT	McBSPi.MCBSPLP_IRQENABLE_REG [11] XUNDFLEN	This event happens when transmit data buffer is empty and a new data is required to be transmitted.
Threshold reached	McBSPi.MCBSPLP_IRQSTATUS_REG[10] XRDY	McBSPi.MCBSPLP_IRQENABLE_REG [10] XRDYEN	This event happens when the transmit buffer free locations are equal or above the THRSH2_REG value.
End of Frame	McBSPi.MCBSPLP_IRQSTATUS_REG[9] XEOF	McBSPi.MCBSPLP_IRQENABLE_REG [9] XEOFLEN	This event happens when a complete frame was transmitted.

Table 18-11. McBSP Transmit Interrupt Events (continued)

Event Name	Status Bit	Mask Bit	Description
Frame-sync	McBSPi.MCBSPLP_IRQSTATUS_REG[8] XFSX	McBSPi.MCBSPLP_IRQENABLE_REG [8] XFSXEN	This event happens when a new transmit frame synchronization is asserted.
Frame-sync error	McBSPi.MCBSPLP_IRQSTATUS_REG[7] XSYNCERR	McBSPi.MCBSPLP_IRQENABLE_REG [7] XSYNCERREN	This event happens when a transmit frame synchronization error is detected.

Table 18-12. McBSP Receive Interrupt Events

Event Name	Status bit	Mask bit	Description
Overflow	McBSPi.MCBSPLP_IRQSTATUS_REG[5] ROVFLSTAT	McBSPi.MCBSPLP_IRQENABLE_REG[5] ROVFLEN	This event happens when receive data buffer is full and a new data is written in this buffer. The new data written is discarded.
Underflow	McBSPi.MCBSPLP_IRQSTATUS_REG[4] XUNDFLSTAT	McBSPi.MCBSPLP_IRQENABLE_REG[4] RUNDFLEN	This event happens when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined.
Threshold reached	McBSPi.MCBSPLP_RQSTATUS_REG[3] RRDY	McBSPi.MCBSPLP_IRQENABLE_REG[3] RRDYEN	This event happens when the receive buffer occupied locations are equal or above the THRSH1_REG value.
End of Frame	McBSPi.MCBSPLP_IRQSTATUS_REG[2] REOF	McBSPi.MCBSPLP_IRQENABLE_REG[2] REOFLEN	This event happens when a complete frame was received.
Frame-sync	McBSPi.MCBSPLP_IRQSTATUS_REG[1] RFSR	McBSPi.MCBSPLP_IRQENABLE_REG[1] RFSREN	This event happens when a new receive frame synchronization is asserted.
Frame-sync error	McBSPi.MCBSPLP_IRQSTATUS_REG[0] RSYNCERR	McBSPi.MCBSPLP_IRQENABLE_REG[0] RSYNCERREN	This event happens when a receive frame synchronization error is detected.

Once an interrupt request is generated, the software must read the McBSPi.MCBSPLP_IRQSTATUS_REG register to check what event has caused the interrupt request generation, and acknowledge each processed event by writing a 1 to the corresponding bit in the McBSPi.MCBSPLP_IRQSTATUS_REG register.

NOTE: All Status bits can be cleared in two ways:

- If the corresponding mask bit is set to '1' (interrupt generation enabled), the status bit is cleared by writing a '1'.
- If the corresponding mask bit is cleared to '0' (interrupt generation disabled), the status bit is cleared when a new start or stop condition is detected on the receiver/transmitter.

18.3.3.2.2 SIDETONE_McBSP Interrupt Requests

The McBSP2 and McBSP3 can generate another interrupt to the MPU subsystem interrupt controller:

- **ST_McBSPi_IRQ:** SIDETONE interrupt request for McBSPi module (where I = 2, 3).

Table 18-13. SIDETONE_McBSP Interrupt Requests

Request Name	Mapping	Destination
ST_McBSP2_IRQ	M_IRQ_4	MPU subsystem interrupt controller
ST_McBSP3_IRQ	M_IRQ_5	MPU subsystem interrupt controller

Table 18-14. SIDETONE_McBSP Interrupt Events

Event Name	Status Bit	Mask Bit	Description
Over-run	McBSPi.ST_IRQSTATUS_REG[0] OVRRERROR	McBSPi.ST_IRQENABLE_REG[0] OVRRERROREN	This event happens when a new data to be processed arrives before the previous one has ended.

Once an interrupt request has been generated, the software must read the McBSPi.ST_IRQSTATUS_REG register to check what event caused the interrupt request generation, and acknowledge each processed event by writing a 1 to the corresponding bit in the McBSPi.ST_IRQSTATUS_REG register.

NOTE: The McBSPi.ST_IRQSTATUS_REG[0] OVRRERROR status bit can be cleared in two ways:

- If the McBSPi.ST_IRQENABLE_REG[0] OVRRERROREN mask bit is set to '1' (interrupt generation enabled), the status bit is cleared by writing a '1'.
- If the McBSPi.ST_IRQENABLE_REG[0] OVRRERROREN mask bit is cleared to '0' (interrupt generation disabled), the status bit is cleared when a new start or stop condition is detected on the SIDETONE channels.

18.4 McBSP Functional Description

This section is a functional description of the McBSP module.

18.4.1 Block Diagram

Figure 18-21, Figure 18-22, and Figure 18-23 show functional block diagrams of the five instances of the McBSP modules.

These figures regroup the McBSP modules by categories:

- McBSP module without audio buffer and SIDETONE core: McBSP1, McBSP4 and McBSP5
- McBSP module with audio buffer and SIDETONE core: McBSP2
- McBSP module without audio buffer, but with SIDETONE core: McBSP3

Figure 18-21. McBSP1, McBSP4 and McBSP5 Block Diagrams

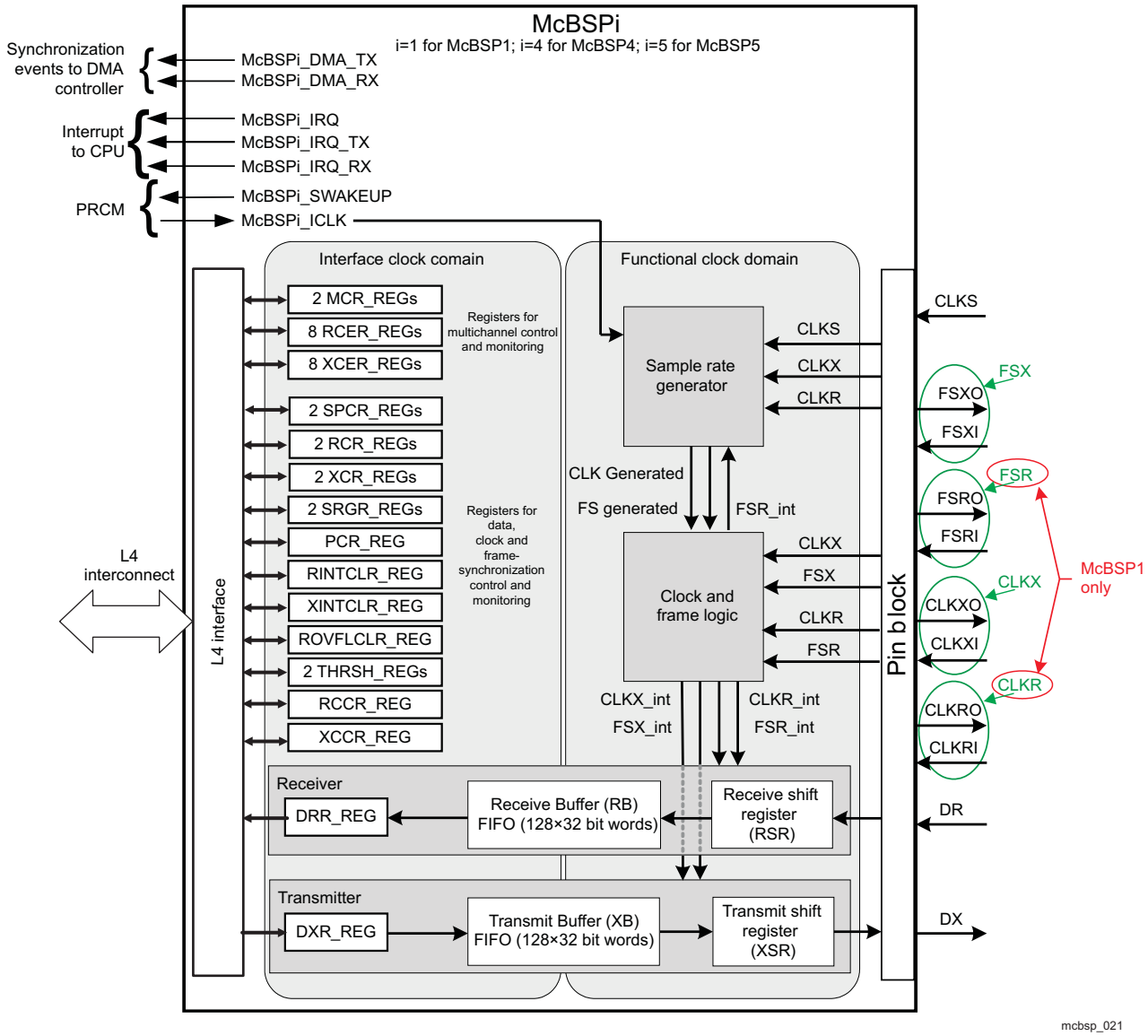


Figure 18-22. McBSP2 Block Diagram

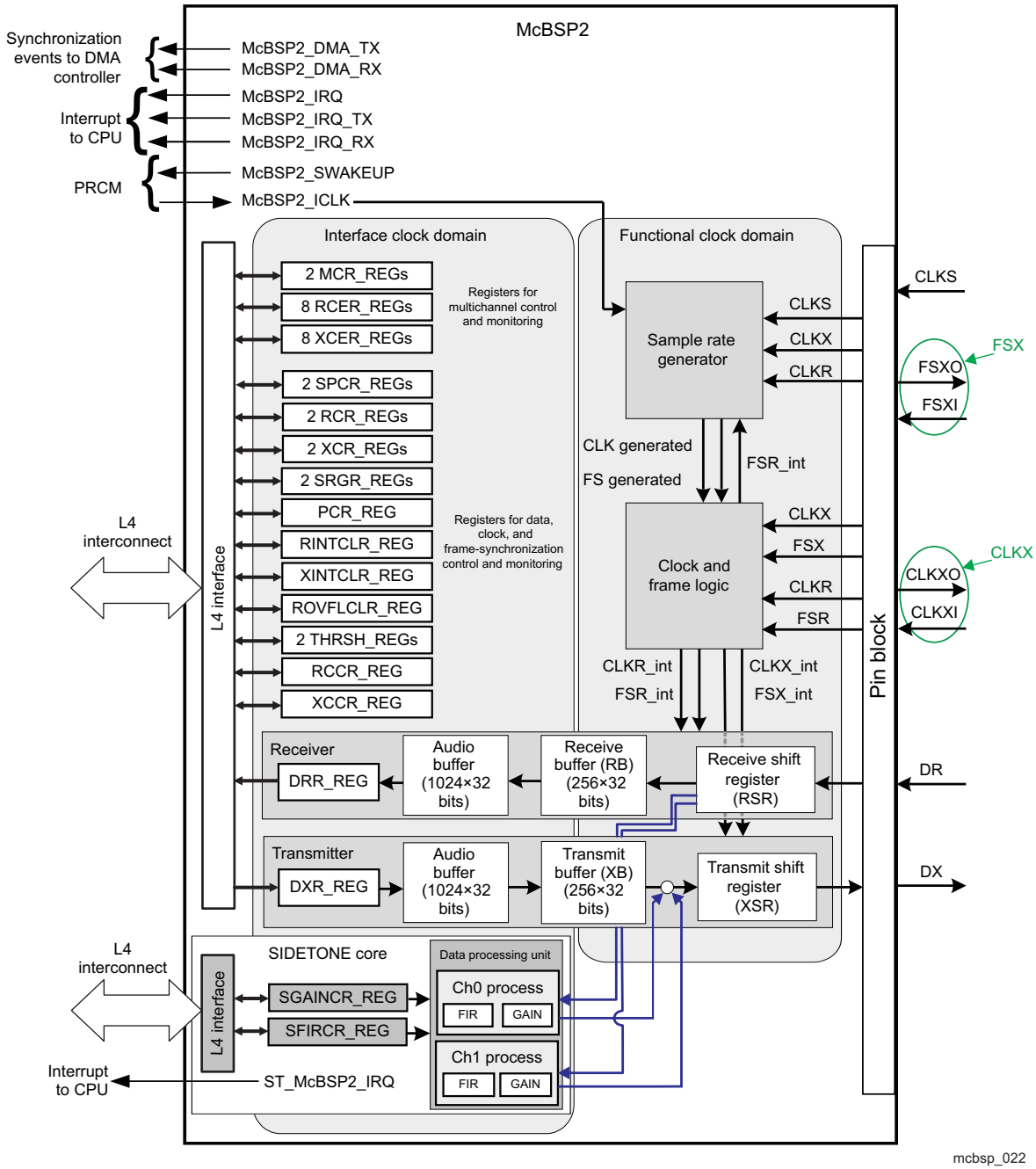
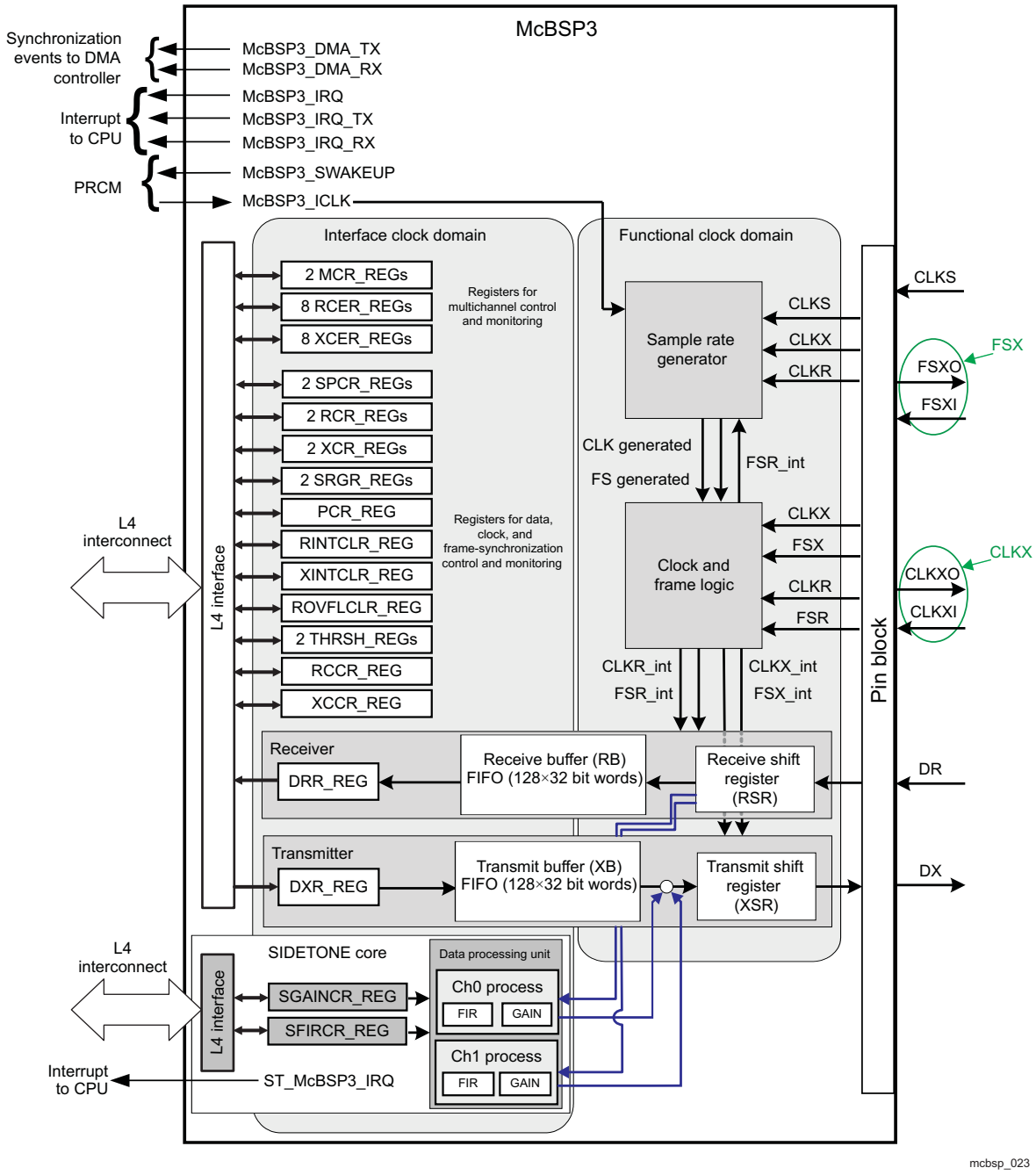


Figure 18-23. McBSP3 Block Diagram



18.4.2 McBSP Data Transfer Process

For McBSP1, McBSP3, McBSP4, and McBSP5 modules, receive and transmit operations are triple-buffered (512 bytes buffers organized in 32-bit words are used).

For the McBSP2 module, receive and transmit operations are quadruple-buffered (5K bytes buffers organized in 32-bit words are used). Receive and transmit buffers are separated in two memories: The same buffer as the others McBSP modules (1K bytes) and an audio buffer (4K bytes). The audio buffer is the largest one and is clocked by the interface clock, while the other buffer is used for synchronizing data to/from the audio buffer with the functional transmit/receive clock domains. Figure 18-25 shows this implementation in the McBSP2 data transfer paths.

All registers of McBSP data transfer paths are 32-bits wide. Figure 18-24 and Figure 18-25 illustrate the McBSP data transfer paths.

CAUTION

The McBSP registers (DRR_REG and DXR_REG) are limited to 32-bit data accesses (L4 Interconnect). 16-bit and 8-bit is not allowed and can corrupt register content.

Figure 18-24. McBSP Data Transfer Paths

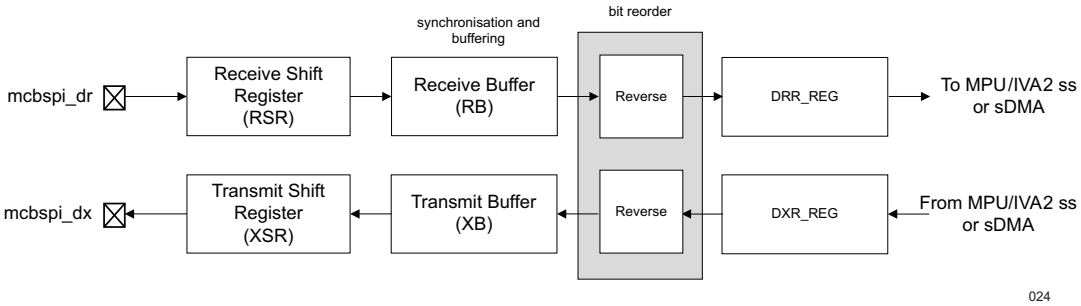
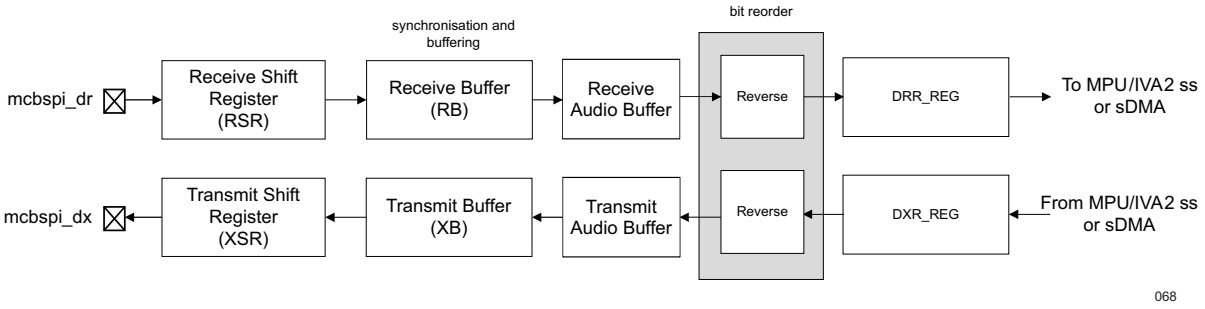


Figure 18-25. McBSP2 Data Transfer Paths



18.4.2.1 Data Transfer Process for 8- / 12- / 16- / 20- / 24- / 32-bits Long Words

CAUTION

For each data word length, one data occupies one 32-bit buffer word.

Receive data arrives on the mcbspi_dr pin and is shifted into the receive shift register (RSR). When a full word (depending on the data length configuration) is received, the content of the shift register is copied into the receive buffer (RB) if it is not full. When the RB threshold is reached the McBSP module asserts DMA or interrupt request and the RB content is then transferred (the sDMA or the eDMA controller reads the data receive register McBSPi.MCBSPLP_DRR_REG).

Transmit data is written by the MPU subsystem, the IVA2.2 subsystem or the DMA controller to data transmit register (McBSPi.MCBSPLP_DXR_REG) using the McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit enable input (when a byte is not enabled, the byte value in the memory will contain the previous written value). If there is no previous data in transmit shift register (XSR), the value from the transmit buffer (XB) is copied to XSR; otherwise, the content is copied to the XSR when the last bit of the previous data is shifted out on the mcbspi_dx pin.

18.4.2.2 Bit Reordering (Option to Transfer LSB First)

Generally, the McBSP module transmits or receives all data with the MSB first. However, some data protocols require the LSB to be transferred first.

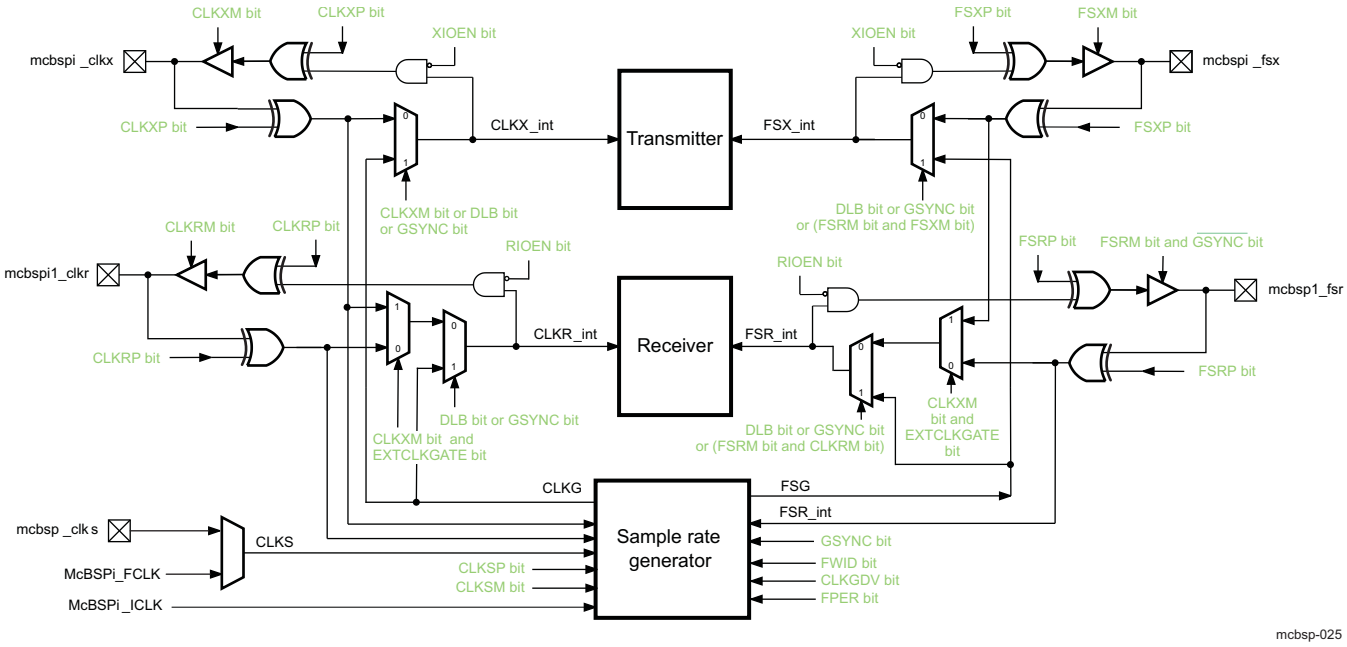
If you set McBSPi.MCBSPLP_XCR2_REG[4:3] XREVERSE=0b01, the bit ordering of the data words is reversed (LSB first) before being sent to the serial port. If you set McBSPi.MCBSPLP_RCR2_REG[4:3] RREVERSE=0b01, the bit ordering of the data words is reversed during reception (LSB first).

This feature is available for all the data formats from 8 up to 32-bit data length.

18.4.2.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

Figure 18-26. Conceptual Block Diagram for Clock and Frame Generation When MCBSPLP_SPCR1_REG[15] ALB = 0 and CONTROL_DEVCONF0[3] MCBSP1_CLKR = 0



The CLKR_int clock signal can be derived from three sources:

- The sample rate generator (CLKG): When MCBSPLP_XCCR_REG[5] DLB = 1 or MCBSPLP_SRGR2_REG[15] GSYNC = 1
- mcbsp1_clkr pin (CLKR): When MCBSPLP_PCR_REG[9] CLKXM = 0 and MCBSPLP_XCCR_REG[15] EXTCLKGATE = 0, and MCBSPLP_XCCR_REG[5] DLB = 0 or MCBSPLP_SRGR2_REG[15] GSYNC = 0
- mcbspi_clkx pin (CLKX): When MCBSPLP_PCR_REG[9] CLKXM = 1 and MCBSPLP_XCCR_REG[15] EXTCLKGATE = 1, and MCBSPLP_XCCR_REG[5]DLB = 0 or MCBSPLP_SRGR2_REG[15] GSYNC = 0

The CLKX_int clock signal can be derived from two sources:

- The sample rate generator (CLKG): When MCBSPLP_PCR_REG[9] CLKXM = 1, MCBSPLP_XCCR_REG[5] DLB = 1, or MCBSPLP_SRGR2_REG[15] GSYNC = 1
- mcbspi_clkx pin (CLKX): When MCBSPLP_PCR_REG[9] CLKXM = 0, MCBSPLP_XCCR_REG[5] DLB = 0, or MCBSPLP_SRGR2_REG[15] GSYNC = 0

The FSR_int frame-sync signal can be derived from three sources:

- The sample rate generator (FSG): When MCBSPLP_XCCR_REG[5] DLB = 1, MCBSPLP_SRGR2_REG[15] GSYNC = 1, or MCBSPLP_PCR_REG[10] FSRM = 1 and MCBSPLP_PCR_REG[8] CLKRM = 1
- mcbsp1_fsr pin (FSR): When MCBSPLP_PCR_REG[9] CLKXM = 0 and MCBSPLP_XCCR_REG[15] EXTCLKGATE = 0 and MCBSPLP_XCCR_REG[5] DLB = 0 or MCBSPLP_SRGR2_REG[15] GSYNC = 0, or MCBSPLP_PCR_REG[10] FSRM = 0 and MCBSPLP_PCR_REG[8] CLKRM = 0

- mcbspi_fsx pin (FSX): When MCBSPLP_PCR_REG[9] CLKXM = 1 and MCBSPLP_XCCR_REG[15] EXTCLKGATE = 1, and MCBSPLP_XCCR_REG[5] DLB = 0 or MCBSPLP_SRGR2_REG[15] GSYNC = 0, or MCBSPLP_PCR_REG[10] FSRM = 0 and MCBSPLP_PCR_REG[8] CLKRM = 0

The FSX_int frame-sync signal can be derived from two sources:

- The sample rate generator (FSG): When MCBSPLP_XCCR_REG[5] DLB = 1 or MCBSPLP_SRGR2_REG[15] GSYNC = 1, or MCBSPLP_SRGR2_REG[12] FSGM = 1 and MCBSPLP_PCR_REG[11] FSXM = 1
- mcbspi_fsx pin (FSX): When MCBSPLP_XCCR_REG[5] DLB = 0 or MCBSPLP_SRGR2_REG[15] GSYNC = 0, or MCBSPLP_SRGR2_REG[12] FSGM = 0 and MCBSPLP_PCR_REG[11] FSXM = 0

18.4.2.3.1 Clocking

Data is shifted one bit at a time from the mcbspi_dr pin to the RSRs or from the XSRs to the mcbspi_dx pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR_int) controls bit transfers from the mcbspi_dr pin to the RSRs. The transmit clock signal (CLKX_int) controls bit transfers from the XSRs to the mcbspi_dx pin. CLKR_int or CLKX_int signals can be derived from a pin at the boundary of the McBSP module (mcbspi_clkr and mcbspi_clkx respectively) or derived from inside the McBSP module (see Figure 18-26). The clocks source is selected by programming the McBSPi.MCBSPLP_PCR_REG[9] CLKXM bit and the McBSPi.MCBSPLP_PCR_REG[8] CLKRM bit respectively.

When the McBSPi.MCBSPLP_PCR_REG[9] CLKXM bit (transmitter clock mode) is set to:

- 0: CLKX_int is driven by an external clock and mcbspi_clkx is an input pin.
- 1: CLKX_int is driven by the internal sample rate generator and mcbspi_clkx is an output pin.

For the McBSPi.MCBSPLP_PCR_REG[8] CLKRM bit (receiver clock mode), see Table 18-15.

Table 18-15. Receiver Clock Mode

Value	Digital loop back mode	mcbspi_clkr pin	Description
0x0	McBSPi.MCBSPLP_XCCR_REG[5] DLB bit =0	input	CLKR_int is driven by an external clock
	McBSPi.MCBSPLP_XCCR_REG[5] DLB bit =1	High-impedance	CLKR_int is driven by CLKX_int. The CLKX_int is derived based on the CLKXM value.
0x1	McBSPi.MCBSPLP_XCCR_REG[5] DLB bit =0	output	CLKR_int is driven by the internal sample rate generator
	McBSPi.MCBSPLP_XCCR_REG[5] DLB bit =1	output	CLKR_int is driven by CLKX_int. The CLKX_int is derived based on the CLKXM value.

The polarities of CLKR and CLKX signals are configured in McBSPi.MCBSPLP_PCR_REG register.

The McBSPi.MCBSPLP_PCR_REG[1] CLKXP defines the transmit clock polarity:

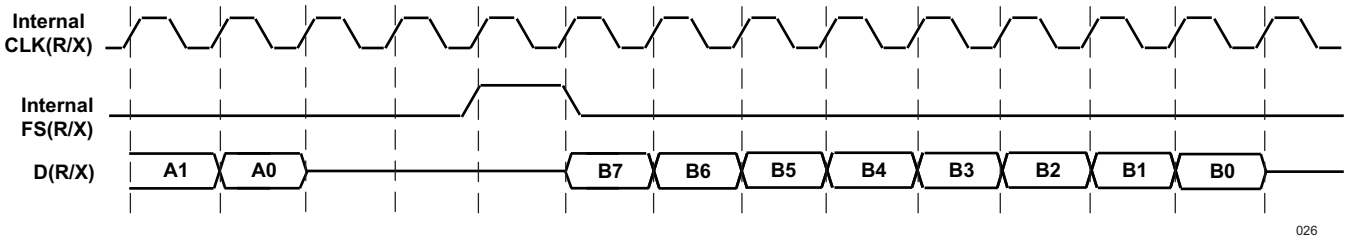
- When set to 0, transmit data driven on rising edge of CLKX signal
- When set to 1, transmit data driven on falling edge of CLKX signal

The McBSPi.MCBSPLP_PCR_REG[0] CLKRP defines the receive clock polarity:

- When set to 0, receive data sampled on falling edge of CLKX signal
- When set to 1, transmit data sampled on rising edge of CLKX signal

Figure 18-27 gives an example where the clock signal controls the timing of each bit transfer on the pin.

Figure 18-27. Clock Signal Control of Bit Transfer Timing



NOTE: The McBSP module is constrained to operate at an internal functional frequency of up to L4 interface frequency divided by 2. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX/CLKR/CLKS, choose an appropriate input clock frequency (up to L4 interface frequency) and divide down value by programming the McBSPi.MCBSPLP_SRGR1_REG[7:0] CLKGDV bit field.

18.4.2.3.2 Serial Words

Bits traveling between a shift register (RSR or XSR) and a data pin (mcbspi_dr or mcbspi_dx) are transferred in a group called a serial word. The software defines how many bits are in a word by programming:

- For the receiver: the McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 field and the McBSPi.MCBSPLP_RCR2_REG[7:5] RWDLEN2 field.
- For the transmitter: the McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 field and the McBSPi.MCBSPLP_XCR2_REG[7:5] XWDLEN2 field.

The difference of use is explained to Section 18.4.2.4.1.

The various possibilities of word length are 8, 12, 16, 20, 24, and 32 bits (for field values, see Section 18.6)

Bits coming from the mcbspi_dr pin are held in the RSR until it holds a full serial word, then the word is passed to the RB and to the McBSPi.MCBSPLP_DRR_REG register.

During transmission, XSR accepts new data from XB after a full serial word has been passed from XSR to the mcbspi_dx pin.

In the example in Figure 18-27, an 8–bit word size was defined (see the transfer of the 8-bit word B).

18.4.2.3.3 Frames and Frame Synchronization

One or more words (up to 128) are transferred in a group called a frame. The software defines how many words are in a frame by programming:

- For the receiver: The McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 field and the McBSPi.MCBSPLP_RCR2_REG[14:8] RFRLEN2 field.
- For the transmitter: The McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 field and the McBSPi.MCBSPLP_XCR2_REG[14:8] XFRLEN2 field.

The difference between these registers is explained to Section 18.4.2.4.1. For the corresponding between field value and words number, see Section 18.6.

All the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP module uses frame-synchronization signals (FSG) to determine when each frame is received/transmitted. When a pulse occurs on a frame-synchronization signal, the McBSP module begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP module receives/transmits the next frame, and so on.

Pulses on the receive frame-synchronization (FSR_int) signal initiate frame transfers on mcbspi_dr. Pulses on the transmit frame-sync (FSX_int) signal initiate frame transfers on mcbspi_dx. FSR_int or FSX_int signals can be derived from a pin at the boundary of the McBSP module (mcbspi_fsr and mcbspi_fsx respectively) or derived from inside the McBSP module (see Figure 18-25). The frame-sync source is selected by programming the McBSPi.MCBSPLP_PCR_REG[11] FSXM bit and the McBSPi.MCBSPLP_PCR_REG[10] FSRM bit respectively.

When the McBSPi.MCBSPLP_PCR_REG[11] FSXM bit (transmitter frame-sync mode) is set to:

- 0: FSX_int is derived from an external source and mcbspi_fsx is an input pin.
- 1: FSX_int is determined by the McBSPi.MCBSPLP_SRGR2_REG[12] FSGM bit and mcbspi_fsx is an output pin.

For the McBSPi.MCBSPLP_PCR_REG[10] FSRM bit (receiver frame-sync mode), is set to:

- 0: FSR_int is generated by an external source and mcbsp1_fsr is an input pin.
- 1: FSR_int is generated internally by sample rate generator. The mcbsp1_fsr is an output pin except when McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit = 0x1.

In the example in Figure 18-26, a one-word frame is transferred when a frame-synchronization pulse occurs. The polarities of FSR and FSX signals are programmable by bits on McBSPi.MCBSPLP_PCR_REG register.

The McBSPi.MCBSPLP_PCR_REG[3] FSXP defines the transmit frame-sync polarity:

- When set to 0, frame-sync pulse FSX is active high.
- When set to 1, frame-sync pulse FSX is active low.

The McBSPi.MCBSPLP_PCR_REG[2] FSRP defines the receive frame-sync polarity:

- When set to 0, frame-sync pulse FSR is active high.
- When set to 1, frame-sync pulse FSR is active low.

In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-synchronization signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

18.4.2.3.4 Detecting Frame-Synchronization Pulses, Even in Reset State

The McBSP module can generate receive and transmit interrupts to the MPU/IVA2.2 subsystems to indicate specific events in the McBSP module. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame-synchronization pulses (see Section 18.5 for further details).

Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating receive interrupt when the receiver is in reset). In this case, McBSPi.MCBSPLP_PCR_REG[0] FSRM bit/McBSPi.MCBSPLP_PCR_REG[1] FSXM bit and McBSPi.MCBSPLP_PCR_REG[2] FSRP bit/McBSPi.MCBSPLP_PCR_REG[3] FSXP bit still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the interface clock (McBSPi_ICLK) and then sent to the MPU/IVA2.2 subsystem in the form of receive interrupt and transmit interrupt at the point where they feed the receiver and transmitter of the serial port. Consequently, a new frame-synchronization pulse can be detected, and then, the MPU/IVA2.2 subsystem can take the serial port out of reset safely.

18.4.2.3.5 Ignoring Frame-Synchronization Pulses

The McBSP module ignores transmit and/or receive frame-synchronization pulses if the frame transfer was started by a previous frame-synchronization pulse (unexpected frame-synchronization pulses). The McBSP module does not support features like retransmit or re-receive of an erroneous frame or word. The receiver or transmitter ignores frame-synchronization pulses until the desired frame length or number of words is reached. For more details on unexpected frame-synchronization pulses, see Section 18.4.4.3, or Section 18.4.4.6

18.4.2.3.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined as shown in the following equation:

Frame frequency = Clock frequency/(Number of clock cycles between two rising edges [or falling edges] of two consecutive frame synchronization pulses)

The frame frequency can be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

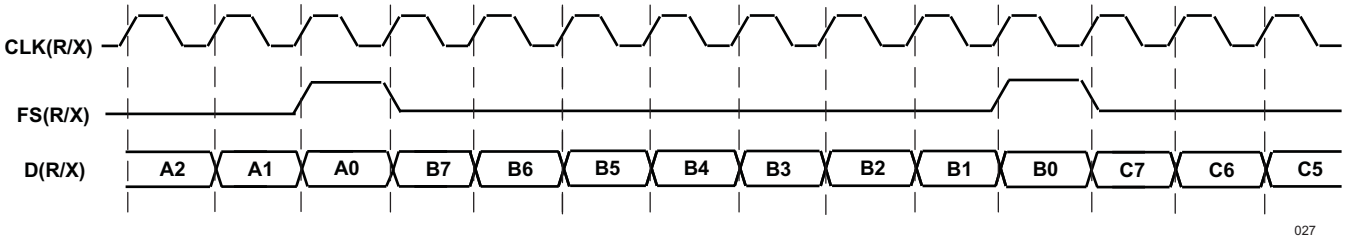
18.4.2.3.7 Maximum Frame Frequency

The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown in the following equation:

Maximum frame frequency = clock frequency/Number of bits per frame

Figure 18-28 below shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

Figure 18-28. McBSP Operating at Maximum Packet Frequency



If there is a 1–bit data delay as shown in Figure 18-28, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, back-to-back transfers.

NOTE: For McBSPi.MCBSPLP_XCR2_REG[1:0] XDATDLY = 0x0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX_int). For more details, see Section 18.5.1.6.2.2.5.

18.4.2.4 Frame Phases (Dual-Phase Frame I2S Support)

The McBSP module allows you to configure each frame to contain one or two phases. The support for dual-phase frames is required to provide I2S fully compliant capabilities (audio left and right channels—stereo audio stream).

CAUTION

The limitation on dual-phase frame support is that the number of words per phase must be set to 1 for both first and second phase. It is the only possible value for word per frame when using the dual phase frame.

The number of bits per word can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, a user might define a frame as consisting of one phase containing one word of 16 bits, followed by a second phase consisting of one word of 32 bits. This configuration allows the user to compose frames for custom applications such as I2S protocol.

18.4.2.4.1 Number of Phases, Words, and Bits per Frame

Table 18-16 below shows which bit fields in the receive control registers (McBSPi.MCBSPLP_RCR1_REG and McBSPi.MCBSPLP_RCR2_REG) and in the transmit control registers (McBSPi.MCBSPLP_XCR1_REG and McBSPi.MCBSPLP_XCR2_REG) determine the number of phases per frame, the number of words per frame, and the number of bits per word for each phase, for both receiver and transmitter. The maximum number of words per frame is limited to 2 when using dual-phase frames (one word for each phase), and to 128 for a single-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

The following legend applies to the table:

- RPHASE = McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit
- XPHASE = McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit
- RFRLEN1 = McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 field
- RFRLEN2 = McBSPi.MCBSPLP_RCR2_REG[14:8] RFRLEN2 field
- XFRLEN1 = McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 field
- XFRLEN2 = McBSPi.MCBSPLP_XCR2_REG[14:8] XFRLEN2 field
- RWDLEN1 = McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 field
- RWDLEN2 = McBSPi.MCBSPLP_RCR2_REG[7:5] RWDLEN2 field
- XWDLEN1 = McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 field
- XWDLEN2 = McBSPi.MCBSPLP_XCR2_REG[7:5] XWDLEN2 field

Table 18-16. Phases, Words and Bits per Frame Control Bit

Operation	Number of phases	Words per frame set with	Bits per word set with
Reception	1 (RPHASE=0)	RFRLEN1	RWDLEN1
Reception	2 (RPHASE=1)	RFRLEN1=0x0 and RFRLEN2=0x0	RWDLEN1 for phase 1 RWDLEN2 for phase 2
Transmission	1 (XPHASE=0)	XFRLEN1	XWDLEN1
Transmission	2 (XPHASE=1)	XFRLEN1=0x0 and XFRLEN2=0x0	XWDLEN1 for phase 1 XWDLEN2 for phase 2

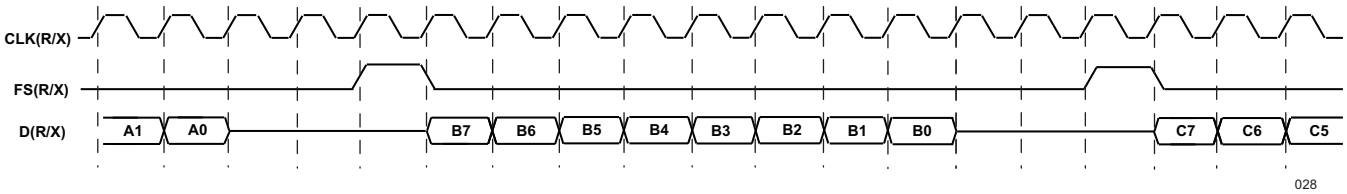
18.4.2.4.2 Single-Phase Frame Example

Figure 18-29 below shows an example of a single-phase data frame containing one 8-bit word. Because the transfer is configured for one data bit delay, the data on the mcbspi_dx and mcbspi_dr pins are available one clock cycle after FS(R/X) goes active. The following table shows the assumptions used in the example of this figure.

Table 18-17. Assumptions for the Single-Phase Frame Example

Assumption	Value	Bit or Field Name
Single-phase frame	‘0’	McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit
		McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit
One word per frame	0x0	McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 field
		McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 field
8-bit word length	0x0	McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 field
		McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 field
word length in register2	ignored	McBSPi.MCBSPLP_RCR2_REG[14:8] RFRLEN2 field
		McBSPi.MCBSPLP_XCR2_REG[14:8] XWDLEN2 field
Receive data clocked on falling edge	‘0’	McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit
Transmit data clocked on rising edge		McBSPi.MCBSPLP_PCR_REG[1] CLKXP bit
Active–high frame–sync signals	‘0’	McBSPi.MCBSPLP_PCR_REG[2] FSRP bit
		McBSPi.MCBSPLP_PCR_REG[3] FSXP bit
1-bit data delay	01b	McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY field
		McBSPi.MCBSPLP_XCR2_REG[1:0] XDARDLY field

Figure 18-29. Single-Phase Frame for a McBSP Data Transfer



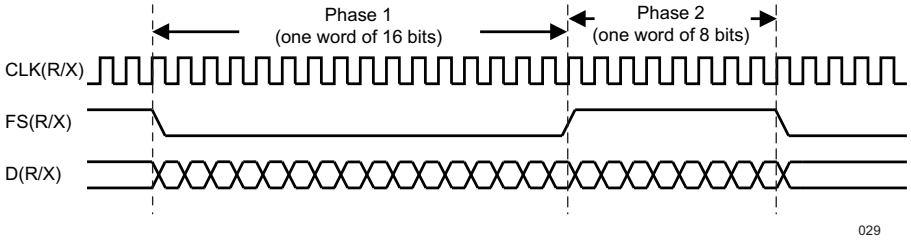
18.4.2.4.3 Dual-Phase Frame Example

Figure 18-30 below shows an example of a frame. The first phase consists of one word of 16 bits, followed by a second phase of one word of 8 bits. The entire bit stream in the frame is contiguous. There are no gaps between words/phases. Table 18-18 shows the assumptions used to the example in Figure 18-30.

Table 18-18. Assumptions for the Dual-Phase Frame Example

Assumption	Value	Bit or Field name
Single-phase frame	'1'	McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit
		McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit
One word per frame	0x0	McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 field
		McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 field
16-bit word length	0x0	McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 field
		McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 field
8-bit word length	0x2	McBSPi.MCBSPLP_RCR2_REG[14:8] RFRLEN2 field
		McBSPi.MCBSPLP_XCR2_REG[14:8] XWDLEN2 field
Receive data clocked on falling edge	'0'	McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit
Transmit data clocked on rising edge		McBSPi.MCBSPLP_PCR_REG[1] CLKXP bit
Active-high frame-sync signals	'0'	McBSPi.MCBSPLP_PCR_REG[2] FSRP bit
		McBSPi.MCBSPLP_PCR_REG[3] FSXP bit
0-bit data delay	00b	McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY field
		McBSPi.MCBSPLP_XCR2_REG[1:0] XDARDLY field

Figure 18-30. Dual-Phase Frame for a McBSP Data Transfer



18.4.2.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP module. For details about how to program the McBSP receiver, see Section 18.5, Section 18.5.1.4, and Section 18.5.1.5.

Figure 18-31 and Figure 18-32 below show how reception occurs in the McBSP module. A description of the process follows the figures. Figure 18-31 shows the physical path for the data.

Figure 18-31. McBSP Reception Physical Data Path

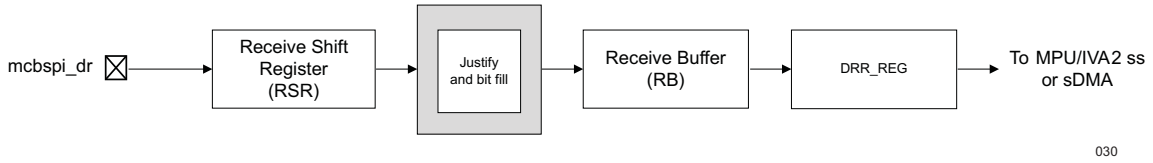
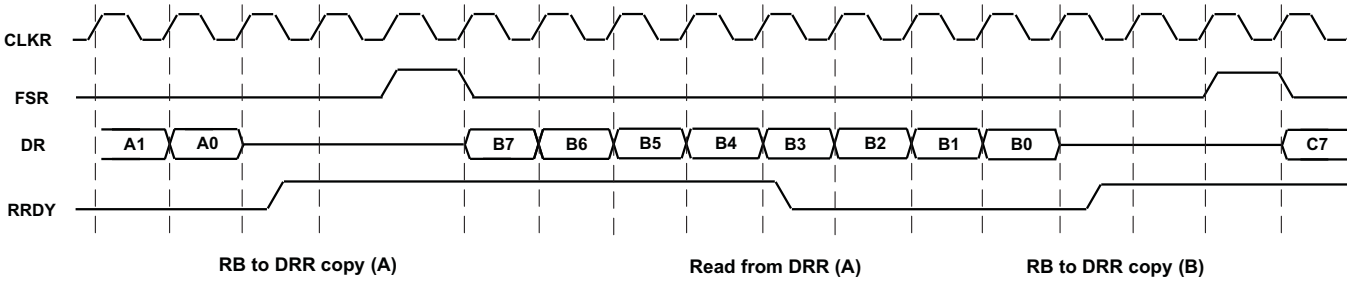


Figure 18-32 is a timing diagram showing signal activity for one possible reception scenario.

Figure 18-32. McBSP Reception Signal Activity



RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the mcbspi_dr pin to the MPU/IVA2.2 subsystem or to the sDMA controller:

1. The McBSP module waits for a receive frame-synchronization pulse on FSR_int.
2. When the pulse arrives, the McBSP module inserts the appropriate data delay that is selected with the McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY bits. In the preceding timing diagram a 1-bit data delay is selected.
3. The McBSP module accepts data bits on the mcbspi_dr pin and shifts them into the RSR. For details on choosing a word length, see Section 18.5.1.5.2.2.2.
4. When a full word is received, the McBSP module copies the contents of the RSR to the RB, provided that RB is not full.
5. When the programmed receive threshold is reached (McBSPi.MCBSPLP_THRSH1_REG[10:0] RTHRESHOLD field), the McBSP module asserts the receiver ready bit (McBSPi.MCBSPLP_SPCR1_REG[1] RRDY bit). This indicates that receive data is ready to be read by the MPU/IVA2.2 subsystem or the sDMA controller by accessing McBSPi.MCBSPLP_DRR_REG register.

   The data copied from RB to McBSPi.MCBSPLP_DRR_REG is justified and bit filled according to the McBSPi.MCBSPLP_SPCR1_REG[14:13] RJUST field.
6. The MPU/IVA2.2 subsystem or the sDMA controller reads the data from the data receive register. When the RB is empty, McBSPi.MCBSPLP_SPCR1_REG[1] RRDY bit is cleared.

18.4.2.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP module. For details about how to program the McBSP transmitter, see Section 18.5, and Section 18.5.1.6.

Figures below show how transmission occurs in the McBSP module. A description of the process follows the figures. Figure 18-33 shows the physical path for the data.

Figure 18-33. McBSP Transmission Physical Data Path

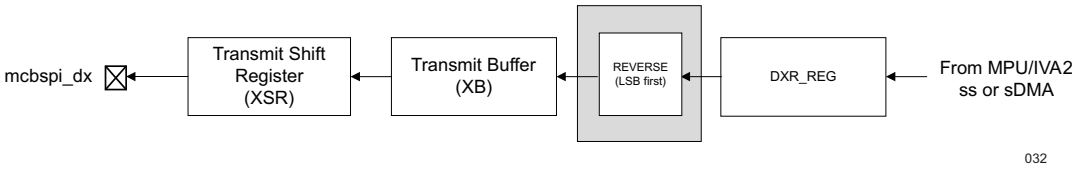
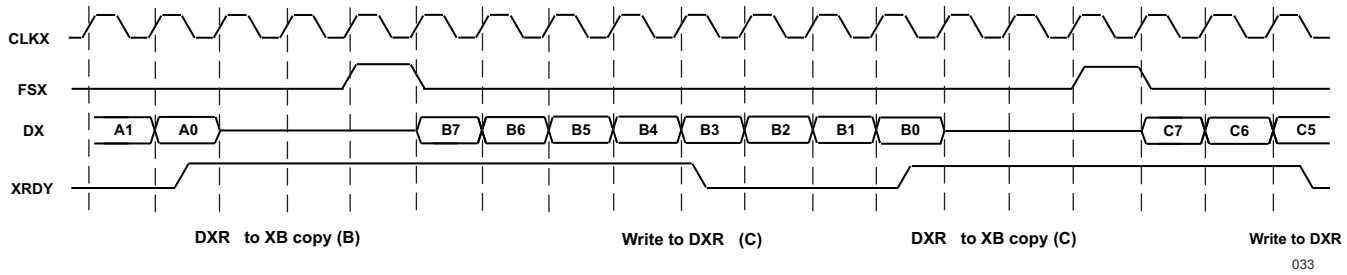


Figure 18-34 is a timing diagram showing signal activity for one possible transmission scenario.

Figure 18-34. McBSP Transmission Signal Activity



XRDY: Status of transmitter ready bit (high is 1)

1. The MPU/IVA2.2 subsystem or the sDMA controller writes data to the data transmit register (McBSPi.MCBSPLP_DXR_REG). When the XB is reached the transmitter ready bit (McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit) is cleared to indicate that the transmitter is not ready for new data. For details on choosing a word length, see Section 18.5.1.6.2.2.2.
2. When new data arrives in McBSPi.MCBSPLP_DXR_REG register, the McBSP module copies the content of the data transmit register to the XB. In addition, the transmit ready bit (McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit) is set as long as the buffer contains at least the transmit threshold number of free locations (McBSPi.MCBSPLP_THRSH2_REG[10:0] XTHRESHOLD field). This indicates that the transmitter is ready to accept new data from the MPU/IVA2.2 subsystem or the sDMA controller.
3. The McBSP module waits for a transmit frame-synchronization pulse on FSX_int.
4. When the pulse arrives, the McBSP module inserts the appropriate data delay that is selected with the McBSPi.MCBSPLP_XCR2_REG[1:0] XDATDLY field.

   In the preceding timing diagram, a 1-bit data delay is selected.
5. The McBSP module shifts data bits from the XSR to the mcbspi_dx pin.

18.4.2.7 Enable/Disable the Transmit and Receive Processes

The McBSP module has the option to stop-resume the transmit/receive process while the module is in functional mode (out of transmit/receive reset).

When the transmit/receive disable bit (McBSPi.MCBSPLP_XCCR_REG[0] XDISABLE/McBSPi.MCBSPLP_RCCR_REG[0] RDISABLE) is set, the McBSP module stops the transmit/receive operation at the next frame boundary (frame corruption avoided).

During the receive disable state, the frames that are sent (when FSR signal is asserted while receive disable) by the remote device are lost, and receive buffer overflow status bit (McBSPi.MCBSPLP_IRQSTATUS_REG[5] ROVFLSTAT) is not set. Also, the frames received by the remote device while McBSPi.MCBSPLP_XCCR_REG[0] XDISABLE bit is set (when FSX signal is asserted while transmit disable) are meaningless undefined data frames, and transmit buffer underflow status bit (McBSPi.MCBSPLP_IRQSTATUS_REG[11] XUNDFLSTAT) is not set. The presence of the frame synchronization, while transmit/receive process is disabled, can be checked by reading the transmit/receive Frame-sync interrupt status: McBSPi.MCBSPLP_IRQSTATUS_REG[8] XFSX / McBSPi.MCBSPLP_IRQSTATUS_REG[1] RFSR bits.

As soon as the McBSPi.MCBSPLP_XCCR_REG[0] XDISABLE/McBSPi.MCBSPLP_RCCR_REG[0] RDISABLE bit is cleared, the transmit/receive process resumes at the next frame boundary.

NOTE: It is not recommended to use this mechanism together with the possibility to interrogate the transmit/receive buffer status register (McBSPi.MCBSPLP_XBUFFSTAT_REG[7:0] XBUFFSTAT/McBSPi.MCBSPLP_RBUFFSTAT_REG[7:0] RBUFFSTAT field indicating the occupied/available buffer locations), since this register is an interface clock (McBSPi_ICLK) synchronous register and does not reflect the exact number of occupied/free locations available on the functional clock domain.

18.4.2.8 MCBSP Data Transfer Mode

NOTE: For all examples in this section, the configured CLKX edge is the rising edge (McBSPi.MCBSPLP_PCR_REG[1] CLKXP bit = 0x0) and the configured CLKR edge is the falling edge (McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit = 0x0). These are the reset values.

In the following timing diagrams, a 1-bit data delay is selected (McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY field = 0x01 and McBSPi.MCBSPLP_XCR2_REG[1:0] XDATDLY bit field = 0x01), because data often follows a 1-cycle active frame-synchronization.

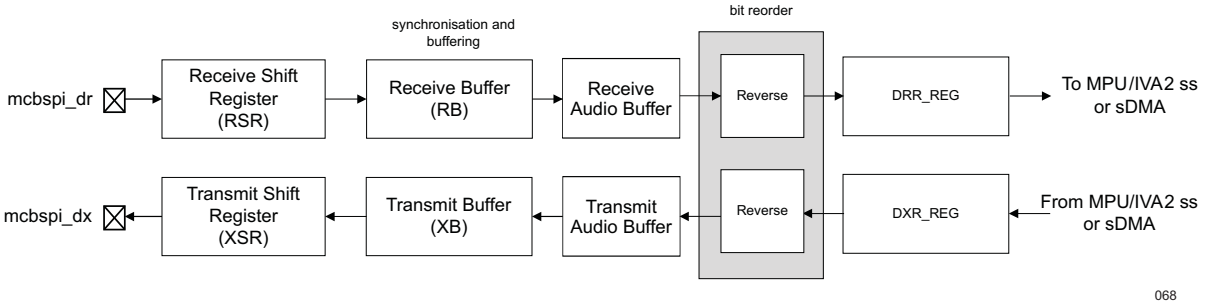
McBSP modules can support two edge selection modes for transmit and receive data transfer at the system level:

- The full cycle mode, for which one clock period is used to transfer the data, generated on one edge and captured on the same edge (one clock period later).
- The half cycle mode, for which one half clock period is used to transfer the data, generated on one edge and captured on the opposite edge (one half clock period later). A new data is generated only every clock period, which permits to ensure the required hold time.

18.4.2.8.1 Transmit Full Cycle Mode

When configured in full cycle mode (McBSPi.MCBSPLP_XCCR_REG[11] XFULL_CYCLE bit = 0x1), the FSX signal is sampled on the configured CLKX edge and the data is driven on the same configured edge. See Figure 18-35.

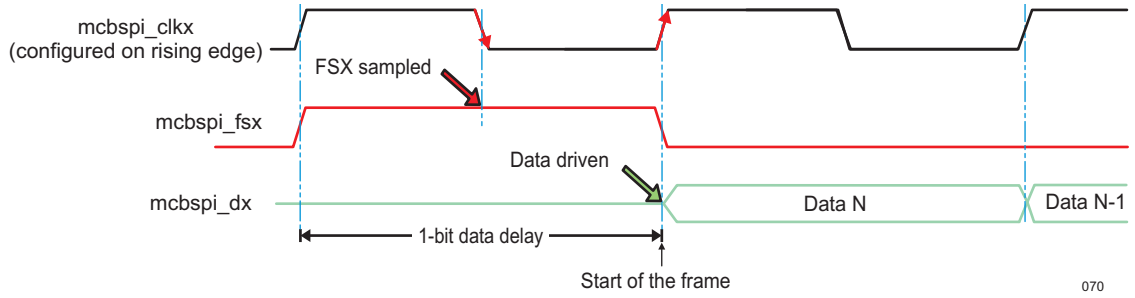
Figure 18-35. Transmit Full Cycle Timing Diagram



18.4.2.8.2 Transmit Half Cycle Mode

When configured in half cycle mode (McBSPi.MCBSPLP_XCCR_REG[11] XFULL_CYCLE bit = 0x0, reset value), the FSX signal is sampled on the opposite configured CLKX edge and the data is driven on the next configured edge. See Figure 18-36.

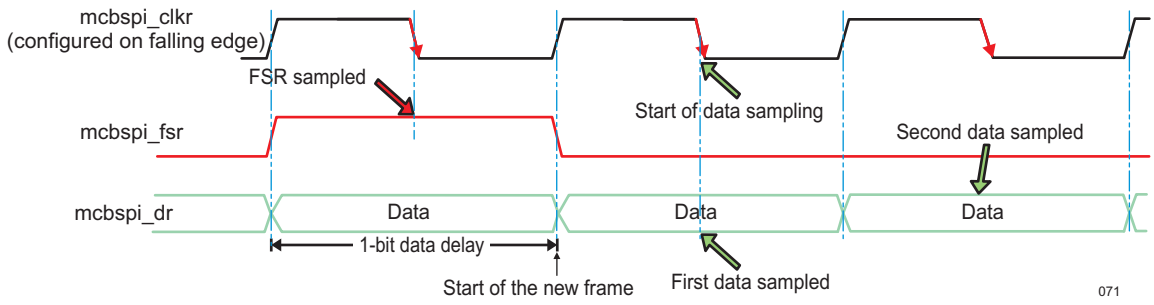
Figure 18-36. Transmit Half Cycle Timing Diagram



18.4.2.8.3 Receive Full Cycle Mode

When configured in full cycle mode (McBSPi.MCBSPLP_RCCR_REG[11] RFULL_CYCLE bit = 0x1, reset value), the FSR signal is sampled on the configured CLKR edge and the data is driven on the same configured edge. See Figure 18-37.

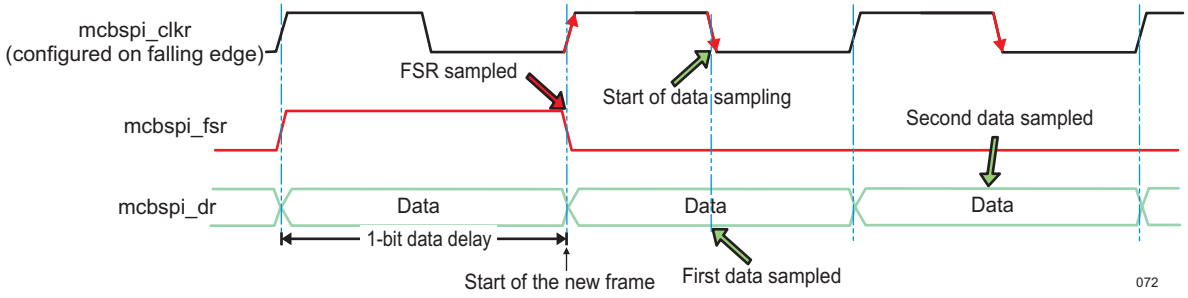
Figure 18-37. Receive Full Cycle Timing Diagram



18.4.2.8.4 Receive Half Cycle Mode

When configured in half cycle mode (McBSPi.MCBSPLP_RCCR_REG[11] RFULL_CYCLE bit = 0x0), the FSR signal is sampled on the opposite configured CLKR edge and the data is driven on the next configured edge. See Figure 18-38.

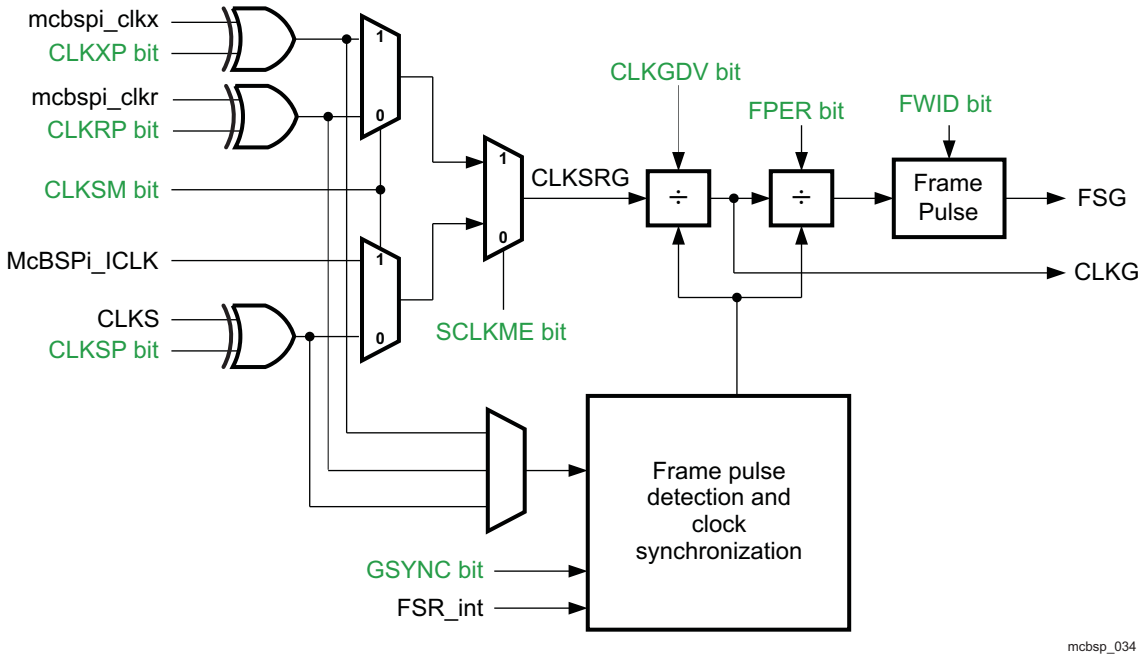
Figure 18-38. Receive Half Cycle Timing Diagram



18.4.3 McBSP SRG

The McBSP module contains an internal SRG that can be used to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive pin (mcbspi_dr) and/or the data transmit pin (mcbspi_dx). FSG can be used to initiate frame transfers on mcbspi_dr pin and/or mcbspi_dx pin. Figure 18-39 is a conceptual block diagram of the SRG.

Figure 18-39. Conceptual Block Diagram of the Sample Rate Generator



The source clock for the SRG (labeled CLKSRG in the diagram) can be supplied by either the interface clock (McBSPi_ICLK), or the functional clock (CLKS input), or by an external pin (mcbspi_clkx, or mcbspi_clkr). The source is selected with the McBSPi.MCBSPLP_PCR_REG[7] SCLKME bit and the McBSPi.MCBSPLP_SRGR2_REG[13] CLKSM bit.

If a pin or CLKS signal is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (McBSPi.MCBSPLP_SRGR2_REG[14] CLKSP bit, McBSPi.MCBSPLP_PCR_REG[1] CLKXP bit, or McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit).

The SRG has a three-stage clock divider that gives CLKG and FSG programmability.

The three stages provide:

- Clock divide-down: The source clock (CLKSRG) is divided according to the McBSPi.MCBSPLP_SRGR1_REG[7:0] CLKGDV field to produce CLKG signal
- Frame period divide-down: CLKG is divided according to the McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER field to control the period from the start of a frame-pulse to the start of the next pulse
- Frame-synchronization pulse-width countdown: CLKG cycles are counted according to the McBSPi.MCBSPLP_SRGR1_REG[15:8] FWID field to control the width of each frame-synchronization pulse

NOTE: The McBSP module cannot operate at an internal functional frequency faster than L4 interface frequency divided by 2. Choose an input clock frequency and a McBSPi.MCBSPLP_SRGR1_REG[7:0] CLKGDV value such that CLKG is less than or equal to L4 interface frequency divided by 2.

In addition to the three-stage clock divider, the sample rate generator has a frame-synchronization pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-synchronization pulse on the mcbspi_fsr pin. This feature is enabled or disabled with the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit.

CLKG is used as source to generate the output clocks CLKX/CLKR when the McBSPi.MCBSPLP_PCR_REG[9] CLKXM / McBSPi.MCBSPLP_PCR_REG[8] CLKRM bit indicates that the clock is an output. The output CLKX/CLKR is generated according to the clock polarity setting (see Figure 18-26).

For details on getting the sample rate generator ready for operation, see Section 18.5.

18.4.3.1 Clock Generation in the SRG

The SRG can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the SRG to drive clocking is controlled by the clock mode bits (McBSPi.MCBSPLP_PCR_REG[9] CLKXM and McBSPi.MCBSPLP_PCR_REG[8] CLKRM) and polarity mode bits (McBSPi.MCBSPLP_PCR_REG[1] CLKXP and McBSPi.MCBSPLP_PCR_REG[0] CLKRP).

When a clock mode bit is set to 1 (CLKRM=1 for reception, CLKXM=1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal SRG output clock (CLKG) according to the polarity setting.

The effects of this setting on the McBSP module are partially affected by the use of the digital loopback (DLB) mode, the analog loop back (ALB) mode and by the synchronous receive/transmit setting, respectively, as described in Table 18-19. The ALB mode is selected with the McBSPi.MCBSPLP_SPCR1_REG[15] ALB bit. The DLB mode is selected with the McBSPi.MCBSPLP_XCCR_REG[5] DLB bit. The synchronous setting is controlled by input signals. These signals are defined by the control registers of the System Control Module (more details, refer Section 18.3.1)

When using the SRG as a clock source, make sure the SRG is enabled (McBSPi.MCBSPLP_SPCR2_REG[6] GRST bit =1).

Table 18-19. Effects of DLB and ALB Bits on Clock Modes

Mode Bit Settings		Effect
CLKRM=1	DLB=0 and ALB = 0 (Digital and analog loop back mode disabled)	mcbsp1_clkr is an output pin driven by the SRG output clock (CLKG).
	DLB=0 and ALB = 1 (Digital loop back mode disabled and Analog loop back mode enabled)	mcbsp1_clkr is an output pin driven by the SRG output clock (CLKG). The receiver functional part internal clock is driven by CLKX input signal provided by mcbspi_clkx pin. The source of CLKX depends on the CLKXM bit. The receive frame synchronization is driven by FSX input signal provided by mcbspi_fsx pin. The receive data is driven by the DX input loop-back pin (mcbspi_dx).
	DLB=1 ALB = 0 (Digital loop back mode enabled and Analog loop back mode disabled)	The SRG and the frame synchronization generator must be enabled. The internal transmit and receive clocks are driven by the SRG (CLKG having the appropriate CLKXP polarity). The transmit and receive frame synchronization signals are driven by FSG (having the appropriate FSXP polarity). The transmit data is connected to the DR input data. Note that in digital loop back mode no serial link activity will be seen by the remote device.
	DLB=1 ALB = 1 (reserved mode)	Undefined functionality.

Table 18-19. Effects of DLB and ALB Bits on Clock Modes (continued)

Mode Bit Settings		Effect
CLKXM= 1	DLB=0 ALB = 0 (Digital analog loop back mode disabled)	mcbspi_clkx is an output pin driven by the SRG output clock (CLKG).
	DLB=0 ALB = 1 (Digital loop back mode disabled and Analog loop back mode enabled)	mcbspi_clkx is an output pin driven by the SRG output clock (CLKG).
	DLB=1 ALB = 0 (Digital loop back mode enabled and Analog loop back mode disabled)	The SRG and the frame synchronization generator need to be enabled. The internal transmit and receive clocks are driven by the SRG (CLKG having the appropriate CLKXP polarity). The transmit and receive frame synchronization signals are driven by FSG (having the appropriate FSXP polarity). The transmit data is connected to the DR input data. Note that in digital loop back mode no serial link activity will be seen by the remote device.
	DLB=1 ALB = 1 (reserved mode)	Undefined functionality.
	SCM.CONTROL_DEVCONF0[3] MCBSP1_CLKR bit =1 (synchronous setting and DLB = 0 ALB = 0)	CLKX is an output pin driven by the SRG output clock (CLKG). CLKR is connected to the CLKX.

18.4.3.2 Frame Sync Generation in the SRG

The SRG can produce a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both.

If you want the receiver to use FSG for frame synchronization, make sure McBSPi.MCBSPLP_PCR_REG[10] FSRM bit =1. (When FSRM=0, receive frame synchronization is supplied via the mcbspi_fsr pin.)

If you want the transmitter to use FSG for frame synchronization, you must set:

- McBSPi.MCBSPLP_PCR_REG[11] FSXM = 1: This indicates that transmit frame synchronization is supplied by the McBSP module itself rather than from the mcbspi_fsx pin.
- MCBSPLP_SRGR2_REG[12] FSGM=1: This indicates that when FSXM=1, transmit frame synchronization is supplied by the SRG.

NOTE: When FSGM=0 and FSXM=1, the transmit frame-sync signal (FSX) is generated when XB is not empty. When FSGM = 0, McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER and McBSPi.MCBSPLP_SRGR1_REG[15:8] FWID field are used to determine the frame synchronization period and width (external FSX is gated by the buffer empty condition).

In either case, the SRG must be enabled (McBSPi.MCBSPLP_SPCR2_REG[6] GRST bit=1) and the frame-synchronization logic in the SRG must be enabled (McBSPi.MCBSPLP_SPCR2_REG[7] FRST bit=0).

18.4.3.2.1 Choosing the Width of the Frame-sync Pulse

Each pulse on FSG has a programmable width. You program the McBSPi.MCBSPLP_SRGR1_REG[15:8] FWID field, and the resulting pulse width is (FWID+1)CLKG cycles, where CLKG is the output clock of the SRG. The range is from 1 to 256 clock periods.

18.4.3.2.2 Controlling the Period Between the Starting Edges of Frame Sync Pulses

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the SRG:

- If the SRG is using an external input clock and McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit =1, FSG pulses in response to an inactive-to-active transition on the mcbspi_fsr pin. Thus, an external device controls the frame-synchronization period.
- Otherwise, the software program the McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER field, and the

resulting frame-synchronization period is (FPER+1)CLKG cycles, where CLKG is the output clock of the SRG. The range is from 1 to 4096 clock periods.

18.4.3.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the SRG, the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit and the mcbspi_fsr pin can be used to configure the timing of FSG pulses.

McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC=1 ensures that the McBSP module and an external device are dividing down the input clock with the same phase relationship.

If McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC=1, an inactive-to-active transition on the mcbspi_fsr pin triggers a resynchronization of CLKG and generation of FSG.

18.4.3.3 Synchronizing SRG Outputs to an External Clock

The SRG can produce a clock signal (CLKG) and a FSG based on an input clock signal that is either the interface clock signal (McBSPi_ICLK), or the CLKS signal (PRCM functional clock or mcbsp_clks), or a signal at the mcbspi_clkr, or mcbspi_clkx pin. When an external clock (mcbsp_clks, or mcbspi_clkr, or mcbspi_clkx) is selected to drive the SRG, the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit and the mcbspi_fsr pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock. Make GSYNC=1 so that the McBSP module and an external device divide down the input clock with the same phase relationship.

If the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit=1:

- An inactive-to-active transition on the mcbspi_fsr pin triggers a resynchronization of CLKG signal and a pulsing of FSG signal.
- CLKG signal always begins with a high state after synchronization.
- FSR signal is always detected at the same edge of the input clock signal that generates CLKG signal, no matter how long the FSR pulse is.
- The McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER field are ignored because the frame-synchronization period on FSG is determined by the arrival of the next frame-synchronization pulse on the mcbspi_fsr pin.

If the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit=0, CLKG signal runs freely and is not resynchronized, and the frame-synchronization period on FSG signal is determined by McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER field.

18.4.3.3.1 Operating the Transmitter Synchronously with the Receiver

When the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit = 1, the transmitter can operate synchronously with the receiver, provided that the FSX signal is programmed to be driven by FSG signal (McBSPi.MCBSPLP_SRGR2_REG[12] FSGM = 1 and McBSPi.MCBSPLP_PCR_REG[11] FSXM = 1). If the FSR input signal has appropriate timing so that it can be sampled by the falling edge of CLKG signal, it can be used, instead, by setting McBSPi.MCBSPLP_PCR_REG[11] FSXM=0 and connecting FSR signal to FSX externally.

The SRG clock drives the transmit and receive clocking (McBSPi.MCBSPLP_PCR_REG[8] CLKRM bit and McBSPi.MCBSPLP_PCR_REG[9] CLKXM bit are set to 1). Therefore, the CLK(R/X) pin must not be driven by any other driving source.

18.4.3.3.2 Synchronization Examples

Figure 18-40 and Figure 18-41 show the clock and frame-synchronization operation with various polarities of CLKS (the chosen input clock) and FSR signals. These figures assume McBSPi.MCBSPLP_SRGR1_REG[15:8] FWID = 0x0, for an FSG pulse that is one CLKG cycle wide. The McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER field are not programmed; the period from the start of a frame-synchronization pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the mcbspi_fsr pin.

Each figure shows what happens to CLKG signal when the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit = 1 and if it is initially synchronized or if it is not initially synchronized. Figure 18-41 has a slower CLKG frequency (it has a larger divide-down value in the McBSPi.MCBSPLP_SRGR1_REG[7:0] CLKGDV field).

Figure 18-40. CLKG Synchronization and FSG Generation (GSYNC = 1 and CLKGDV = 0x1)

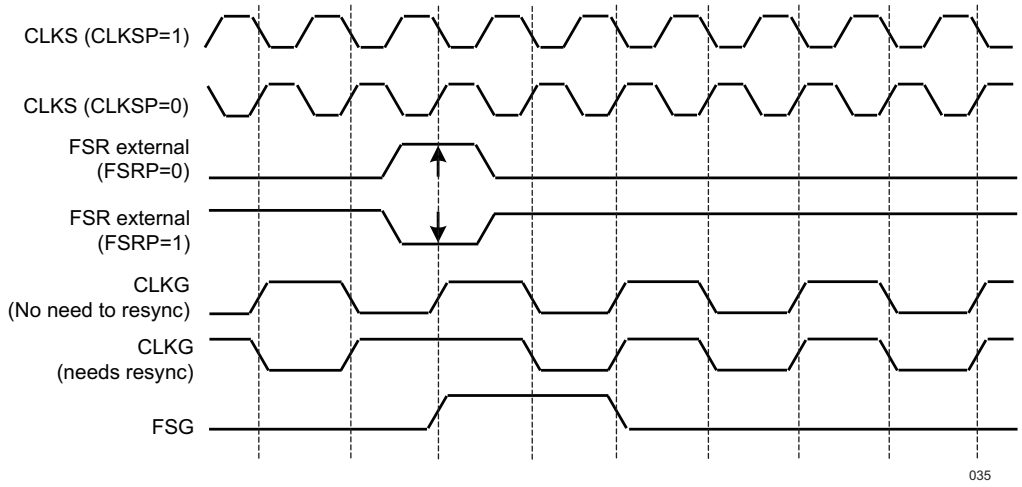
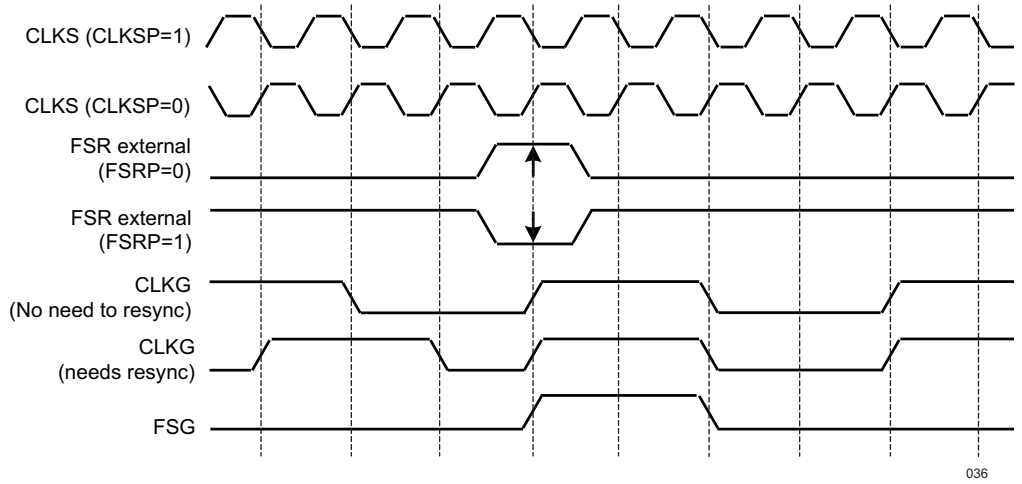


Figure 18-41. CLKG Synchronization and FSG Generation (GSYNC = 1 and CLKGDV = 0x3)



18.4.4 McBSP Exception/Error Conditions

18.4.4.1 Introduction

There are several serial port events that can constitute a system error. Any error conditions can be a source of an interrupt:

- Receiver overrun (McBSPi.MCBSPLP_IRQSTATUS_REG[5] ROVFLSTAT bit is set to 1, and legacy mode McBSPi.MCBSPLP_SPCR1_REG[2] RFULL bit is set to 1)

  This occurs when RB is full and RSR are full with another new word shifted in from mcbspi_dr. Therefore, McBSPi.MCBSPLP_IRQSTATUS_REG[5] ROVFLSTAT (McBSPi.MCBSPLP_SPCR1_REG[2] RFULL) indicates an error condition wherein any new data that can arrive at this time on mcbspi_dr replaces the contents of the RSR, and the previous word is lost. The RSR continues to be overwritten as long as new data arrives on mcbspi_dr and McBSPi.MCBSPLP_DRR_REG register is not read. For more details about overrun in the receiver, see Section 18.4.4.2.
- Unexpected receive frame-synchronization pulse (McBSPi.MCBSPLP_IRQSTATUS_REG[0]

RSYNCERR bit is set to 1, and legacy mode McBSPi.MCBSPLP_SPCR1_REG[3] RSYNCERR bit is set to 1).

This occurs during reception when an unexpected frame-synchronization pulse arrives. An unexpected frame-synchronization pulse is one that is supposed to begin the next frame transfer before all the bits of the current frame have been received. Such a pulse is ignored by the receiver, but sets the McBSPi.MCBSPLP_SPCR1_REG[3] RSYNCERR bit. For more details about receive frame-synchronization errors, see Section 18.4.4.3, *Unexpected Receive Frame-Sync Pulse*.

- Receiver underflow (McBSPi.MCBSPLP_IRQSTATUS_REG[4] RUNDFLSTAT bit is set to '1')

  This occurs when sDMA controller or MPU/IVA2.2 subsystem reads data from an empty receive buffer. For more details about underflow in the receiver, see Section 18.4.4.4.

- Transmitter underflow (McBSPi.MCBSPLP_IRQSTATUS_REG[11] XUNDFLSTAT bit is set to '1', and legacy mode McBSPi.MCBSPLP_SPCR2_REG[2] XEMPTY bit is set to '0')

  If a new frame-synchronization signal arrives when XB is empty, the previous data in the XSR is sent again. This procedure continues for every new frame-synchronization pulse that arrives until McBSPi.MCBSPLP_DXR_REG register is loaded with new data (and the XB is no longer empty). For more details about underflow in the transmitter, see Section 18.4.4.5.

- Unexpected transmit frame-synchronization pulse (McBSPi.MCBSPLP_IRQSTATUS_REG[7] XSYNCERR bit is set to '1', and legacy mode McBSPi.MCBSPLP_SPCR2_REG[3] XSYNCERR bit is set to '1')

  This occurs during transmission when an unexpected frame-synchronization pulse arrives. An unexpected pulse is one that is supposed to begin the next frame transfer before all the bits of the current frame have been transferred. Such a pulse is ignored by the transmitter, but sets the McBSPi.MCBSPLP_SPCR2_REG[3] XSYNCERR bit. For more details see Section 18.4.4.6.

- Transmitter overflow (McBSPi.MCBSPLP_IRQSTATUS_REG[12] XOVFLSTAT bit is set to '1')

  This occurs when sDMA controller or MPU/IVA2.2 subsystem writes data to a full XB. For more details about underflow in the receiver, see Section 18.4.4.7.

18.4.4.2 Overrun in the Receiver

When McBSPi.MCBSPLP_IRQSTATUS_REG[5] ROVFLSTAT bit set to '1', and McBSPi.MCBSPLP_SPCR1_REG[2] RFULL bit set to '1' (legacy mode) indicates that the receiver has experienced overrun and is in an error condition. Receive overrun is set when all of the following conditions are met:

1. McBSPi.MCBSPLP_DRR_REG is not read even if the McBSPi.MCBSPLP_IRQSTATUS_REG[3] RRDY bit is set (legacy mode) and DMA or interrupt request has been asserted.
2. RB is full
3. RSR is full

As previously described, data arriving on mcbspi_dr is continuously shifted into the Receive Shift Register (RSR). Once a complete word is shifted into the RSR, an RSR-to-RB copy can occur only if the RB is not full.

Either of the following events clears the legacy mode McBSPi.MCBSPLP_SPCR1_REG[2] RFULL bit and allows subsequent transfers to be read properly:

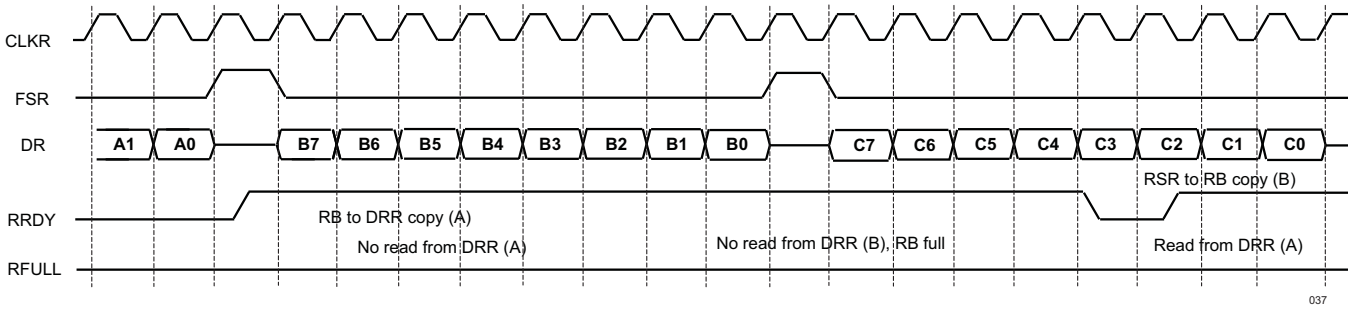
- The MPU/IVA2.2 subsystems or sDMA controller reads McBSPi.MCBSPLP_DRR_REG.
- The receiver is reset individually (McBSPi.MCBSPLP_SPCR1_REG[0] RRST bit =0) or as part of a global reset.

Another frame-synchronization pulse is required to restart the receiver.

According to the McBSPi.MCBSPLP_IRQENABLE_REG register setting, this condition can generate the McBSPi_IRQ line to be asserted low. Writing 1 to the corresponding bit in McBSPi.MCBSPLP_IRQSTATUS_REG register clears the interrupt.

Figure 18-42 shows the receive overrun condition.

Figure 18-42. Overrun in the McBSP Receiver



18.4.4.3 Unexpected Receive Frame-sync Pulse

18.4.4.3.1 Possible Responses to Receive Frame-sync Pulses

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-synchronization pulse, and the receiver sets the receive frame-synchronization error bit McBSPi.MCBSPLP_IRQSTATUS_REG[0] RSYNCERR (and the legacy McBSPi.MCBSPLP_SPCR1_REG[3] RSYNCERR bit).

According to the McBSPi.MCBSPLP_IRQENABLE_REG register settings this condition can generate the McBSPi_IRQ line to be asserted low. Writing 1 to the corresponding bit in McBSPi.MCBSPLP_IRQSTATUS_REG register clears the interrupt.

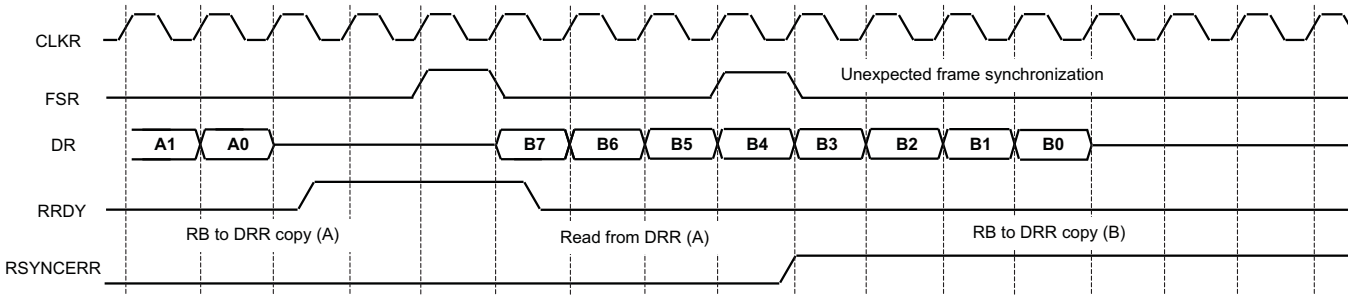
Using the legacy mode, McBSPi.MCBSPLP_SPCR1_REG[3] RSYNCERR bit can be cleared only by a receiver reset or by writing 0 to this bit. If you want the McBSP module to notify the MPU/IVA2.2 subsystem of receive frame-synchronization errors, set the legacy mode receive interrupt with the McBSPi.MCBSPLP_SPCR1_REG[5:4] RINTM field. When RINTM = 0b11, the McBSP module sends a receive interrupt (legacy mode) request to the MPU/IVA2.2 subsystems each time that RSYNCERR is set.

18.4.4.3.2 Example of an Unexpected Receive Frame-sync Pulse

Figure 18-43 shows an unexpected receive frame-synchronization pulse during normal operation of the serial port, with time intervals between data packets.

NOTE: The unexpected receive frame-synchronization pulse does not influence the data receive process, being ignored by the data receive state-machine.

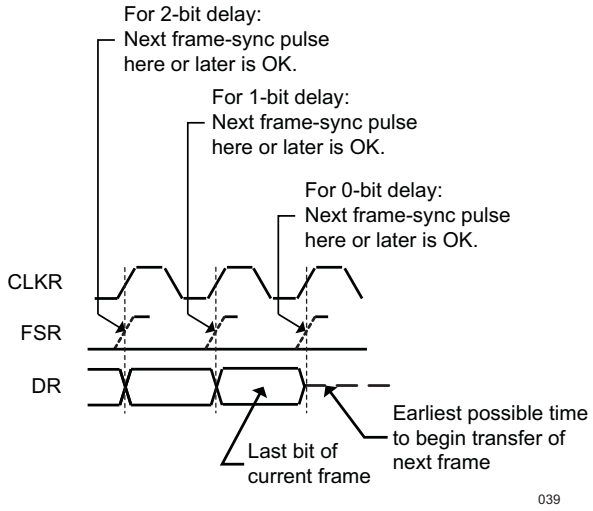
Figure 18-43. Unexpected Frame-sync Pulse During a McBSP Reception



18.4.4.3.3 Preventing Unexpected Receive Frame-sync Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKR cycles, depending on the value of the McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY field. For each possible data delay, Figure 18-44 shows when a new frame-synchronization pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 18-44. Proper Positioning of Receive Frame-sync Pulses



18.4.4.4 Underflow in the Receiver

The McBSP module indicates a receiver underflow condition by setting the McBSPi.MCBSPLP_IRQSTATUS_REG[4] RUNDFLSTAT bit. This error occurs when sDMA controller or MPU/IVA2.2 subsystem reads data from an empty RB this happens only if the MPU/IVA2.2 subsystem or sDMA controller does not respect the DMA length, does not wait for DMA request, or does not check the buffer status before reading data. According to the McBSPi.MCBSPLP_IRQENABLE_REG register settings this condition can generate the McBSPi_IRQ line to be asserted low. Writing 1 to the corresponding bit in McBSPi.MCBSPLP_IRQSTATUS_REG register clears the interrupt.

18.4.4.5 Underflow in the Transmitter

The McBSP module indicates a transmitter empty (or underflow) condition by setting the McBSPi.MCBSPLP_IRQSTATUS_REG[11] XUNDFLSTAT bit. Also the legacy mode McBSPi.MCBSPLP_SPCR2_REG[2] XEMPTY bit is cleared. Either of the following events activates XEMPTY bit (XEMPTY = 0):

- McBSPi.MCBSPLP_DXR_REG has not been loaded and XB is empty, and all bits of the data word in the XSR have been shifted out on the mcbspi_dx pin.
- The transmitter is reset (by forcing McBSPi.MCBSPLP_SPCR2_REG[0] XRST=0, or by an global reset) and is then restarted.

XEMPTY bit is deactivated (XEMPTY=1) when a new word in McBSPi.MCBSPLP_DXR_REG is transferred to Transmit Buffer (XB). If McBSPi.MCBSPLP_PCR_REG[11] FSXM=1 and McBSPi.MCBSPLP_SRGR2_REG[12] FSGM=0, the transmit frame-sync signal (FSX) is generated when Transmit Buffer (XB) is not empty. When McBSPi.MCBSPLP_SRGR2_REG[12] FSGM=0, McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER and McBSPi.MCBSPLP_SRGR1_REG[15:8] FWID are used to determine the frame-synchronization period and width (external FSX is gated by the buffer empty condition). Otherwise, the transmitter waits for the next frame-synchronization pulse before sending out the next frame on mcbspi_dx.

When the transmitter is taken out of reset (McBSPi.MCBSPLP_SPCR2_REG[0] XRST=1), it is in a transmitter ready state (McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit =1) and transmitter empty (McBSPi.MCBSPLP_SPCR2_REG[2] XEMPTY=0) state. If McBSPi.MCBSPLP_DXR_REG is loaded by the MPU/IVA2.2 subsystem or the sDMA controller before internal FSX goes active high, a valid XB-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-synchronization pulse is generated or detected. Alternatively, if a transmit frame-synchronization pulse is detected before McBSPi.MCBSPLP_DXR_REG is loaded, zeros are output on mcbspi_dx.

The McBSPi.MCBSPLP_IRQSTATUS_REG[11] XUNDFLSTAT bit indicates a real underflow condition, in which the frame is corrupted due to lack of data availability during transmit process. According to the McBSPi.MCBSPLP_IRQENABLE_REG register settings this condition can generate the McBSPi_IRQ line to be asserted low. Writing 1 to the corresponding bit in McBSPi.MCBSPLP_IRQSTATUS_REG register clears the interrupt.

18.4.4.6 Unexpected Transmit Frame-sync Pulse

18.4.4.6.1 Possible Responses to Transmit Frame-sync Pulses

If a frame-synchronization pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-synchronization pulse, and the transmitter sets the transmit frame-synchronization error bit McBSPi.MCBSPLP_IRQSTATUS_REG[7] XSYNCERR (and the legacy McBSPi.MCBSPLP_SPCR2_REG[3] XSYNCERR bit).

According to the McBSPi.MCBSPLP_IRQENABLE_REG register settings, this condition can generate the McBSPi_IRQ line to be asserted low. Writing 1 to the corresponding bit in status register clears the interrupt.

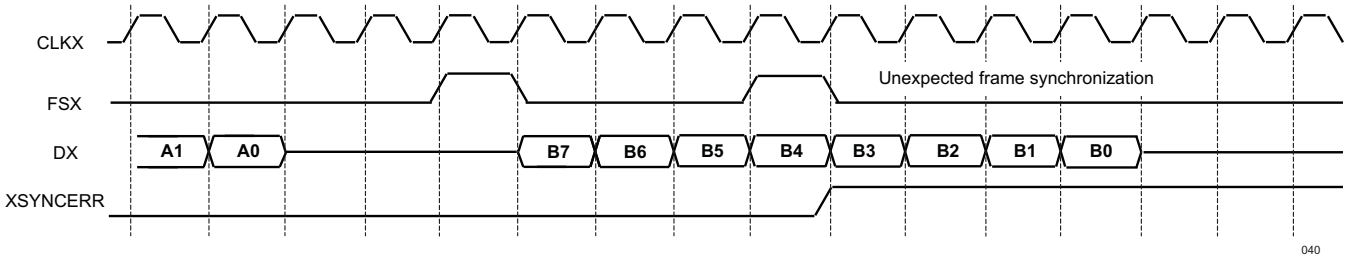
Using the legacy mode, McBSPi.MCBSPLP_SPCR2_REG[3] XSYNCERR bit can be cleared only by a transmitter reset or by a write of 0 to this bit. If you want the McBSP module to notify the MPU/IVA2.2 subsystem of frame-synchronization errors, you can set a special transmit interrupt mode with the McBSPi.MCBSPLP_SPCR2_REG[5:4] XINTM field. When XINTM=0b11, the McBSP module sends a transmit interrupt request to the MPU/IVA2.2 subsystem each time that XSYNCERR is set.

18.4.4.6.2 Example of Unexpected Transmit Frame-Synchronization Pulse

Figure 18-45 shows an unexpected transmit frame-synchronization pulse during normal operation of the serial port with intervals between the data packets.

NOTE: The unexpected transmit frame-synchronization pulse does not influence the data transmit process, being ignored by the data transmit state-machine.

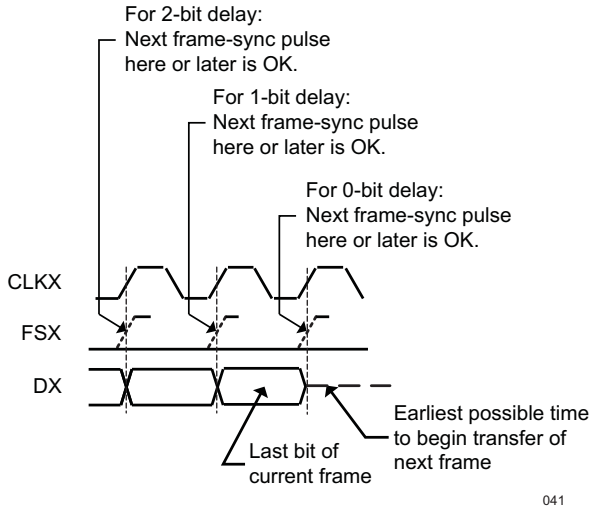
Figure 18-45. Unexpected Frame-sync Pulse During a McBSP Transmission



18.4.4.6.3 Preventing Unexpected Transmit Frame-sync Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the McBSPi.MCBSPLP_XCR2_REG[1:0] XDATDLY field. For each possible data delay, Figure 18-46 shows when a new frame-synchronization pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 18-46. Proper Positioning of Transmit Frame-sync Pulses



18.4.4.7 Overflow in the Transmitter

The McBSP module indicates a transmitter overflow condition by setting the McBSPi.MCBSPLP_IRQSTATUS_REG[12] XOVFLSTAT bit. This error occurs when sDMA controller or MPU/IVA2.2 subsystem write data to a full XB (this may happen only if the MPU/IVA2.2 subsystem or sDMA controller does not respect the DMA length, does not wait for DMA request or does not check the buffer status before writing data). According to the McBSPi.MCBSPLP_IRQENABLE_REG register settings this condition can generate the McBSPi_IRQ line to be asserted low. Writing 1 to the corresponding bit in status register clears the interrupt.

18.4.5 McBSP DMA Configuration

The McBSP receive and transmit data DMA requests are active after the receive McBSPi.MCBSPLP_SPCR1_REG[0] RRST and transmit McBSPi.MCBSPLP_SPCR2_REG[0] XRST are released. After reset the default DMA threshold (and length) is one.

The receive and transmit DMA requests can be individually disabled by setting the McBSPi.MCBSPLP_RCCR_REG[3] RDMAEN, McBSPi.MCBSPLP_XCCR_REG[3] XDMAEN bits to 0. When disabling the DMA, the DMA request line is de-asserted even if a DMA transfer is pending and the DMA state-machine is not reset.

The DMA threshold and length configuration is done through McBSPi.MCBSPLP_THRSH1_REG and McBSPi.MCBSPLP_THRSH2_REG registers as follows:

- (THRSH1_REG + 1) value represents the required receive DMA request length (the length of the transfer is the same as the threshold value plus one). As long as the RB occupied locations level is above or equal to the THRSH1_REG value + 1, the DMA request is asserted. After transferring the configured (THRSH1_REG + 1) number of words, the receive DMA request is de-asserted and reasserted as soon as the conditions are met again.
- (THRSH2_REG + 1) value represents the required transmit DMA request length (the length of the transfer is the same as the threshold value plus one). As long as the XB free locations level is above or equal to the THRSH2_REG value + 1, the DMA request is asserted. After transferring the configured (THRSH2_REG + 1) number of words, the transmit DMA request is de-asserted and reasserted as soon as the conditions are met again.

NOTE: The MPU/IVA2.2 subsystem can decide not to use the DMA to transfer the data. In this case, the DMA must be disabled (or the DMA request can be ignored by MPU/IVA2.2 subsystem) and the common interrupt line (McBSPi_IRQ) can be used. The McBSPi.MCBSPLP_SPCR1_REG[1] RRDY bit for receive and McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit for transmit will indicate when the threshold values are reached. Also, by reading the receive buffer status McBSPi.MCBSPLP_RBUFFSTAT_REG register and transmit buffer status McBSPi.MCBSPLP_XBUFFSTAT_REG register, the MPU/IVA2.2 subsystem can decide to transfer data even if the threshold is not reached. This mechanism is useful on the last transfer on receive side when the threshold value is bigger than the occupied locations inside the receive buffer and the MPU/IVA2.2 subsystem needs to read this data. Since no interrupt or DMA request is asserted the only option in this case is to read the RB status register value and to transfer the remaining data without using the DMA or interrupt indication.

18.4.6 Multichannel Selection Modes

18.4.6.1 Channels, Blocks, and Partitions

A McBSP channel is a time slot for shifting in/out the bits of one serial word. The McBSP module 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 contain 16 contiguous channels each:

- Block 0: Channels 0–15
- Block 1: Channels 16–31
- Block 2: Channels 32–47
- Block 3: Channels 48–63
- Block 4: Channels 64–79
- Block 5: Channels 80–95
- Block 6: Channels 96–111
- Block 7: Channels 112–127

The blocks are assigned to partitions according to the selected partition mode. In the two-partition mode described in Section 18.4.6.6, 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 Section 18.4.6.4, 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 of each other. For example, it is possible to use two receive partitions (A and B) and eight transmit partitions (A–H).

18.4.6.2 Multichannel Selection

When a McBSP module uses a time-division multiplexed (TDM) data stream while communicating with other McBSP modules or serial devices, the McBSP module 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.

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.

The McBSP module has one receive multichannel selection mode Section 18.4.6.5, and three transmit multichannel selection modes Section 18.4.6.7.

18.4.6.3 Configuring a Frame for Multichannel Selection

Before enabling a multichannel selection mode, make sure you properly configure the data frame:

- Select a single-phase frame (McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit and McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit = 0). Each frame represents a TDM data stream.

- Set a frame length (in McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 field and in McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 field) that includes the highest-numbered channel 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 (RFRLEN1 = 39). If RFRLEN1 = 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.

18.4.6.4 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions (as previously described). If you choose the 8-partition mode (McBSPi.MCBSPLP_MCR1_REG[9] RMCME = 1 for reception, McBSPi.MCBSPLP_MCR2_REG[9] XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H.

In response to a frame-synchronization 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-synchronization pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8-partition mode, the McBSPi.MCBSPLP_MCR1_REG[6:5] RPABLK and McBSPi.MCBSPLP_MCR2_REG[6:5] XPABLK, and McBSPi.MCBSPLP_MCR1_REG[8:7] RPBBLK and McBSPi.MCBSPLP_MCR2_REG[8:7] XPBBLK bit fields are ignored, and the 16 channel blocks are assigned to the partitions as shown in Table 18-20 through Table 18-21. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 18-20. Eight Partitions – Receive Channel Assignment and Control

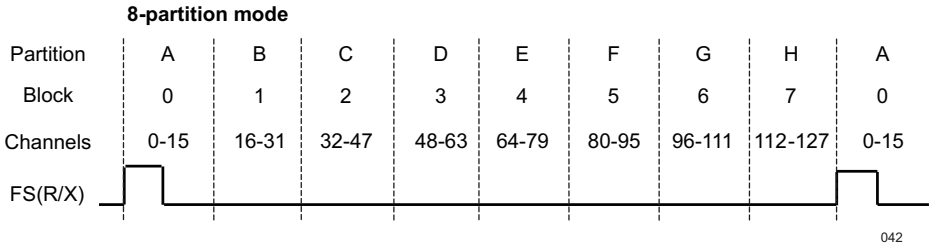
Receive Partition	Assigned Block of Receive Channels	Register Used for Channel Control
A	Block 0: Channels 0 through 15	McBSPi.MCBSPLP_RCERA_REG
B	Block 1: Channels 16 through 31	McBSPi.MCBSPLP_RCERB_REG
C	Block 2: Channels 32 through 47	McBSPi.MCBSPLP_RCERC_REG
D	Block 3: Channels 48 through 63	McBSPi.MCBSPLP_RCERD_REG
E	Block 4: Channels 64 through 79	McBSPi.MCBSPLP_RCERE_REG
F	Block 5: Channels 80 through 95	McBSPi.MCBSPLP_RCERF_REG
G	Block 6: Channels 96 through 111	McBSPi.MCBSPLP_RCERG_REG
H	Block 7: Channels 112 through 127	McBSPi.MCBSPLP_RCERH_REG

Table 18-21. Eight Partitions – Transmit Channel Assignment and Control

Transmit Partition	Assigned Block of Receive Channels	Register Used for Channel Control
A	Block 0: Channels 0 through 15	McBSPi.MCBSPLP_XCERA_REG
B	Block 1: Channels 16 through 31	McBSPi.MCBSPLP_XCERB_REG
C	Block 2: Channels 32 through 47	McBSPi.MCBSPLP_XCERC_REG
D	Block 3: Channels 48 through 63	McBSPi.MCBSPLP_XCERD_REG
E	Block 4: Channels 64 through 79	McBSPi.MCBSPLP_XCERE_REG
F	Block 5: Channels 80 through 95	McBSPi.MCBSPLP_XCERF_REG
G	Block 6: Channels 96 through 111	McBSPi.MCBSPLP_XCERG_REG
H	Block 7: Channels 112 through 127	McBSPi.MCBSPLP_XCERH_REG

Figure 18-47 shows an example of the McBSP using the 8-partition mode. In response to a frame-synchronization pulse, the McBSP module 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 18-47. McBSP Data Transfer in 8-Partition Mode



18.4.6.5 Receive Multichannel Selection Mode

The McBSPi.MCBSPLP_MCR1_REG[0] RMCM bit 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 enabled channels are those selected in the appropriate receive channel enable registers (McBSPi.MCBSPLP_RCERA_REG to McBSPi.MCBSPLP_RCERH_REG). The channels are assigned to the McBSPi.MCBSPLP_RCERA_REG to McBSPi.MCBSPLP_RCERH_REG registers depends on the number of receive channel partitions (2 or 8), as defined by the McBSPi.MCBSPLP_MCR1_REG[9] RMCME bit.
  - If a receive channel is disabled, any bits received in that channel are not transferred to the RB, and as a result, the receiver ready bit (RRDY) is not set. Therefore, no DMA synchronization event is generated and, if the receiver interrupt mode depends on RRDY (McBSPi.MCBSPLP_SPCR1_REG[5:4] RINTM = 0b00), no interrupt is generated.

As an example of how the McBSP module 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 module:

1. Accepts bits shifted in from the mcbspi_dr pin in channel 0
2. Ignores bits received in channels 1–14
3. Accepts bits shifted in from the mcbspi_dr pin in channel 15
4. Ignores bits received in channels 16–38
5. Accepts bits shifted in from the mcbspi_dr pin in channel 39

18.4.6.6 Using Two Partitions (Legacy Only)

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 (McBSPi.MCBSPLP_MCR1_REG[9] RMCME = 0 for reception, McBSPi.MCBSPLP_MCR2_REG[9] XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-synchronization 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-synchronization pulse occurs, the next frame is transferred beginning with the channels in partition A.

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the McBSPi.MCBSPLP_MCR1_REG[6:5] RPABLK field. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register A

(McBSPi.MCBSPLP_RCERA_REG).

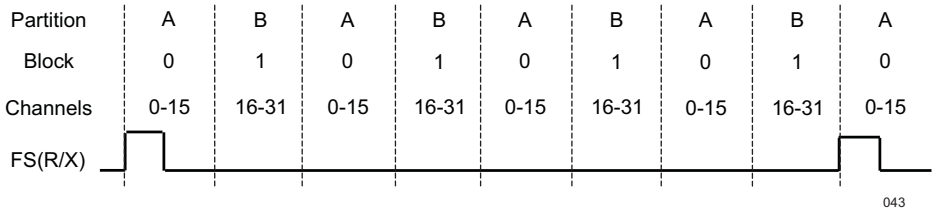
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the McBSPi.MCBSPLP_MCR1_REG[8:7] RPBBLK field. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (McBSPi.MCBSPLP_RCERB_REG).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the McBSPi.MCBSPLP_MCR2_REG[6:5] XPABLK fields. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register A (McBSPi.MCBSPLP_XCERA_REG).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the McBSPi.MCBSPLP_MCR1_REG[8:7] XPBBLK field. In one of the transmit multichannel selection modes, the channels in this partition are controlled by transmit channel enable register B (McBSPi.MCBSPLP_XCERB_REG).

Figure 18-48 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-synchronization pulse, the McBSP module begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 18-48. Alternating Between Partitions A and B Channels



18.4.6.7 Transmit Multichannel Selection Modes

The McBSPi.MCBSPLP_MCR2_REG[1:0] XMCM field determine whether all channels or only selected channels are enabled and unmasked for transmission. The McBSP module has three transmit multichannel selection modes (XMCM = 0b01, XMCM = 0b10, and XMCM = 0b11), which are described in Table 18-22.

Table 18-22. Selecting a Transmit Multichannel Selection Mode With the XMCM Bit Field

XMCM	Transmit Multichannel Selection Mode
0b00	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
0b01	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (McBSPi.MCBSPLP_XCERA_REG/McBSPi.MCBSPLP_XCERH_REG). If enabled, a channel in this mode is also unmasked. The McBSPi.MCBSPLP_MCR2_REG[9] XMCME bit determines whether 32 channels or 128 channels are selectable in the McBSPi.MCBSPLP_XCERA_REG/McBSPi.MCBSPLP_XCERH_REG registers.
0b10	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (McBSPi.MCBSPLP_XCERA_REG/McBSPi.MCBSPLP_XCERH_REG). The McBSPi.MCBSPLP_MCR2_REG[9] XMCME bit determines whether 32 channels or 128 channels are selectable in the McBSPi.MCBSPLP_XCERA_REG/McBSPi.MCBSPLP_XCERH_REG registers.
0b11	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (McBSPi.MCBSPLP_RCERA_REG/McBSPi.MCBSPLP_RCERH_REG). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (McBSPi.MCBSPLP_XCERA_REG/McBSPi.MCBSPLP_XCERH_REG). The McBSPi.MCBSPLP_MCR2_REG[9] XMCME bit determines whether 32 channels or 128 channels are selectable in McBSPi.MCBSPLP_RCERA_REG/McBSPi.MCBSPLP_RCERH_REG registers and McBSPi.MCBSPLP_XCERA_REG/McBSPi.MCBSPLP_XCERH_REG registers.

As an example of how the McBSP module behaves in a transmit multichannel selection mode, suppose that XMCM = 0b01 (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 module:

1. Shifts data to the mcbspi_dx pin in channel 0
2. Places the mcbspi_dx pin in the high impedance state in channels 1–14
3. Shifts data to the mcbspi_dx pin in channel 15
4. Places the mcbspi_dx pin in the high impedance state in channels 16–38
5. Shifts data to the mcbspi_dx pin in channel 39

18.4.6.7.1 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 definitions in Table 18-23 explain the channel control options:

Table 18-23. McBSP Channel Control Options

Enabled channel	A channel that can begin transmission by passing data from the data transmit register (McBSPi.MCBSPLP_DXR_REG) to the XSR through XB.
Masked channel	A channel that cannot complete transmission. The mcbspi_dx pin is held in the high impedance state; data cannot be shifted out on the mcbspi_dx pin. In systems where symmetric transmit and receive provide 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.
Disabled channel	A channel that is not enabled. A disabled channel is also masked. Because no DXR–to–XB copy occurs, the McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit is not set. Therefore, no DMA synchronization event is generated, and if the transmit interrupt mode depends on XRDY (McBSPi.MCBSPLP_SPCR2_REG[5:4] XINTM=00b), no interrupt is generated. The McBSPi.MCBSPLP_SPCR2_REG[2] XEMPTY bit is not affected.
Unmasked channel	A channel that is not masked. Data in the XSR(s) is shifted out on the mcbspi_dx pin.

18.4.6.7.2 Activity on McBSP Pins for Different Values of XMCM

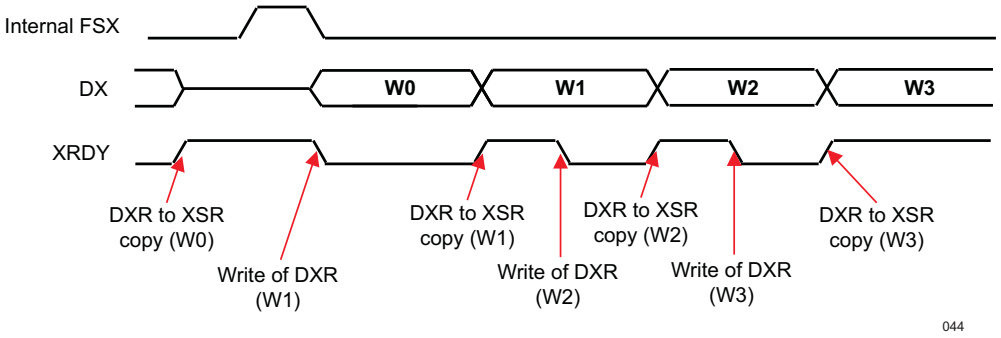
Figure 18-49 shows the activity on the McBSP pins for the various McBSPi.MCBSPLP_MCR2_REG[1:0] XMCM values. In all cases, the transmit frame is configured as follows:

- XPHASE=0: Single–phase frame (required for multichannel selection modes)
- XFRLEN1=0b0000011: 4 words per frame
- XWDLEN1=0b000: 8 bits per word
- XMCME=0: 2–partition mode (only partitions A and B used)

In the case where McBSPi.MCBSPLP_MCR2_REG[1:0] XMCM=0b11, 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.

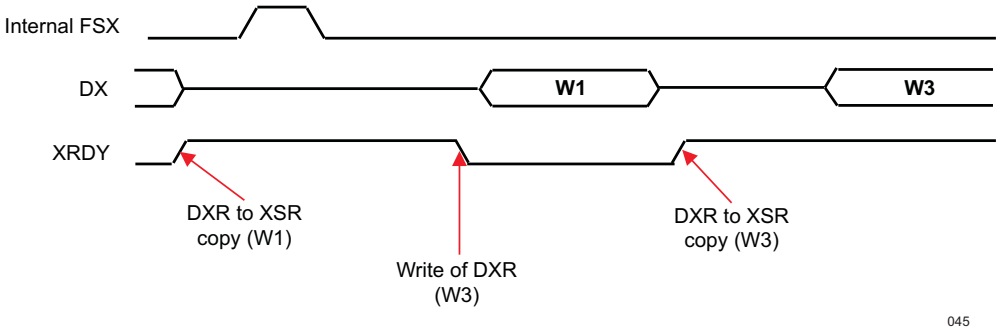
In Figure 18-49, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

Figure 18-49. Activity on McBSP Pins When XMCM=0b00



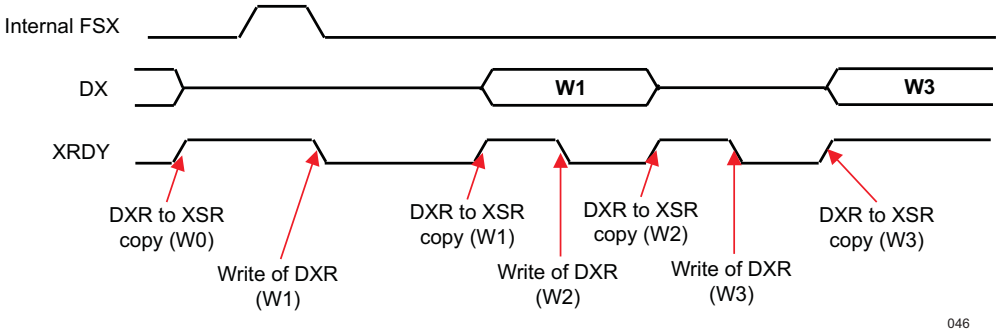
If XMCM = 0b00, all channels are enabled and unmasked. Words W0, W1, W2, and W3 are written to the XB, then, from the XB, there are transferred by mcbspi_dx.

Figure 18-50. Activity on McBSP Pins When XMCM=0b01



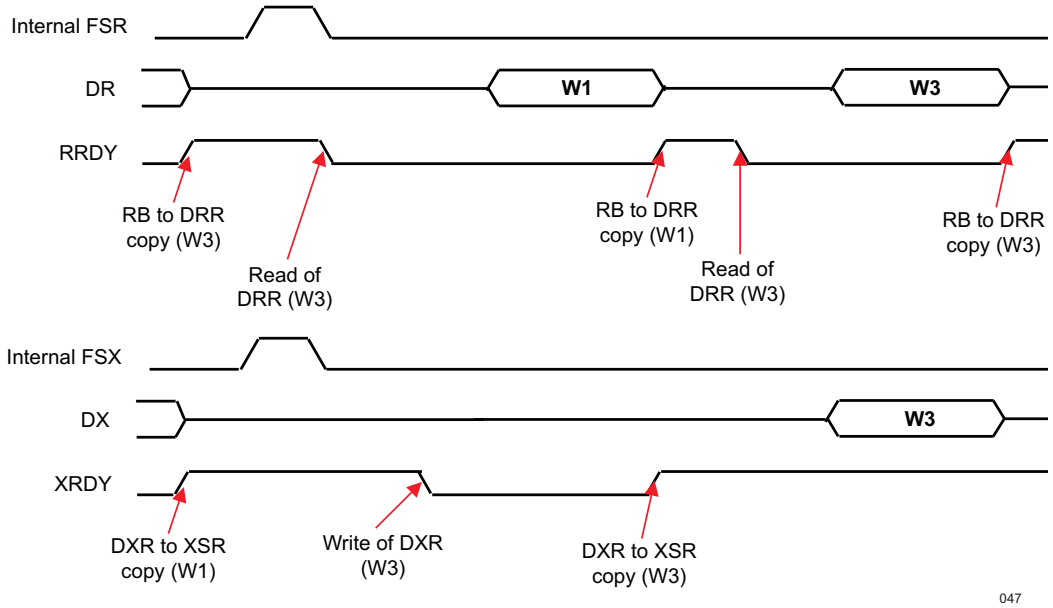
In Figure 18-50 if XMCM = 0b01, XPABLK = 0b00, and XCERA = 0b1010, only channels 1 and 3 are enabled and unmasked. Words W1 and W3 are written to the XB, then, from the XB, there are transferred by mcbspi_dx.

Figure 18-51. Activity on McBSP Pins When XMCM=0b10



In Figure 18-51 if XMCM = 0b10, XPABLK = 0b00, and XCERA = 0b1010, all channels are enabled, only 1 and 3 unmasked. Words W0, W1, W2, and W3 are written to the XB, but only W1 and W3, from the XB, there are transferred by mcbspi_dx.

Figure 18-52. Activity on McBSP Pins When XMCM=0b11



In Figure 18-52 if XMCM = 0b11, RPABLK = 0b00, XPABLK = 0bX, RCERA = 0b1010 and XCERA = 0b1000, channels 1 and 3 are enabled in receive and transmit mode, but only 3 unmasked. Words W1 and W3 are written to the XB, but only W3, from the XB, there are transferred by mcbspi_dx.

18.4.7 SIDETONE Mode (ALP)

18.4.7.1 Introduction

In multimedia-rich mobile communication devices, loopback signals from audio inputs to audio outputs are renamed. A traditional example is the telephone SIDETONE (that is, the telephone user expects to also hear his own voice in the earpiece).

Some of the features using the loopback have strict delay requirements.

It is required that two of the audio input channels be looped back, filtered, and mixed to the corresponding two audio output channels. The SIDETONE mode filters and applies gain to each sample received.

18.4.7.2 SIDETONE Interface

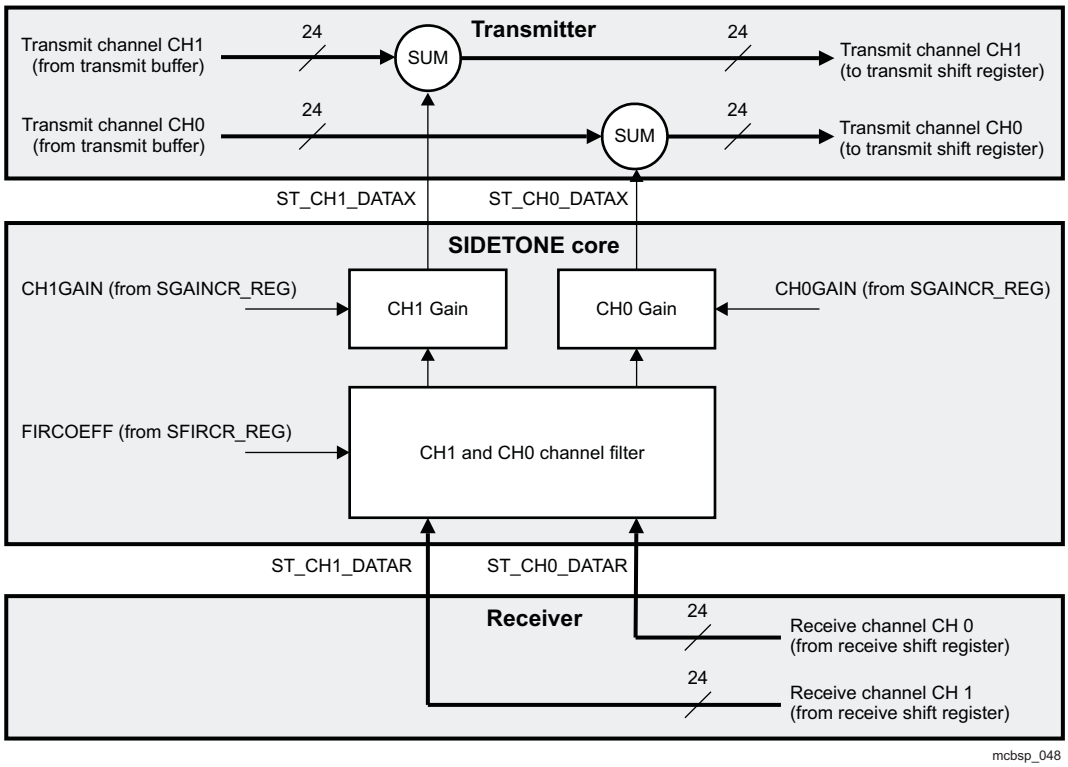
The data from digital microphone, two (out of four) channels, can be configured to be input in the external SIDETONE core. After filtering, the data from digital microphone data is mixed and sent out to the speaker output channels using two (out of eight) configured output channels (separate configuration bits are used). The McBSP module synchronizes the incoming data (filtered by the external SIDETONE core). The transmit and receive part of the McBSP module are not required to operate on the same functional frequency.

The SIDETONE interface offers the following features:

- Send out two 24-bit data channels for the configured SIDETONE received channels (channels can be the same)
- Send out two control signals to indicate to the SIDETONE module that the data is valid (toggle signals which are changing the value from 0 to 1 or 1 to 0 each time a new data is available)
- Receive two 24-bit filtered data channels from the external SIDETONE module and send these channels to the configured transmit channels after mixing the data with the incoming data (from the L4 interface). The sum between the incoming data and SIDETONE loopback data is a saturated sum.
- Receive two control signals to indicate that the SIDETONE module filtered data is available (toggle signals which are changing the value from 0 to 1 or 1 to 0 each time a new data is available).

Figure 18-53 shows the SIDETONE external module data path:

Figure 18-53. SIDETONE Data Path



Before you enable a SIDETONE selection mode, make sure you properly configure the data frame for multichannel and SIDETONE mode:

- Select a single–phase frame (McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit and McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit=0). Each frame represents a TDM data stream.
- Set McBSPi.MCBSPLP_MCR1_REG[0] RMCM=1 to select multichannel mode enable.
- Set a frame length (in McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 bit field and in McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 bit field) that includes the highest–numbered channel to be used (a maximum of 4 channels can be used in this configuration).
- Set a word length (in McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 bit field and in McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 bit field) to be either 16, 24 or 32 (see note).
- Select the input/output channels configured as SIDETONE channels and enable SIDETONE by setting the McBSPi.MCBSPLP_SSELCR_REG[1:0] ICH0ASSIGN/McBSPi.MCBSPLP_SSELCR_REG[6:4] OCH0ASSIGN , McBSPi.MCBSPLP_SSELCR_REG[3:2] ICH1ASSIGN/McBSPi.MCBSPLP_SSELCR_REG[9:7] OCH1ASSIGN fields (2 out of 4 channels external SIDETONE assignment) and McBSPi.MCBSPLP_SSELCR_REG[10] SIDETONEEN bit to 1.

NOTE: Word width in the loop is 24 bits. If input channel word width is less than 24 bits, LSBs of the samples are zero padded. If input channel word width is more than 24 bits, samples are truncated.

Figure 18-54 describes the data exchange protocol between McBSP module and SIDETONE core:

Figure 18-54. McBSP to SIDETONE Data Exchange

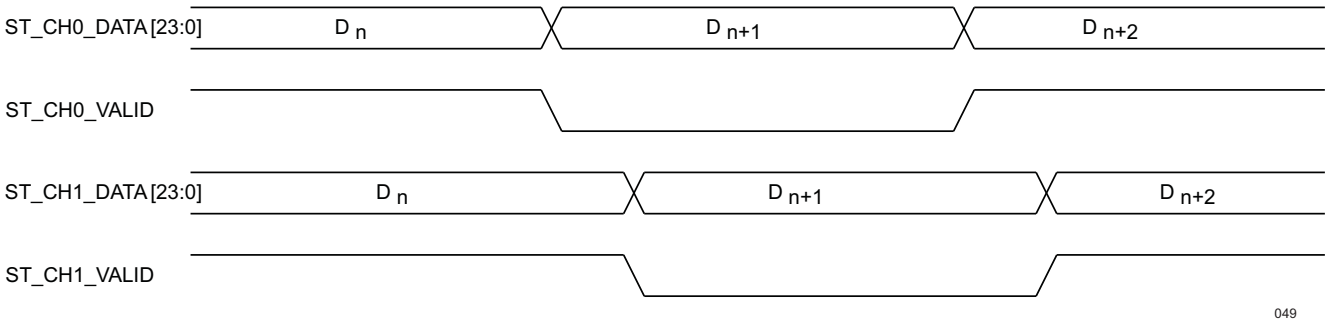
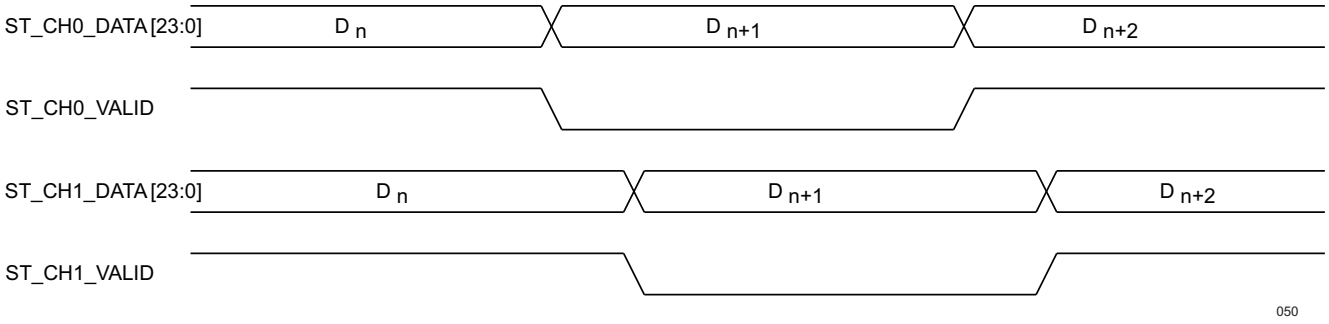


Figure 18-55 describes the data exchange protocol between SIDETONE core and McBSP module:

Figure 18-55. SIDETONE to McBSP Data Exchange



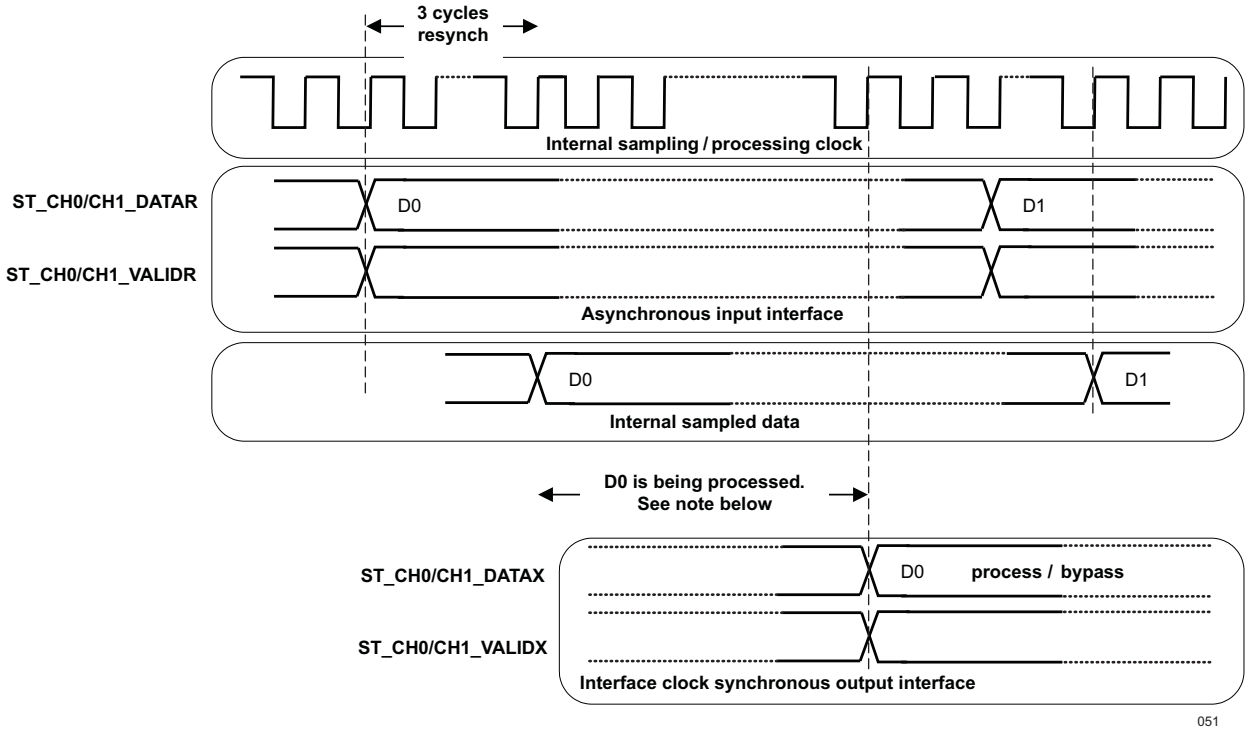
NOTE: To use the same input channel as source for both SIDETONE output channels the McBSPi.MCBSPLP_SSELCR_REG[1:0] ICH0ASSIGN should be equal to McBSPi.MCBSPLP_SSELCR_REG[3:2] ICH1ASSIGN. The McBSP module does not support two input channels to be assigned to only one SIDETONE output channel.

18.4.7.3 Data Processing Path

The SIDETONE core receives the data to be processed through two 24-bits parallel data interfaces, one for each audio channel. When enabled through McBSPi.ST_SSELCR_REG[0] SIDETONEEN bit-field, the module applies the filtering and gain functions to each data item (frame) and outputs it through two similar 24-bits parallel data interfaces.

Figure 18-56 below describes how the data is sampled and output through the dedicated data interfaces.

Figure 18-56. SIDETONE Processed Data Interfaces



NOTE: The processing time needed is 132 clock cycles (SIDETONE enabled). The total number of cycles needed for outputting the new processed data upon arrival of a sample is maximum 135 (including synchronizations). When not enabled, the data response takes maximum 5 clock cycles.

The ST_CH0_VALIDR and ST_CH1_VALIDR are 1 bit inputs and their toggling information is used to signal that new data is valid for sampling from ST_CH0_DATAR / ST_CH1_DATAR.

The ST_CH0_VALIDX and ST_CH1_VALIDX are 1 bit outputs and their toggling information is used to signal that new data is valid on the output data bus ST_CH0_DATAX / ST_CH1_DATAX.

18.4.7.4 Data Processing

The processing consists in 2 stages filtering and applying gain to signal. The whole process takes 132 clock cycles.

If a new frame comes too early causing the interrupt assertion, the current frame will be completely processed and this new frame will be ignored. Consequent frames will also be ignored until the current frame processing has ended.

18.4.7.4.1 Filtering

A 128 length FIR filter scheme is used. The module must process the two channels in parallel so two instances of filter will work in parallel sharing only the same coefficients.

Samples data are signed values in interval (-1..1) in Q23 format, negative values are expressed in 2's complement. Coefficients are also signed values in interval (-1..1) in Q15 format, negative values are expressed in 2's complement too.

The module handles overflows in the sums of the product but the user should choose the coefficients with the sum of magnitudes in absolute value is smaller than 1, otherwise the user must ensure that gain applying brings the samples value below |1|, to avoid saturation.
|C0| + |C1| + … + |C127| 1

CAUTION

The coefficients cannot be loaded while the filtering is active (SIDETONE enabled).

18.4.7.4.2 Applying Gain

Each output sample will be multiplied with the gain value specified in the McBSPi.ST_SGAINCR_REG. The gain is independent for each channel and can be modified anytime through L4-interface with immediate effect. Gain values are in the interval (-2..2) in Q1.14 format, negative values are expressed in 2's complement.

The user must choose the gain's value according to the magnitude of the samples to avoid overflow in the final product between the FIR data and gain value. If an overflow occurs, the output data is saturated.

18.4.7.4.3 Enabling SIDETONE

When the SIDETONE operation is enabled (McBSPi.ST_SSELCR_REG[0] SIDETONEEN is 1), the module starts collecting samples received through the data interface without returning any sample or toggling any of the ST_CH0/CH1_VALIDX outputs. The first 127 samples for each channel are only accumulated, no processed data is provided. The first processed frame comes after receiving the 128th sample (with the specific delay).

When the SIDETONE operation is disabled, data is output through the data interface as it comes on the data input interface with the resynchronization latency of maximum 5 clock cycles. Re-enabling it requires waiting another 128 samples before providing new processed samples (each transition of McBSPi.ST_SSELCR_REG[0] SIDETONEEN from 0 to 1 triggers this initialization process).

NOTE: After reset, the McBSPi.ST_SSELCR_REG[0] SIDETONEEN bit is 0.

18.4.7.4.4 FIR Accuracy

All the arithmetic inside the module is performed without any truncation/saturation until the last stage – the gain applying, where the result of the product between gain and FIR value is saturated to +/-1 if overflow occurs and the LSBs are truncated.

18.4.7.5 Interrupt Operation

The SIDETONE core has a single interrupt line and a single event that may trigger the interrupt. The event (an error one) occurs when the input interface data rate overflows the processing ability of the module. As described in the data processing path section, the SIDETONE core completes a frame processing in 132 cycles. If the input frame rate for any channel exceeds 1 frame/132 x interface clock period while SIDETONE is enabled , the McBSPi.ST_IRQSTATUS_REG[0] OVRRERROR bit in is set and, if McBSPi.ST_IRQENABLE_REG[0] OVRRERROREN bit is set to 1, the interrupt line is asserted.

18.5 McBSP Basic Programming Model

This section describes the programming model of typical McBSP module and SIDETONE core applications.

18.5.1 McBSP Core

CAUTION

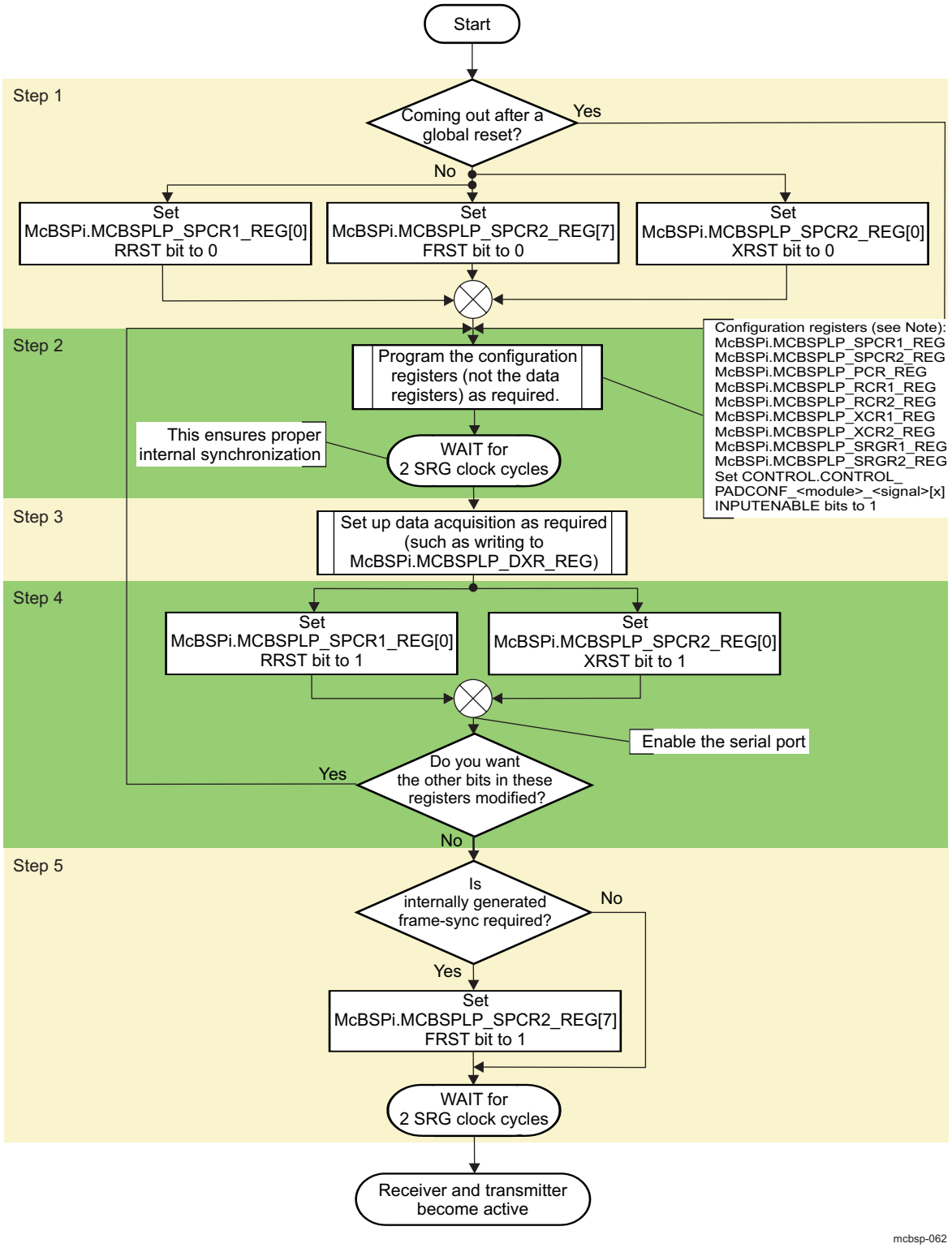
For all descriptions in this section, the McBSPi.MCBSPLP_XCCR_REG[11] XFULL_CYCLE bit and the McBSPi.MCBSPLP_RCCR_REG[11] RFULL_CYCLE bit are their reset value (XFULL_CYCLE bit=0 and RFULL_CYCLE bit=1).

18.5.1.1 McBSP Initialization Procedure

This procedure for reset/initialization can be applied in general when the Receiver or Transmitter has to be reset during its normal operation, and also when the Sample Rate Generator is not used for either operation.

Figure 18-57 shows the serial port initialization procedure for master mode.

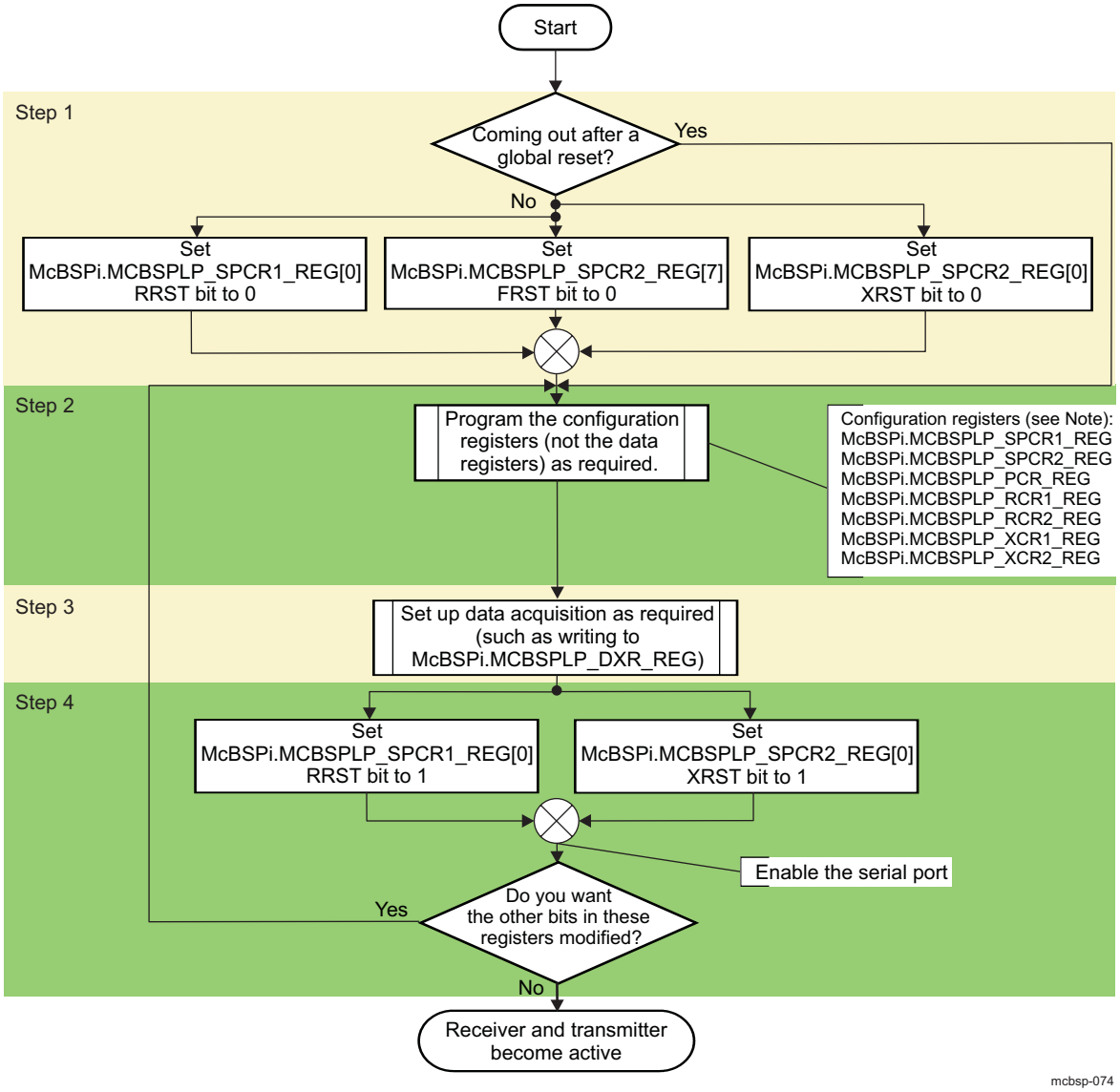
Figure 18-57. Flow Diagram of McBSP Initialization Procedure for Master Mode



Alternatively, on write (step 1 or 4), the transmitter and receiver can be placed in or taken out of reset by modifying the McBSPi.MCBSPLP_SPCR2_REG[0] XRST bit and the McBSPi.MCBSPLP_SPCR1_REG[0] RRST bit, respectively.

Figure 18-58 shows the flow diagram of the McBSP initialization procedure for slave mode.

Figure 18-58. Flow Diagram of McBSP Initialization Procedure for Slave Mode



NOTE:

- The necessary duration of the active-low period of XRST or RRST is at least two CLKR/CLKX cycles.
- The appropriate bits in serial port configuration registers (McBSPi.MCBSPLP_SPCR1_REG, McBSPi.MCBSPLP_SPCR2_REG, McBSPi.MCBSPLP_PCR_REG, McBSPi.MCBSPLP_RCR1_REG, McBSPi.MCBSPLP_RCR2_REG, McBSPi.MCBSPLP_XCR1_REG, McBSPi.MCBSPLP_XCR2_REG, McBSPi.MCBSPLP_THRSH2_REG, McBSPi.MCBSPLP_XCCR_REG, McBSPi.MCBSPLP_SYSCONFIG_REG, McBSPi.MCBSPLP_SRGR1_REG, and McBSPi.MCBSPLP_SRGR2_REG) should only be modified when the affected portion of the serial port is in its reset state.
- In most cases, the data transmit register (McBSPi.MCBSPLP_DXR_REG) should be loaded by the MPU/IVA2 subsystem or the sDMA controller only when the transmitter is enabled (McBSPi.MCBSPLP_SPCR2_REG[0] XRST = 1). An exception to this rule is when these registers are used for loopback internal data.
- The bits of the channel control registers (McBSPi.MCBSPLP_MCR1_REG, McBSPi.MCBSPLP_MCR2_REG, McBSPi.MCBSPLP_RCER{A-H}_REG and McBSPi.MCBSPLP_XCER{A-H}_REG) can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode.
- The SRG is reset by setting McBSPi.MCBSPLP_SPCR2_REG[6] GRST bit to 0.
- It is not necessary to wait if SRG is not used.
- Modification on-the-fly has no effect if a reset is not done first.

18.5.1.2 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the Sample Rate Generator:

- Place the McBSP Sample Rate Generator in RESET.

  During a Global RESET, the Sample Rate Generator, the Receiver, and the Transmitter reset bits (GRST, RRST, and XRST) are automatically forced to '0'. Otherwise, during normal operation, the Sample Rate Generator can be reset by making McBSPi.MCBSPLP_SPCR2_REG[6] GRST=0, provided that CLKG and/or FSG internal signal is not used by any portion of the McBSP module. Depending on the system needs, the software may also to reset the Receiver (McBSPi.MCBSPLP_SPCR1_REG[0] RRST=0) and reset the Transmitter (McBSPi.MCBSPLP_SPCR2_REG[0] XRST bit =0).
- Program the registers that affect the Sample Rate Generator.

  Program the Sample Rate Generator registers (McBSPi.MCBSPLP_SPCR1_REG and McBSPi.MCBSPLP_SPCR2_REG) as required for your application. Refer to Figure 18-59.

  If necessary, other control registers can be loaded with desired values provided the respective portion of the McBSP module (the Receiver or Transmitter) is in reset. Table 18-24 presents the McBSP configuration when one of the clock sources is selected, but others registers can be impacted in function of the user application.

Table 18-24. McBSP Configuration in Function of the SRG Clock Source Selected

SRG Clock Source Selected	Module Configuration	McBSPi.MCBSPLP_PCR_REG Configuration			
		CLKRM bit [1]	CLKXM bit	FSRM bit [1]	FSXM bit
McBSPi_ICLK or CLKS	Master Transmitter and Slave Receiver	0	1	0	1
	Master Receiver and Slave Transmitter	1	0	1	0
	Master Transmitter and Receiver	1	1	1	1
CLKR	Master Transmitter and Slave Receiver	0	1	0	1

(1) This configuration is correct if McBSPi.MCBSPLP_XCCR_REG[5] DLB bit=0x0. When the DLB bit is set to 1, the CLKR clock (not the mcbspi_clkr pin) is driven by the CLKX clock, which is based on the CLKXM bit.

Table 18-24. McBSP Configuration in Function of the SRG Clock Source Selected (continued)

SRG Clock Source Selected	Module Configuration	McBSPi.MCBSPLP_PCR_REG Configuration			
		CLKRM bit [1]	CLKXM bit	FSRM bit [1]	FSXM bit
CLKX	Master Receiver and Slave Transmitter	1	0	1	0

After the Sample Rate Generator registers are programmed, wait for 2 CLKSRG cycles. This ensures proper synchronization internally.

- Enable the Sample Rate Generator (take it out of reset).

  Set McBSPi.MCBSPLP_SPCR2_REG[6] GRST bit to1 to enable the Sample Rate Generator.

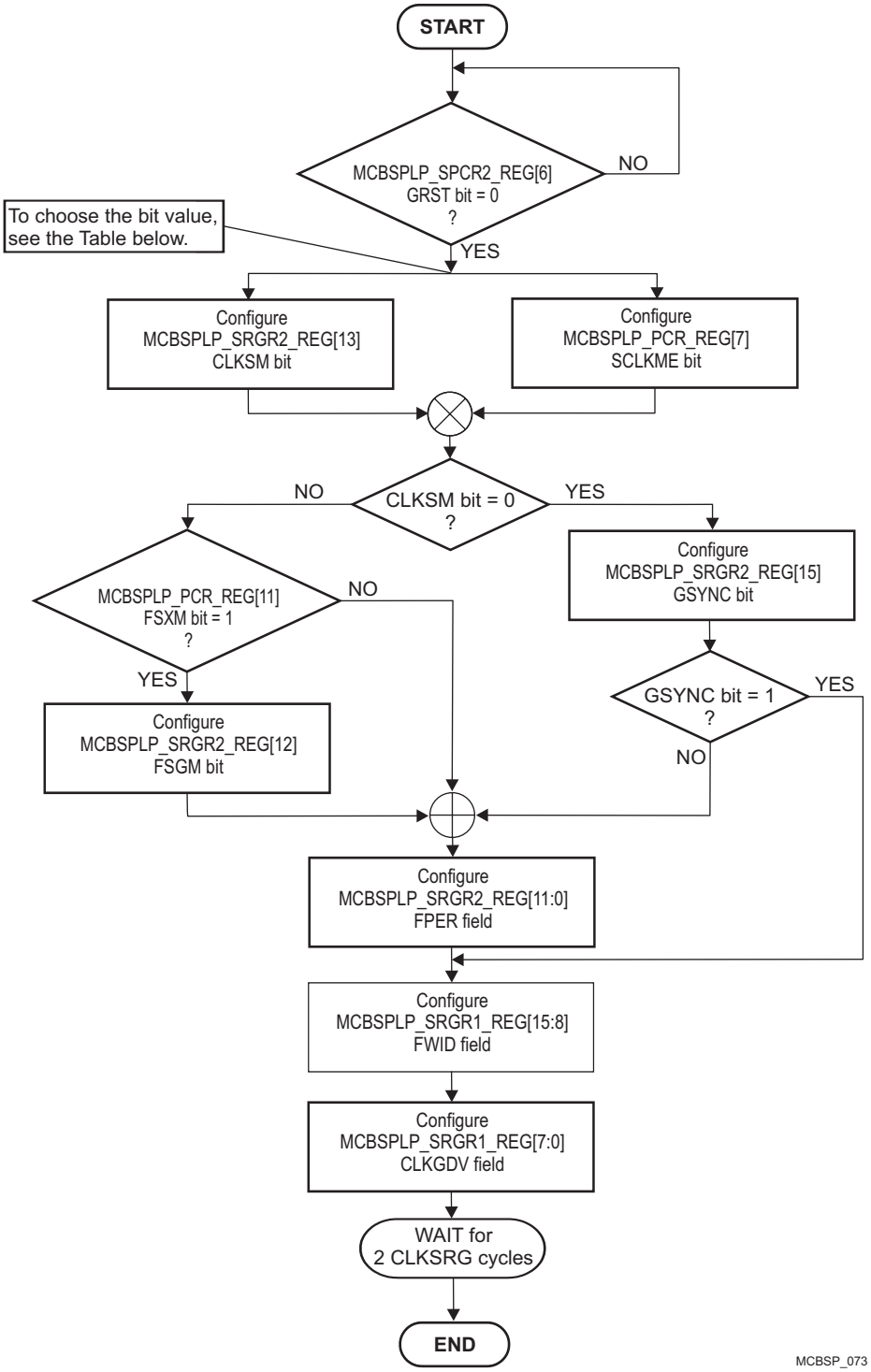
  After the Sample Rate Generator is enabled, wait for 2 CLKG cycles for the Sample Rate Generator logic to stabilize.

  On the next rising edge of CLKSRG, the CLKG signal transitions to '1' and starts clocking with a frequency equal to (input clock frequency)/(CLKGDV + 1).

- If necessary, enable the receiver and/or the transmitter.

  If necessary, remove the receiver and/or transmitter from reset by setting McBSPi.MCBSPLP_SPCR1_REG[0] RRST bit and/or McBSPi.MCBSPLP_SPCR2_REG[0] XRST =1.

Figure 18-59. Flow Diagram for the SRG Registers Programmation



The input clock is selected with the McBSPi.MCBSPLP_PCR_REG[7] SCLKME bit and the McBSPi.MCBSPLP_SRGR2_REG[13] CLKSM bit in one of the following configurations (see Table 18-25):

Table 18-25. Input Clock Selection for Sample Rate Generator

SCLKME bit	CLKSM bit	Input Clock for Sample Rate Generator
0	0	Signal on mcbsp_clks pin
0	1	McBSPi_ICLK clock
1	0	Signal on mcbsp_clkr pin
1	1	Signal on mcbsp_clkx pin

18.5.1.3 Data Transfer DMA Request Configuration

To configure the McBSP receive/transmit data DMA requests (McBSPi_DMA_RX and McBSPi_DMA_TX), perform the following procedure:

- Write the receive McBSPi.MCBSPLP_THRSH1_REG register with the required receive DMA request length (the length of the transfer is the same as the threshold value + 1). As long as the RB occupied locations level is above or equal to the THRSH1_REG value + 1, the DMA request will be asserted. After transferring the configured THRSH1_REG value + 1 number of words, the receive DMA request will be de-asserted and reasserted as soon as the conditions are met again.

NOTE: In case of a number of transfers that exceed the number of the programmed DMA length the McBSP module will respond to the command, and will perform the transfer regardless of the receive buffer empty condition. When the receive buffer is empty a data transfer access will trigger a receive underflow interrupt, if enabled by McBSPi.MCBSPLP_IRQENABLE_REG[4] RUNDFLEN bit.

- Write the transmit McBSPi.MCBSPLP_THRSH2_REG register with the required transmit DMA request length (the length of the transfer is the same as the threshold value + 1). As long as the XB free locations level is above or equal to the THRSH2_REG value + 1, the DMA request will be asserted. After transferring the configured THRSH2_REG value + 1 number of words, the transmit DMA request will be de-asserted and reasserted as soon as the conditions are met again.

NOTE: In case of a number of transfers that exceed the number of the programmed DMA length the McBSP module will respond to the command, and will perform the transfer regardless of the transmit buffer full condition. When the transmit buffer is full a data transfer access will trigger a transmit overflow interrupt, if enabled by McBSPi.MCBSPLP_IRQENABLE_REG[12] XOVFLEN bit.

18.5.1.4 Interrupt Configuration

The McBSP module offers two interrupt schemes:

- L4 compliant interrupt request scheme using a common receive/transmit interrupt request line
- The legacy interrupt compliant scheme using 3 interrupt lines: one for receive, one for transmit and the common interrupt line.

18.5.1.4.1 L4-Compliant Interrupt Line

The L4-compliant interrupt line can be configured by using the McBSPi.MCBSPLP_IRQENABLE_REG register. When the McBSPi.MCBSPLP_IRQSTATUS_REG bit is set and the corresponding McBSPi.MCBSPLP_IRQENABLE_REG bit is set to one, the interrupt line is asserted. Writing one to a bit in McBSPi.MCBSPLP_IRQSTATUS_REG register clears the bit.

There are several conditions, which can be configured to generate an interrupt as follows:

1. Transmit buffer empty at end of frame (McBSPi.MCBSPLP_IRQSTATUS_REG[14] XEMPTYEOF bit is set to one when a complete frame was transmitted and the transmit buffer is empty .
2. Transmit buffer overflow (McBSPi.MCBSPLP_IRQSTATUS_REG[12] XOVFLSTAT bit is set to one when transmit buffer overflow; the data written while overflow condition is discarded).
3. Transmit buffer underflow (McBSPi.MCBSPLP_IRQSTATUS_REG[11] XUNDFLSTAT bit is set to one

when the transmit data buffer is empty, new data needs to be transmitted).

4. Transmit buffer threshold reached (McBSPi.MCBSPLP_IRQSTATUS_REG[10] XRDY bit is set to one when the transmit buffer free locations are equal or above the [THRSH2_REG + 1] value).
5. Transmit end of frame (McBSPi.MCBSPLP_IRQSTATUS_REG[9] XEOF is set to one when a complete frame was transmitted).
6. Transmit frame synchronization (McBSPi.MCBSPLP_IRQSTATUS_REG[8] XFSX bit is set to one when a new transmit frame synchronization is asserted).
7. Transmit frame synchronization Error (McBSPi.MCBSPLP_IRQSTATUS_REG[7] XSYNCERR is set to one when a transmit frame synchronization error is detected).
8. Receive buffer overflow (McBSPi.MCBSPLP_IRQSTATUS_REG[5] ROVFLSTAT bit is set to one when receive buffer overflow; the data which is written while overflow condition is discarded).
9. Receive buffer underflow (McBSPi.MCBSPLP_IRQSTATUS_REG[4] RUNDFLSTAT bit is set to one when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined).
10. Receive buffer threshold Reached (McBSPi.MCBSPLP_IRQSTATUS_REG[3] RRDY bit is set to one when the receive buffer occupied locations are equal or above the [THRSH1_REG + 1] value).
11. Receive end of frame (McBSPi.MCBSPLP_IRQSTATUS_REG[2] REOF is set to one when a complete frame was received).
12. Receive frame synchronization (McBSPi.MCBSPLP_IRQSTATUS_REG[1] RFSR bit is set to one when a new receive frame synchronization is asserted).
13. Receive frame synchronization error (McBSPi.MCBSPLP_IRQSTATUS_REG[0] RSYNCERR is set to one when a receive frame synchronization error is detected).

18.5.1.4.2 Legacy Interrupt Line

McBSPi_IRQ_TX and McBSPi_IRQ_RX are legacy interrupts. Not to be used for new development. McBSPi_IRQ (common interrupt line) should be preferred.

18.5.1.4.2.1 Set the receive interrupt line (legacy only)

The McBSPi.MCBSPLP_SPCR1_REG[5:4] RINTM bit field determines which event generates a receive interrupt request, McBSPi_IRQ_RX, to the MPU/IVA2.2 subsystem.

The receive interrupt informs the MPU/IVA2.2 subsystem of changes to the serial port status. Four options exist for configuring this interrupt.

- RINTM=0b00: The receive interrupt generated when the McBSPi.MCBSPLP_SPCR1_REG[1] RRDY bit changes from 0 to 1. Interrupt on every serial word by tracking the McBSPi.MCBSPLP_SPCR1_REG[1] RRDY bit. Regardless of the value of RINTM, RRDY bit can be read to detect the RRDY=1 condition.
- RINTM = 0b01: The receive interrupt generated by an end-of-frame condition in the receive multichannel selection mode. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.
- RINTM = 0b10: The receive interrupt generated by a new receive frame-synchronization pulse. Interrupt on detection of receive frame-synchronization pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the McBSPi_ICLK clock and sending it to the MPU/IVA2.2 subsystem via the receive interrupt .
- RINTM = 0b11: The receive interrupt generated when McBSPi.MCBSPLP_SPCR1_REG[3] RSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see Section 18.4.4.3.

The McBSP module also provides a common interrupt line McBSPi_IRQ, which can be used by setting the McBSPi.MCBSPLP_IRQENABLE register. All the above settings have equivalent enable bits in the McBSPi.MCBSPLP_IRQENABLE register to enable the common interrupt line:

- RRDYEN is equivalent with RINTM = 0 setting

- REOFEN is equivalent with RINTM = 0b01setting (the interrupt is generated by an end-of-frame condition regardless of the multichannel selection mode)
- RFSREN is equivalent with RINTM = 0b10 setting
- RSYNCERREN is equivalent with RINTM = 0b11 setting

This interrupt line has its own status register, McBSPi.MCBSPLP_IRQSTATUS_REG.

18.5.1.4.2.2 Set the transmit interrupt line (legacy only)

The McBSPi.MCBSPLP_SPCR2_REG[5:4] XINTM bit field determines which event generates a transmit interrupt request (McBSPi_IRQ_TX) to the MPU/IVA2.2 subsystem.

The transmit interrupt informs the MPU/IVA2.2 subsystem of changes to the serial port status. Four options exist for configuring this interrupt.

- XINTM = 0b00: The transmit interrupt generated when the McBSPi.MCBSPLP_SPCR2_REG[1] XRDY bit changes from 0 to 1. Interrupt on every serial word by tracking the XRDY bit. Regardless of the value of XINTM, XRDY bit can be read to detect the XRDY=1 condition.
- XINTM = 0b01: The transmit interrupt generated by an end-of-frame condition in the transmit multichannel selection mode. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.
- XINTM = 0b10: The transmit interrupt generated by a new transmit frame-synchronization pulse. Interrupt on detection of transmit frame-synchronization pulses. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-synchronization pulse to the McBSPi_ICLK clock and sending it to the MPU/IVA2.2 subsystem via transmit interrupt.
- XINTM = 0b11: The transmit interrupt generated when McBSPi.MCBSPLP_SPCR2_REG[3] XSYNCERR is set. Interrupt on frame-synchronization error. Regardless of the value of XINTM, XSYNCERR bit can be read to detect this condition. For information on using XSYNCERR bit, see Section 18.4.4.6.

The McBSP module provides also a common interrupt line McBSPi_IRQ, which can be used by setting the McBSPi.MCBSPLP_IRQENABLE register. All the above settings have equivalent enable bits in the McBSPi.MCBSPLP_IRQENABLE register to enable the common interrupt line:

- XDYEN is equivalent with XINTM = 0b00 setting
- XEOFEN is equivalent with XINTM = 0b01setting (the interrupt is generated by an end-of-frame condition regardless of the multichannel selection mode)
- XFSXEN is equivalent with XINTM = 0b10 setting
- XSYNCERREN is equivalent with XINTM = 0b11 setting

This interrupt line has its own status register, McBSPi.MCBSPLP_IRQSTATUS_REG.

18.5.1.5 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1. Place the McBSP receiver in reset .
2. Program the McBSP registers for the desired receiver operation.
3. Take the receiver out of reset.

These 3 steps are detailed in the following subsections.

18.5.1.5.1 Resetting (Step 1) and Enabling (Step 3) the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take it out of reset).

The serial port can be reset in the following 2 ways:

1. A global reset places the receiver, transmitter, and SRG in reset. When the device reset is removed, McBSPi.MCBSPLP_SPCR2_REG[6] GRST, McBSPi.MCBSPLP_SPCR2_REG[7] FRST, McBSPi.MCBSPLP_SPCR1_REG[0] RRST and McBSPi.MCBSPLP_SPCR2_REG[0] XRST bits = 0, which keeps the entire serial port in the reset state.

2. The serial port receiver can be reset directly using the McBSPi.MCBSPLP_SPCR1_REG[0] RRST bit. If the SRG needs to be used, SRG must be reset directly using the McBSPi.MCBSPLP_SPCR2_REG[6] GRST bit. Similar operations with frame synchronization generator also require using the McBSPi.MCBSPLP_SPCR2_REG[7] FRST bit when the frame-sync signal must be generated.

To enable the receiver, the preceding bits, cleared to 0, must be set to 1.

18.5.1.5.2 Programming the McBSP Registers for the Desired Receiver Configuration (Step 2)

Figure 18-60 and Figure 18-61 lists important tasks to be performed when the software is configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields.

Figure 18-60. Important Tasks to Configure the McBSP Receiver (Part 1)

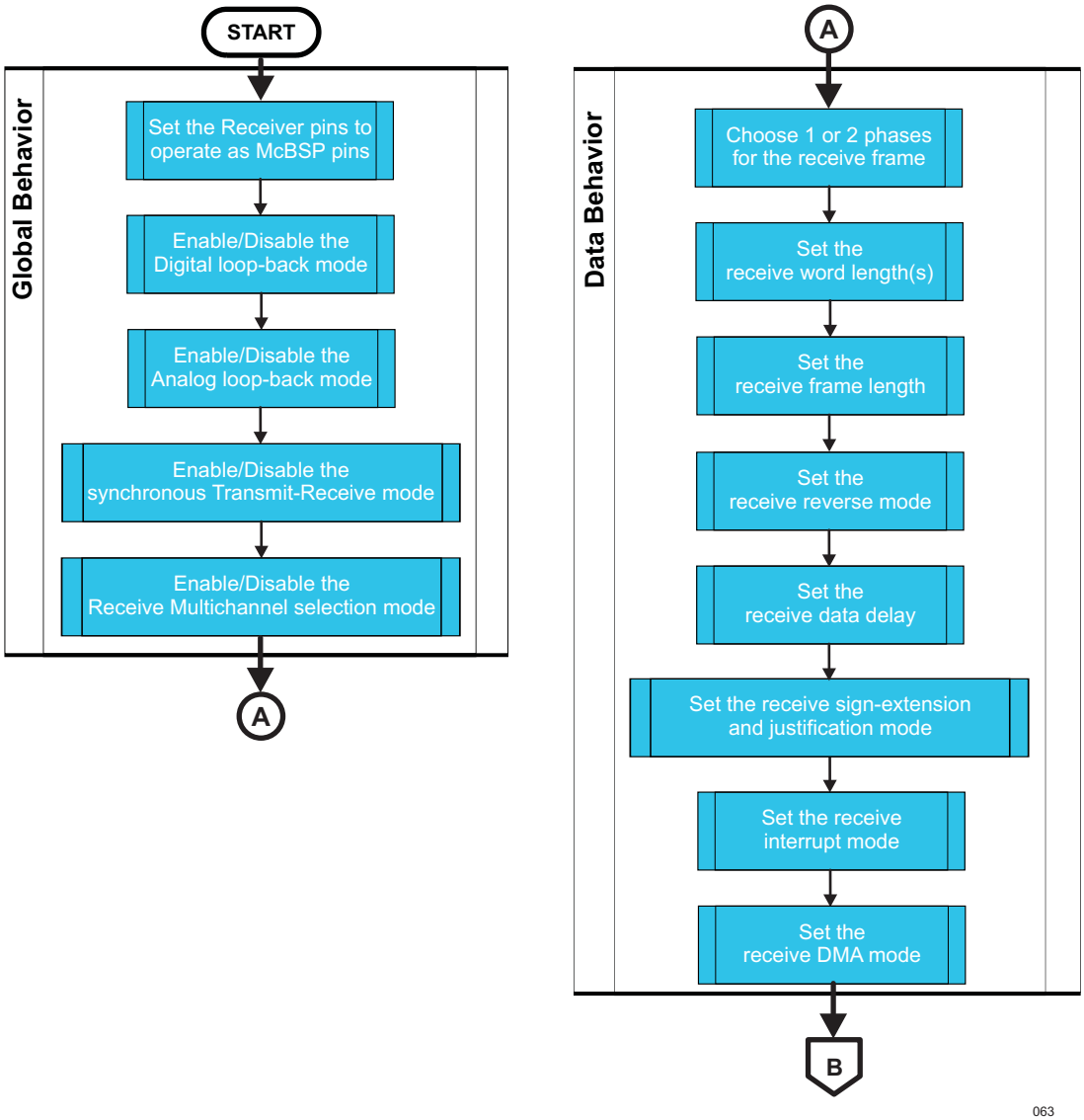
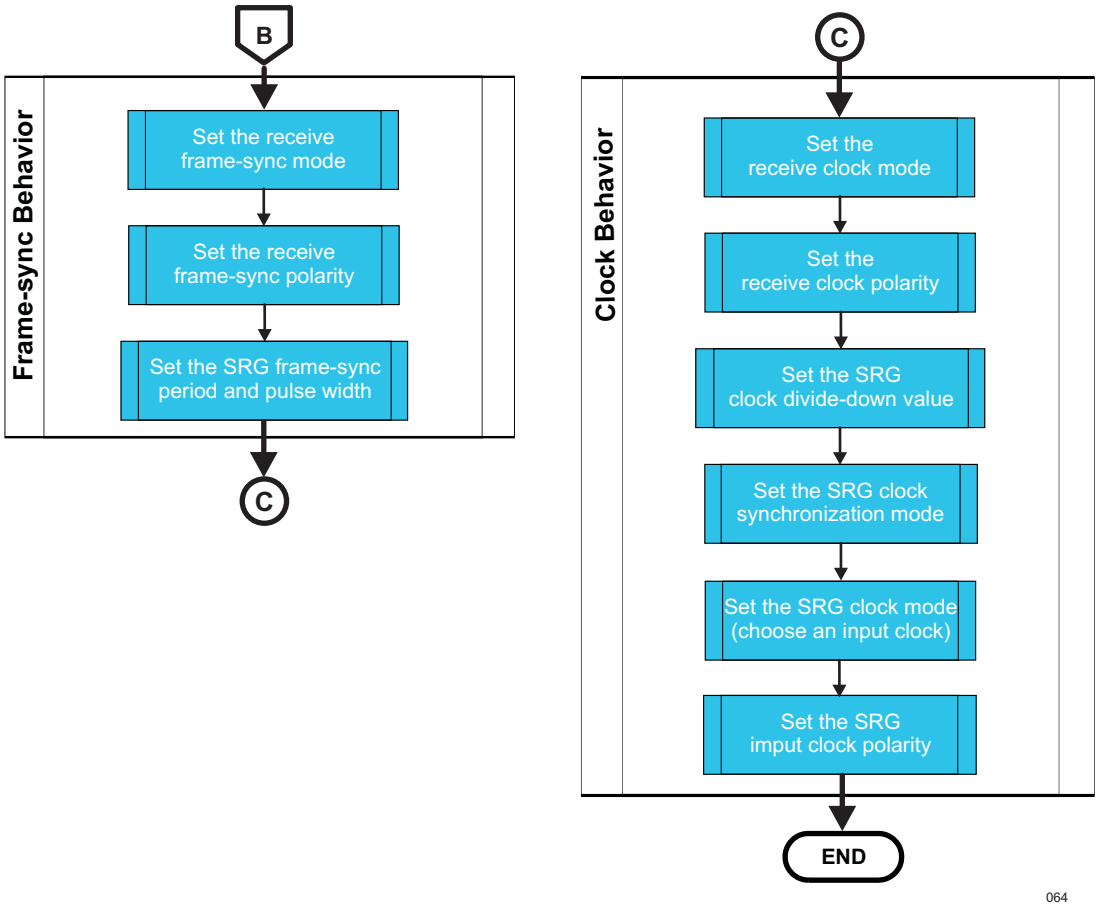


Figure 18-61. Important Tasks to Configure the McBSP Receiver (Part 2)



18.5.1.5.2.1 Global Behavior

18.5.1.5.2.1.1 Set the Receiver Pins to Operate as McBSP Pins

The McBSPi.MCBSPLP_PCR_REG[12] RIOEN bit determines whether the receiver pins are McBSP pins (RIOEN bit=0) or general–purpose I/O pins (RIOEN bit=1).

Please refer to Section 18.5.1.7 which describes how to use McBSP pins as GPIO pins.

18.5.1.5.2.1.2 Enable/Disable the Digital Loop Back Mode

The McBSPi.MCBSPLP_XCCR_REG[5] DLB bit determines whether the digital loopback mode is on or off.

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals:

- DR signal is connected on DX signal to receive the transmitted data
- FSR is connected to FRX output signal
- CLKR is connected to the CLKX output signal

This mode allows testing of serial port; the McBSP module receives the data it transmits. This loopback mode is not done through pads, all output signals being disabled.

NOTE: That in digital loopback mode the SRG and the frame synchronization generator need to be enabled to generate the CLKX and FSX signals.

18.5.1.5.2.1.3 Enable/Disable the Analog Loopback Mode

The McBSPi.MCBSPLP_SPCR1_REG[15] ALB bit determines whether the analog loopback mode is on or off.

In the analog loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit loop back signals:

- DR is connected to transmit loop back data on DX input pin
- FSR is connected to FRX input pin
- CLKR is connected to the CLKX input pin

This Analog Loopback mode is also done to test the Input/Output buffers, through pads.

18.5.1.5.2.1.4 Enable/disable the synchronous transmit-receive mode (McBSP1 only)

The control registers of the System Control Module is used to configure the synchronous transmit-receive mode.

The MCBSP1_CLKR bit of the CONTROL_DEVCONF0[3] register is used to select the McBSP1 module CLKR signal source:

- When set to '0', the CLKR source is from the CLKR input signal
- When set to '1', the CLKR source is from the CLKX input signal

The MCBSP1_FSR bit of the CONTROL_DEVCONF0[4] register is used to select the McBSP1 module FSR signal source:

- When set to '0', the FSR source is from the FSR input signal
- When set to '1', the FSR source is from the FSX input signal

18.5.1.5.2.1.5 Enable/Disable the Receive Multichannel Selection Mode

The McBSPi.MCBSPLP_MCR1_REG[0] RMCM bit determines whether the receive multichannel selection mode is on or off.

For further details, see Section 18.4.6.5.

18.5.1.5.2.2 Data Behavior

18.5.1.5.2.2.1 Choose 1 or 2 Phases for the Receive Frame

The McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit determines whether the receive data frame has one (Single phase) or two phases (Dual phase). When dual-phase is selected the number of words per phase must be set to one.

18.5.1.5.2.2.2 Set the Receive Word Length(s)

The McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 and McBSPi.MCBSPLP_RCR2_REG[7:5] RWDLEN2 bit fields determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

If a single-phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame.

If a dual–phase frame is selected, RWDLEN1 and RWDLEN2 must be set to select the both length. These both bits can have values different.

18.5.1.5.2.2.3 Set the Receive Frame Length

The receive frame length is the number of serial words in the receive frame.

The McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 and McBSPi.MCBSPLP_RCR2_REG[14:8] RFRLEN2 bit fields determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

If a dual-phase frame is selected (RPHASE=1), the frame length must be two words(one word for phase 1 plus the one word for phase 2). Others values must not be used.

The 7–bit RFRLEN1 field allows up to 128 words per phase when single-phase frame. See Table 18-26 below for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the RFRLEN fields with [W - 1], where W represents the number of words per phase. For example, to get a phase length of 128 words in phase 1, load 127 words into RFRLEN1.

Table 18-26. How to Calculate the Length of the Receive Frame

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Don't care	(RFRLEN1value +1) words
1	RFRLEN1 = 0	RFRLEN2 = 0	2 words

18.5.1.5.2.2.4 Set the Receive Reverse Mode

The McBSPi.MCBSPLP_RCR2_REG[4:3] RREVERSE bit field determines whether reverse (LSB first) data transfer option is chosen for McBSP reception.

For further information about reverse mode, see Section 18.4.2.2.

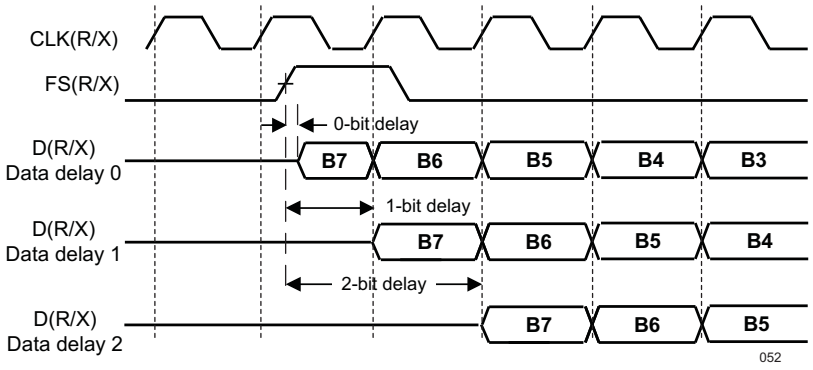
18.5.1.5.2.2.5 Set the Receive Data Delay

The McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY bit field determines the length of the data delay for the receive frame.

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.

McBSPi.MCBSPLP_RCR2_REG[1:0] RDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit–clocks (RDATDLY=0b00–0b10), as shown in Figure 18-62 below. In this figure, the data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1–cycle active frame-synchronization pulse.

Figure 18-62. Range of Programmable Data Delay



18.5.1.5.2.2.5.1 0-Bit Data Delay

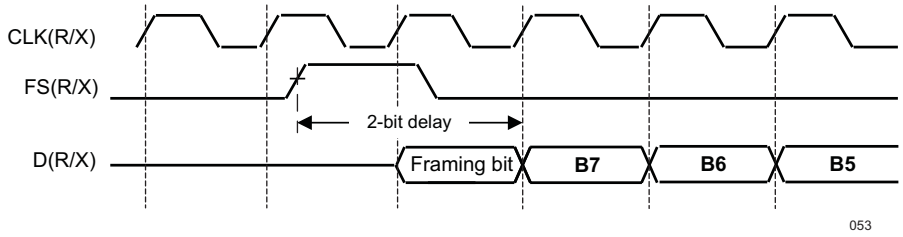
Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0–bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active–high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR, and thus on mcbsp_dx. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the mcbsp_dx pin.

18.5.1.5.2.2.5.2 2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (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 18-63. In this figure, the data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on.

Figure 18-63. 2-Bit Data Delay Used to Skip a Framing Bit



18.5.1.5.2.2.6 Set the Receive Sign-Extension and Justification Mode

The McBSPi.MCBSPLP_SPCR1_REG[14:13] RJUST bit field determines whether data received by the McBSP module is sign–extended or not and how it is justified.

RJUST bit selects whether data in RB is right- or left-justified (with respect to the MSB) in McBSPi.MCBSPLP_DRR_REG register and whether unused bits in McBSPi.MCBSPLP_DRR_REG are filled with zeroes or with sign bits.

Table 18-27 and Table 18-28 show the effects of various RJUST values; the effect on an example 12–bit receive–data value 0xABC, and the effect on an example 20–bit receive–data value 0xABCDE, respectively.

Table 18-27. Example: Use of RJUST Bit Field With 12-bit Data Value 0xABC

RJUST	Justification	Extension	Value in DRR_REG
0b00	Right	Zero fill MSBs	0x0000 0ABC
0b01	Right	Sign extend data into MSBs	0xFFFF FABC
0b10	Left	Zero fill LSBs	0xABC0 0000
0b11	Reserved	Reserved	Reserved

Table 18-28. Example: Use of RJUST Bit Field With 20-bit Data Value 0xABCDE

RJUST	Justification	Extension	Value in DRR_REG
0b00	Right	Zero fill MSBs	0x000A BCDE
0b01	Right	Sign extend data into MSBs	0xFFFA BCDE
0b10	Left	Zero fill LSBs	0xABCD E000
0b11	Reserved	Reserved	Reserved

18.5.1.5.2.2.7 Set the Receive Interrupt Mode

Please refer to Section 18.5.1.4.2.1.

18.5.1.5.2.2.8 Set the Receive DMA Mode

The McBSP receive-data DMA requests (McBSPi_DMA_RX) are active after the McBSPi.MCBSPLP_SPCR1_REG [0] RRST bit is released. After reset, the default DMA threshold (and length) is 1.

The receive DMA requests can be disabled by setting the McBSPi.MCBSPLP_RCCR_REG[3] RDMAEN bit to 0. When disabling the DMA, the DMA request line is deasserted even if a DMA transfer is pending, and the DMA state-machine is not reset.

To configure the McBSP receive data DMA requests, perform the following:

- Write the receive McBSPi.MCBSPLP_THRSH1_REG register with the required receive DMA request length (the length of the transfer is the same as the threshold value + 1).
- As long as the occupied locations level in the RB is above or equal to the THRSH1_REG value + 1, the DMA request is asserted.
- After transferring the configured (THRSH1_REG value + 1) number of words, the receive DMA request is deasserted, and then reasserted as soon as the conditions are met again.

18.5.1.5.2.3 Frame-Sync Behavior

18.5.1.5.2.3.1 Set the Receive Frame-Sync Mode

McBSPi.MCBSPLP_PCR_REG[10] FSRM bit, McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit, McBSPi.MCBSPLP_SPCR1_REG[15] ALB bit and McBSPi.MCBSPLP_XCCR_REG[5] DLB bit field are used to determine the source for receive frame synchronization and the function of the mcbsp_fsr pin.

Table 18-29 below shows how you can select various sources to provide the receive frame-synchronization signal and the effect on the mcbsp_fsr pin. The polarity of the signal on the mcbsp_fsr pin is determined by the McBSPi.MCBSPLP_PCR_REG[2] FSRP bit.

Table 18-29. FSRM and GSYNC Effects on Frame-Sync Signal and mcbsp_fsr Pin

FSRM	GSYNC	Source of Receive Frame Synchronization	MCBSPLP.FSR Pin Status
0	0 or 1	An external frame synchronization signal enters the McBSP module through the mcbsp_fsr pin. The signal is then inverted as determined by FSRP bit before being used as internal FSR.	Input
1	0	Internal FSR is driven by the SRG frame-synchronization signal (FSG).	Output. FSG is inverted as determined by FSRP bit before being driven out on the mcbsp_fsr pin.
1	1	Internal FSR is driven by the SRG frame-synchronization signal (FSG).	Input. The external frame-synchronization input on the mcbsp_fsr pin is used to synchronize CLKG and generate FSG pulses.

In digital loop-back mode (DLB=1), the transmit frame-synchronization signal is used as the receive frame-synchronization signal.

Also in the analog loop back mode (ALB=1), the internal receive clock signal (CLKR), and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

For more details on clock and frame-sync configuration, see Section 18.4.3.

18.5.1.5.2.3.2 Set the Receive Frame-Sync Polarity

The McBSPi.MCBSPLP_PCR_REG[2] FSRP bit determines whether frame-synchronization pulses are active high or active low on the mcbsp_fsr pin.

Receive frame-synchronization pulses can be generated internally by the SRG or driven by an external source. The source of frame synchronization is selected by programming the McBSPi.MCBSPLP_PCR_REG[10] FSRM bit. FSR is also affected by the McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit. For information about the effects of FSRM and GSYNC, see Section 18.5.1.5.2.3.1, *Set the Receive Frame-Sync Mode*.

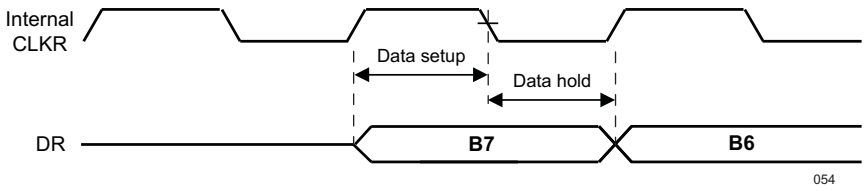
When FSR and FSX are inputs (FSXM = FSRM = 0, external frame-synchronization pulses), the McBSP module detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the mcbspi_dr pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from an external source via CLK(R/X) pins or driven by the SRG clock (CLKG) internal to the McBSP module.

When FSR and FSX are outputs, implying that they are driven by the SRG, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X). Similarly, data on the mcbsp_dx pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP bit fields in the pin control register (McBSPi.MCBSPLP_PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-synchronization signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-synchronization signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.

Figure 18-64 shows how data clocked by an external serial device using a rising edge can be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 18-64. Data Externally Clocked on a Rising Edge and Sampled on a Falling Edge



18.5.1.5.2.3.3 Set the SRG Frame-Sync Period and Pulse Width

The SRG can produce a clock signal, CLKG, and FSG. If the SRG is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

McBSPi.MCBSPLP_SRGR2_REG[11:0] FPER bit field is used to set the SRG frame-sync period and McBSPi.MCBSPLP_SRGR1_REG[15:8] FWID bit field is used to set the SRG pulse width.

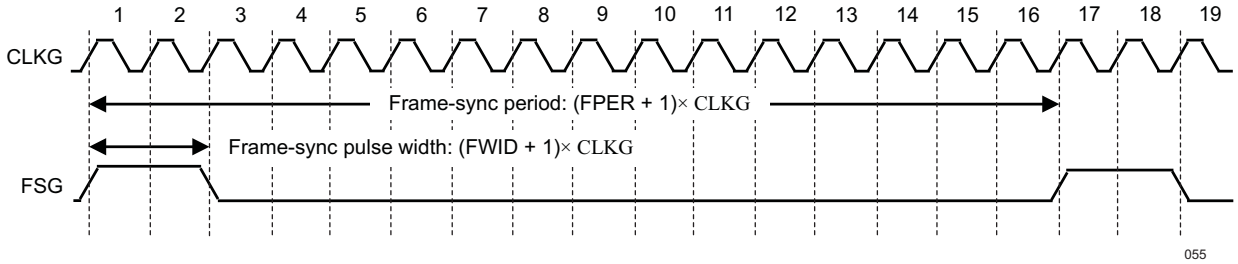
On FSG, the period from the start of a frame-synchronization pulse to the start of the next pulse is (FPER+1) CLKG cycles. The 12 bits of FPER allow a frame-synchronization period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC=1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID+1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down–counters. The 12–bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8–bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 18-65 shows a frame-synchronization period of 16 CLKG periods (FPER=15 or 00001111b) and a frame-synchronization pulse with an active width of 2 CLKG periods (FWID=1).

Figure 18-65. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the SRG comes out of reset, FSG is in its inactive state. Then, when GRST=1 and FSGM=1, a frame-synchronization pulse is generated. The frame width value (FWID+1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER+1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

18.5.1.5.2.4 Clock Behavior

18.5.1.5.2.4.1 Set the receive clock mode

McBSPi.MCBSPLP_PCR_REG[8] CLKRM bit, McBSPi.MCBSPLP_SPCR1_REG[15] ALB bit and McBSPi.MCBSPLP_XCCR_REG[5] DLB bit are used to set the receive clock mode.

Table 18-30 shows how to select various sources to provide the receive clock signal and affect the mcbsp_clkr pin. The McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit determines the polarity of the signal on the mcbsp_clkr pin.

Table 18-30. CLKRM Effect on Receive Clock Signal and mcbsp_clkr Pin

CLKRM	Source of Receive Clock	mcbsp_clkr Pin Status
0	The mcbsp_clkr pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP bit before being used.	Input
1	The SRG clock (CLKG) drives internal CLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the mcbsp_clkr pin.

In the digital loop-back mode (DLB=1) or analog loop-back mode (ALB = 1), the transmit clock signal is used as the receive clock signal. For more details on clock configuration, see Section 18.4.3.1.

18.5.1.5.2.4.2 Set the Receive Clock Polarity

McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit is used to set the receive clock polarity.

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 internal CLKR. 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 triggered 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 triggered clock before being sent out on the mcbsp_clkr pin.

NOTE: CLKRP=CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge.

18.5.1.5.2.4.3 Set the SRG Clock Divide-Down Value

McBSPi.MCBSPLP_SRGR1_REG[7:0] CLKGDV bit field is used to set the SRG clock divide-down value.

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV+1) of SRG input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. The CLKG duty cycle is 50%.

18.5.1.5.2.4.4 Set the SRG Clock Synchronization Mode

McBSPi.MCBSPLP_SRGR2_REG[15] GSYNC bit is used to set the SRG clock synchronization mode.

For more information about the clock synchronization feature, see Section 18.4.3.

18.5.1.5.2.4.5 Set the SRG Clock Mode (Choose an Input Clock)

McBSPi.MCBSPLP_PCR_REG[7] SCLKME bit and McBSPi.MCBSPLP_SRGR2_REG[13] CLKSM bit are used to set the SRG clock mode.

The SRG can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock.

For further details about the clock synchronization feature, see Section 18.4.3.

18.5.1.5.2.4.6 Set the SRG Input Clock Polarity

McBSPi.MCBSPLP_SRGR2_REG[14] CLKSP bit, McBSPi.MCBSPLP_PCR_REG[1] CLKXP bit and McBSPi.MCBSPLP_PCR_REG[0] CLKRP bit are used to set the SRG input clock polarity.

The SRG can produce a clock signal (CLKG) and a frame-synchronization signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the SRG must be driven by an input clock signal derived from the McBSP_FCLK clock or from an external clock on the mcbsp_clks, mcbsp_clkx, or mcbsp_clkr pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKSP for the mcbsp_clks pin, CLKXP for the mcbsp_clkx pin, CLKRP for the mcbsp_clkr pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

18.5.1.6 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1. Place the McBSP transmitter in reset
2. Program the McBSP registers for the desired transmitter operation
3. Take the transmitter out of reset

These 3 steps are described in more details in the sub-sections below.

18.5.1.6.1 Resetting (Step 1) and Enabling (Step 3) the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take it out of reset).

The serial port can be reset in the following two ways:

1. A global reset places the receiver, transmitter, and SRG in reset. When the device reset is removed, McBSPi.MCBSPLP_SPCR2_REG[6] GRST, McBSPi.MCBSPLP_SPCR2_REG[7] FRST, McBSPi.MCBSPLP_SPCR1_REG[0] RRST and McBSPi.MCBSPLP_SPCR2_REG[0] XRST bits = 0, which keeps the entire serial port in the reset state.
2. The serial port receiver can be reset directly using the McBSPi.MCBSPLP_SPCR2_REG[0] XRST bit. If the SRG needs to be used, SRG must be reset directly using the McBSPi.MCBSPLP_SPCR2_REG[6] GRST bit. Similar operation with the Frame Synchronization Generator also requires using the McBSPi.MCBSPLP_SPCR2_REG[7] FRST bit when the frame-sync signal must be generated.

To enable the transmitter, the preceding bits, cleared to 0, must be set to 1.

18.5.1.6.2 Programming the McBSP Registers for the Desired Transmitter Operation (Step 2)

Figure 18-66 and Figure 18-67 list important tasks to be performed when configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields.

Figure 18-66. Important Tasks to Configure the McBSP Transmitter (Part 1)

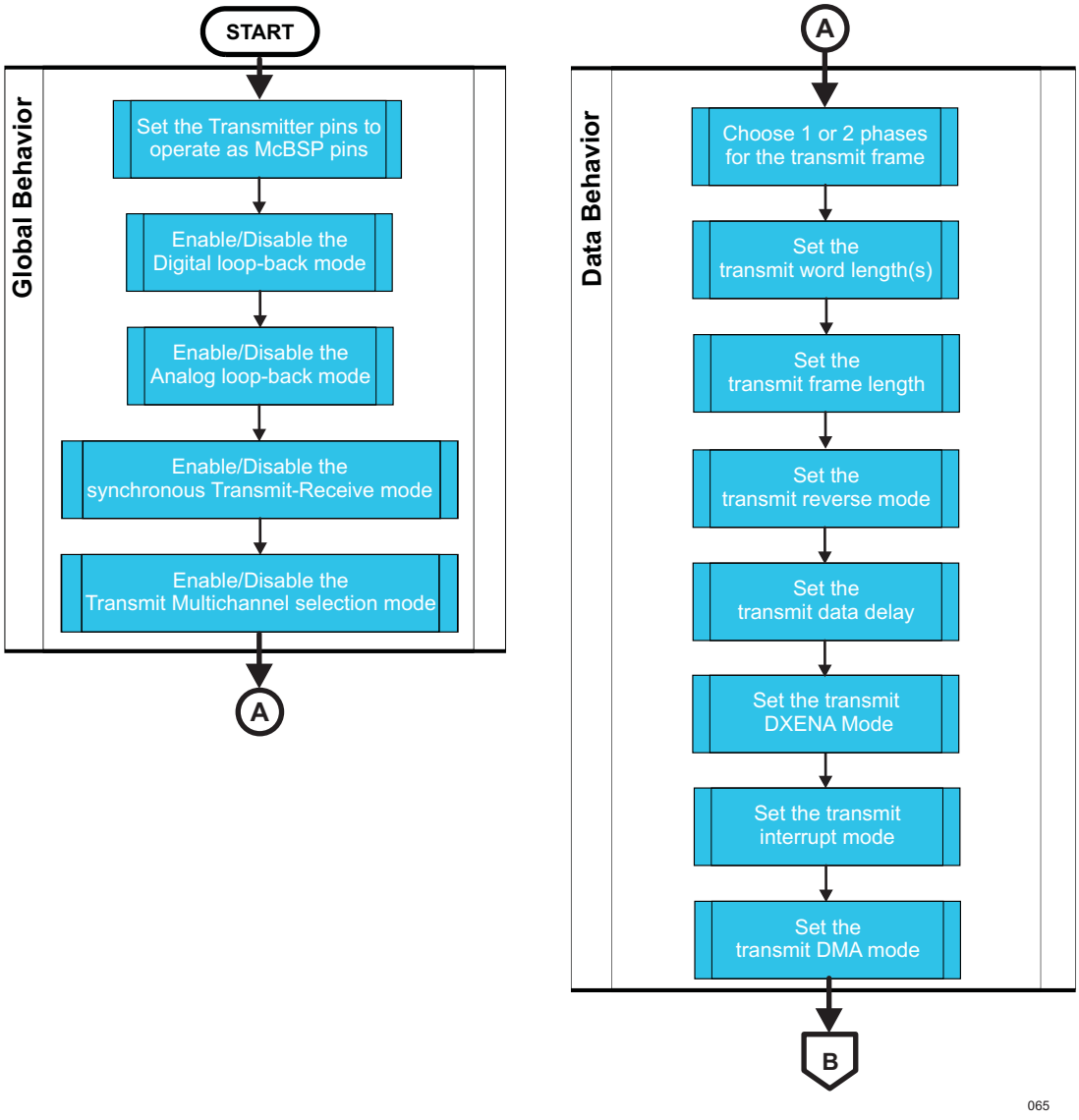
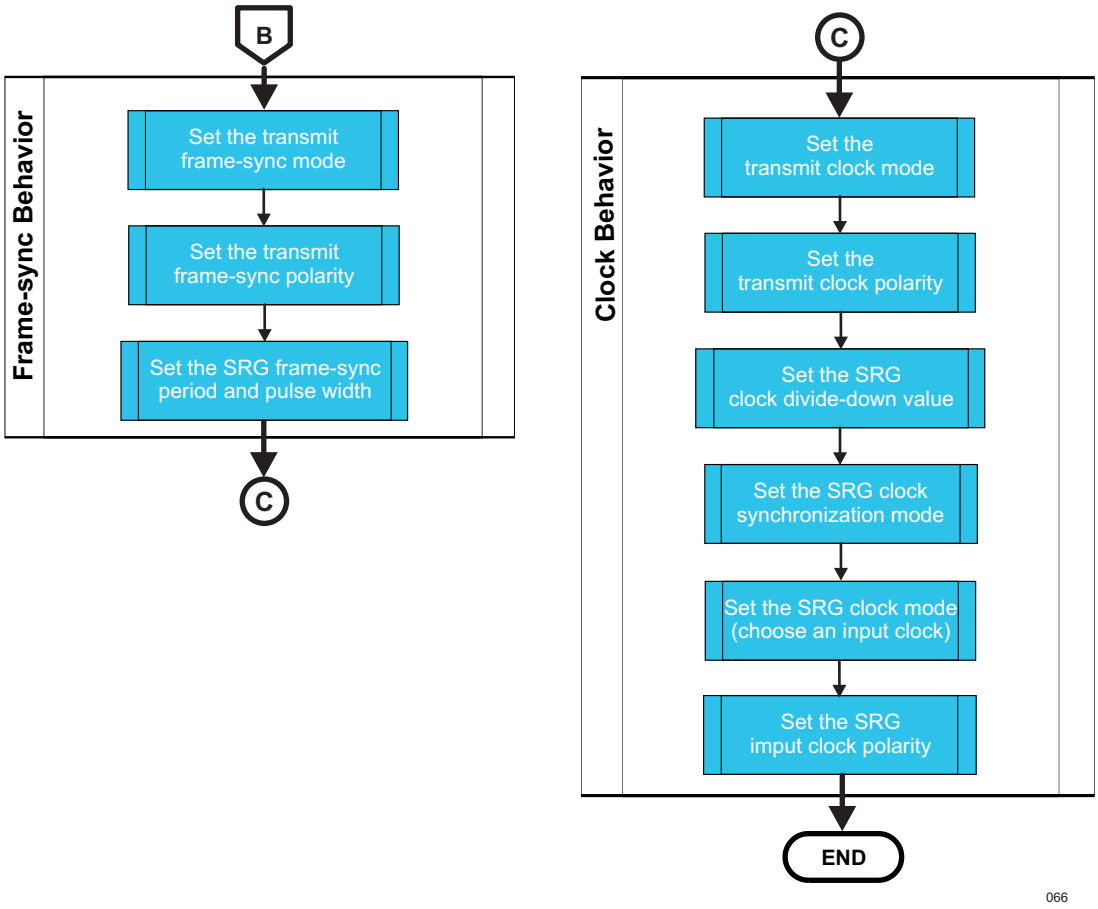


Figure 18-67. Important Tasks to Configure the McBSP Transmitter (Part 2)



18.5.1.6.2.1 Global Behavior

18.5.1.6.2.1.1 Set the Transmitter Pins to Operate as McBSP Pins

McBSPi.MCBSPLP_PCR_REG[13] XIOEN bit determines whether the transmitter pins are McBSP pins (XIOEN=0) or general–purpose I/O pins (XIOEN=1).

Refer to Section 18.5.1.7 which describes how to use McBSP pins as GPIO pins.

18.5.1.6.2.1.2 Enable/Disable the Digital Loop-Back Mode

Refer to it, Section 18.5.1.5.2.1.2.

18.5.1.6.2.1.3 Enable/Disable the Analog Loop-Back Mode

Refer to it, Section 18.5.1.5.2.1.3.

18.5.1.6.2.1.4 Enable/Disable the Synchronous Transmit-Receive Mode (McBSP1 only)

Refer to it, Section 18.5.1.5.2.1.4.

18.5.1.6.2.1.5 Enable/Disable the Transmit Multichannel Selection

McBSPi.MCBSPLP_MCR2_REG[1:0] XMCM bit field determines whether the transmit multichannel selection mode is on or off.

Refer to Section 18.4.6.7.

18.5.1.6.2.2 Data Behavior

18.5.1.6.2.2.1 Choose 1 or 2 Phases for the Transmit Frame

McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit determines whether the transmit data frame has one (Single phase) or two phases (Dual phase). When dual-phase is selected the number of words per phase must be set to one.

18.5.1.6.2.2.2 Set the Transmit Word Length(s)

McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 and McBSPi.MCBSPLP_XCR2_REG[7:5] XWDLEN2 bit fields determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the transmit data frame.

If a single–phase frame is selected, XWDLEN1 selects the length for every serial word received in the frame.

If a dual–phase frame is selected, XWDLEN1 and XWDLEN2 must be set to select the both length. These both bits can have values different.

18.5.1.6.2.2.3 Set the Transmit Frame Length

The transmit frame length is the number of serial words in the transmit frame.

McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 and McBSPi.MCBSPLP_XCR2_REG[14:8] XFRLEN2 bit fields determine how many serial words are in phase 1 and in phase 2, respectively, of the transmit data frame.

If a dual–phase frame is selected (XPHASE=1), the frame length is 2 words, the length of phase 1 (one word) plus the length of phase 2 (one word). Others values must not be used.

The 7–bit XFRLEN fields allow up to 128 words per phase. See Table 18-31 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Program the XFRLEN fields with [W minus 1], where W represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

Table 18-31. How to Calculate the Length of the Transmit Frame

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Don't care	(XFRLEN1value +1) words
1	XFRLEN1= 0	XFRLEN2 = 0	2 words

18.5.1.6.2.2.4 Set the Transmit Reverse Mode

McBSPi.MCBSPLP_XCR2_REG[4:3] XREVERSE bit field determines whether reverse (LSB first) data transfer option is chosen for McBSP reception.

For additional information about reverse mode, see Section 18.4.2.2.

18.5.1.6.2.2.5 Set the Transmit Data Delay

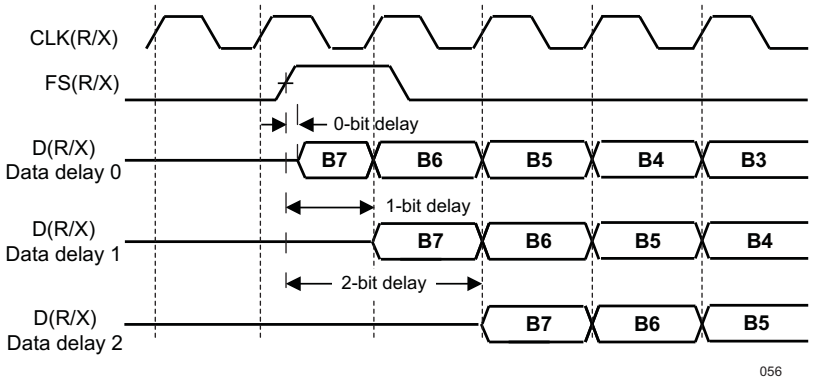
McBSPi.MCBSPLP_XCR2_REG[1:0] XDATDLY bit field determines the length of the data delay for the transmit frame.

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

XDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit–clocks (XDATDLY=00b–10b), as shown in Figure 18-68. In this figure, the data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on.

NOTE: Typically a 1–bit delay is selected, because data often follows a 1–cycle active frame-synchronization pulse.

Figure 18-68. Range of Programmable Data Delay



18.5.1.6.2.2.5.1 0-Bit Data Delay:

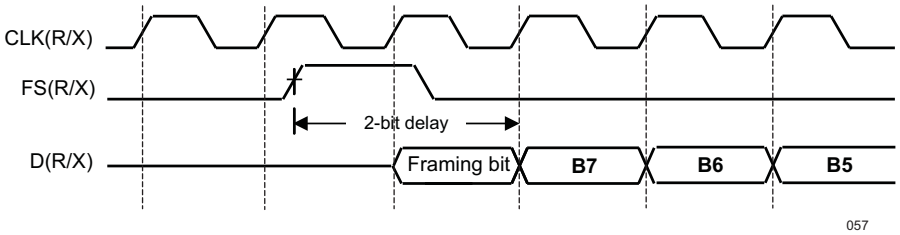
Normally, a frame-synchronization pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0–bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception, this problem is solved because receive data is sampled on the first falling edge of CLKR where an active–high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on mcbspi_dx. The transmitter then asynchronously detects the frame-synchronization signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the mcbsp_dx pin.

18.5.1.6.2.2.5.2 2-Bit Data Delay:

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (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 18-69. The data transferred is an 8–bit value with bits labeled B7, B6, B5, and so on.

Figure 18-69. 2-Bit Data Delay Used to Skip a Framing Bit



18.5.1.6.2.2.6 Set the Transmit DXENA Mode

McBSPi.MCBSPLP_SPCR1_REG[7] DXENA bit is used to set the transmit DXENA (DX delay enable) mode.

The DXENA bit controls the delay enabler on the mcbsp_dx pin. Set DXENA to enable an extra delay for turn–on time. This bit does not control the data itself, so only the first bit is delayed (the delay is given by a combinatorial delay buffer). The inserted delay: 13 ns, 18 ns (default), 24 ns or 30 ns can be set using the McBSPi.MCBSPLP_XCCR_REG[13:12] DXENDLY field. If you tie together the mcbsp_dx pins of multiple McBSP modules, make sure DXENA=1, to avoid having more than one McBSP transmitting on the data line at one time.

18.5.1.6.2.2.7 Set the Transmit Interrupt Mode

Refer to Section 18.5.1.4.2.2.

18.5.1.6.2.2.8 Set the Transmit DMA Mode

The McBSP transmit data DMA requests (McBSPi_DMA_TX) are active after the transmit McBSPi.MCBSPLP_SPCR2_REG[0] XRST bit is released. After reset, the default DMA threshold (and length) is 1.

The transmit DMA requests can be disabled by setting the McBSPi.MCBSPLP_XCCR_REG[3] XDMAEN bit to 0. When disabling the DMA, the DMA request line is deasserted even if a DMA transfer is pending, and the DMA state-machine is not reset.

To configure the McBSP transmit data DMA requests, follow this procedure:

- Write the transmit McBSPi.MCBSPLP_THRSH2_REG register with the required transmit DMA request length (the length of the transfer is the same as the threshold value + 1).
- As long as the free locations level in XB is above or equal to the THRSH2_REG value + 1, the DMA request is asserted.
- After transferring the configured (THRSH2_REG value + 1) number of words, the transmit DMA request is deasserted, and then reasserted as soon as the conditions are met again.

18.5.1.6.2.3 Frame-Synchronization Behavior

18.5.1.6.2.3.1 Set the Transmit Frame-Sync Mode

McBSPi.MCBSPLP_PCR_REG[11] FSXM bit and McBSPi.MCBSPLP_SRGR2_REG[12] FSGM bit are used to set the transmit frame-sync mode.

Table 18-32 shows how FSXM and FSGM select the source of transmit frame-synchronization pulses. The three choices are:

1. External frame-synchronization input
2. Sample rate generator frame-synchronization signal (FSG)
3. Sample rate generator frame-synchronization signal (FSG) gated by the transmit buffer XB empty condition.

Table 18-32 also shows the effect of each bit setting on the mcbsp_fsx pin. The FSXP bit determines the polarity of the signal on the mcbsp_fsx pin.

Table 18-32. How FSXM and FSGM Select the Source of Transmit Frame-Sync Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	mcbsp_fsx Pin Status
0	0 or 1	An external FSG enters the McBSP through the mcbsp_fsx pin. The signal is then inverted by FSXP bit before being used as internal FSX.	Input
1	1	Internal FSX is driven by the SRG FSG.	Output. FSG is inverted by FSXP bit before being driven out on mcbsp_fsx pin.
1	0	A XB empty condition causes the McBSP not to generate a transmit frame-synchronization pulse. The frame synchronization is generated taking into account the FWID and FPER bits as long as the transmit buffer is not empty. When the buffer is empty the generated frame synchronization signal is gated.	Output. The generated frame-synchronization pulse is inverted as determined by FSXP bit before being driven out on mcbsp_fsx pin.

If the SRG creates a frame-synchronization signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the mcbsp_fsr pin. For more details, see Section 18.4.3.3.

In the digital loopback mode (DLB=1) or analog loopback mode (ALB = 1), the transmit frame synchronization signal is used as the receive frame synchronization signal. For more details on frame-sync configuration, see Section 18.4.3.2.

18.5.1.6.2.3.2 Set the Transmit Frame-Sync Polarity

McBSPi.MCBSPLP_PCR_REG[3] FSXP bit determines whether frame-synchronization pulses are active high or active low on the mcbsp_fsx pin.

Transmit frame-synchronization pulses can be generated internally by the SRG or driven by an external source. The source of frame synchronization is selected by programming the McBSPi.MCBSPLP_PCR_REG[11] FSXM bit. FSX is also affected by the McBSPi.MCBSPLP_SRGR2_REG[12] FSGM bit. For information about the effects of FSXM and FSGM, see Section 18.5.1.6.2.3.1).

When FSR and FSX are inputs (FSXM=FSRM=0, external frame-synchronization pulses), the McBSP module detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the mcbsp_dr pin is also sampled on the falling edge of internal CLKR. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the SRG clock (CLKG) internal to the McBSP module.

When FSR and FSX are outputs, implying that they are driven by the SRG, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the mcbsp_dx pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR_REG) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All FSG (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP) and FSRP=FSXP=1, the external active-low frame-synchronization signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC=0) is selected and the polarity bit FS(R/X)P=1, the internal active–high FSG are inverted before being sent to the FS(R/X) pin.

18.5.1.6.2.3.3 Set the SRG Frame-Sync Period and Pulse Width

See Section 18.5.1.5.2.3.3.

18.5.1.6.2.4 Clock Behavior

18.5.1.6.2.4.1 Set the Transmit Clock Mode

The McBSPi.MCBSPLP_PCR_REG[9] CLKXM bit is used to set the transmit clock mode.

Table 18-33 shows how the CLKXM bit selects the transmit clock and the corresponding status of the mcbsp_clkx pin. The CLKXP bit determines the polarity of the signal on the mcbsp_clkx pin.

Table 18-33. CLKXM Bit Effect on Transmit Clock and MCBSPLP.CLKX Pin

CLKXM	Source of Transmit Clock	mcbsp_clkx Pin Status
0	Internal CLKX is driven by an external clock on the mcbsp_clkx pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Internal CLKX is driven by the SRG clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on mcbsp_clkx.

If the SRG creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the mcbsp_fsr pin.

In the digital loopback mode (DLB=1) or analog loopback mode (ALB = 1), the transmit frame synchronization signal is used as the receive frame synchronization signal. For more details on clock configuration, see Section 18.4.3.1.

18.5.1.6.2.4.2 Set the Transmit Clock Polarity

The McBSPi.MCBSPLP_PCR_REG[1] CLKXP bit is used to set the transmit clock polarity.

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 internal CLKX. 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, internal CLKX, is inverted before being sent out on the mcbsp_clkx pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). 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 internal CLKR. 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 triggered 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 triggered clock before being sent out on the mcbsp_clkr pin.

NOTE: CLKRP=CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge.

18.5.1.6.2.4.3 Set the SRG Clock Divide-Down Value

See Section 18.5.1.5.2.4.3.

18.5.1.6.2.4.4 Set the SRG Clock Synchronization Mode

See Section 18.5.1.5.2.4.4.

18.5.1.6.2.4.5 Set the SRG Clock Mode (Choose an Input Clock)

See Section 18.5.1.5.2.4.5.

18.5.1.6.2.4.6 Set the SRG Input Clock Polarity

See Section 18.5.1.5.2.4.6.

18.5.1.7 General-Purpose I/O on the McBSP Pins (Legacy Only)

Table 18-34 summarizes how to use the McBSP pins as general–purpose I/O pins. All the bits mentioned in the table except XRST and RRST bits are in the pin control register (McBSPi.MCBSPLP_PCR_REG). McBSPi.MCBSPLP_SPCR2_REG[0] XRST bit and McBSPi.MCBSPLP_SPCR1_REG[0] RRST bit are in the serial port control registers.

To use receiver pins mcbsp_clkr, mcbsp_fsr, and mcbsp_dr as general–purpose I/O pins rather than as serial port pins, you must set two conditions:

1. The receiver of the serial port is in reset (McBSPi.MCBSPLP_SPCR1_REG[0] RRST bit =0).
2. General–purpose I/O is enabled for the serial port receiver (McBSPi.MCBSPLP_PCR_REG[12] RIOEN=1).

The mcbsp_clkr and mcbsp_fsr pins can be individually configured as either input or output pins with the CLKRM and FSRM bits, respectively. The mcbsp_dr pin can only be an input pin. Table 18-34 shows, which bits in McBSPi.MCBSPLP_PCR_REG are used to read from/write to these pins.

For the transmitter pins mcbsp_clkx, mcbsp_fsx, and mcbsp_dx, you must meet two conditions:

1. The transmitter of the serial port is in reset (McBSPi.MCBSPLP_SPCR2_REG[0] XRST=0).
2. General–purpose I/O is enabled for the serial port transmitter (McBSPi.MCBSPLP_PCR_REG[12] XIOEN=1).

The mcbsp_clkx and mcbsp_fsx pins can be individually configured as input or output pins with the CLKXM and FSXM bits, respectively. The mcbsp_dx pin can only be an output pin. Table 18-34 shows the bits in McBSPi.MCBSPLP_PCR_REG used to read from/write to these pins.

For the mcbsp_clks pin (common to all McBSP modules), all of the reset and I/O enable conditions must be met:

1. Both the receiver and transmitter of the serial port are in reset (RRST=0 and XRST=0).
2. General–purpose I/O is enabled for both the receiver and the transmitter (RIOEN=1 and XIOEN=1).

The mcbsp_clks pin can only be an input pin. To read the status of the signal on the mcbsp_clks pin, read the McBSPi.MCBSPLP_PCR_REG[6] CLKS_STAT bit.

Table 18-34. Using McBSP Pins for General-Purpose I/O

Pin	General–Purpose use enabled by this bit combination	Selected as output when	Output value driven from this bit	Selected as input when	Input value read from this bit
mcbsp_clkx	XRST = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
mcbsp_fsx	XRST = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
mcbsp_dx	XRST = 0 XIOEN = 1	Always	DX_STAT	Never	Does not apply
mcbsp_clkr	RRST = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
mcbsp_fsr	RRST = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
mcbsp_dr	RRST = 0 RIOEN = 1	Never	Does not apply	Always	DR_STAT
mcbsp_clks	RRST = XRST = 0 RIOEN = XIOEN = 1	Never	Does not apply	Always	CLKS_STAT

18.5.1.8 Data Packing Examples

This section describes two ways to implement data packing in the McBSP: Using frame length and word length, and using word length and ignoring frame sync pulses.

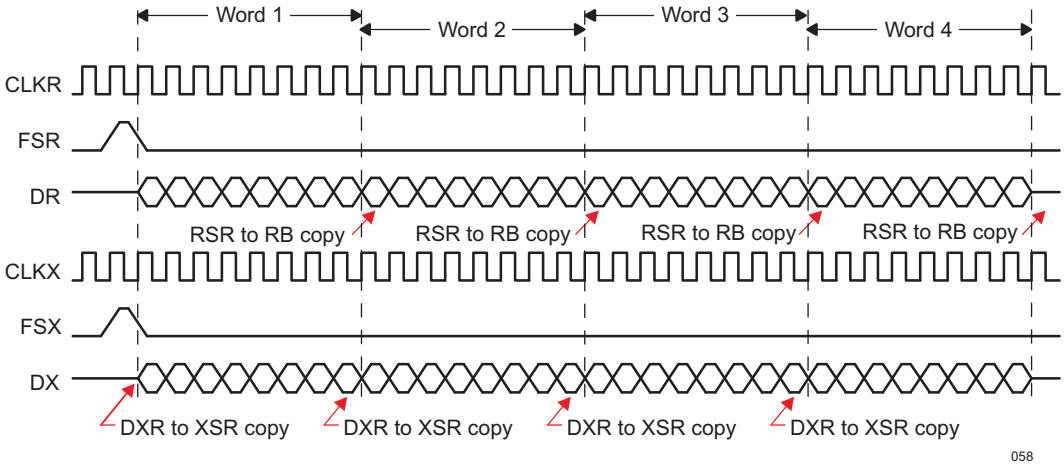
18.5.1.8.1 Data Packing Using Frame Length and Word Length

Frame length and word length can be manipulated to pack data effectively. For example, four 8-bit words are transferred in a single-phase frame, as shown in Figure 18-70. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 00011b: 4-word frame
- (R/X)WDLEN1 = 101b: 32-bit words

Four 8-bit data words are transferred to and from the McBSP by the MPU/IVA2.2 subsystems or the sDMA controller. Thus, four reads from McBSPi.MCBSPLP_DRR_REG and four writes to McBSPi.MCBSPLP_DXR_REG are necessary for each frame.

Figure 18-70. Four 8-bit Data Words Transferred To/From McBSP Module

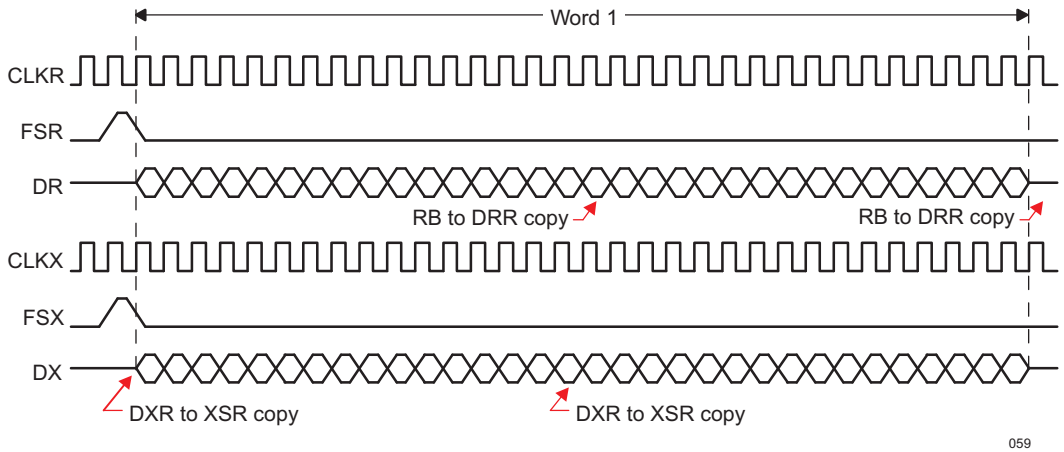


This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 18-71. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Two 16-bit data words are transferred to and from the McBSP by the MPU/IVA2.2 subsystems or the sDMA controller. Thus, two reads from McBSPi.MCBSPLP_DRR_REG and two writes to McBSPi.MCBSPLP_DXR_REG are necessary for each frame. This results in only half the number of transfers, as compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

Figure 18-71. One 32-bit Data Word Transferred To/From McBSP Module



18.5.1.8.2 Data Packing Using Word Length and the Frame-Sync Ignore Function

When there are multiple words per frame, data can be packed by increasing the word length (defining a serial word with more bits) and by ignoring frame-sync pulses. Figure 18-72 shows the McBSP operating at the maximum packet frequency. Here, each frame has only one 8-bit word. Notice the frame-sync pulse that initiates each frame transfer for reception and for transmission. For reception, this configuration requires one read operation for each word. For transmission, this configuration requires one write operation for each word.

Figure 18-72. 8-bit Data Words Transferred at Maximum Packet Frequency

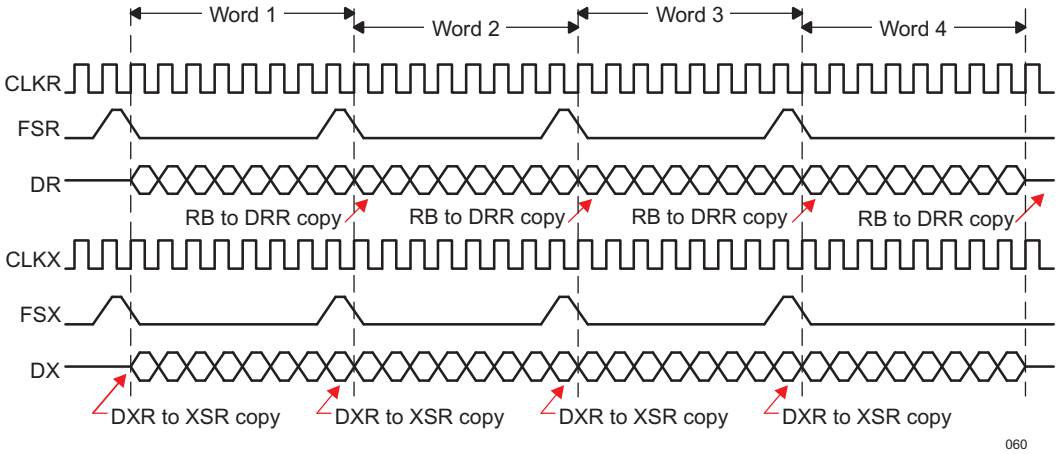
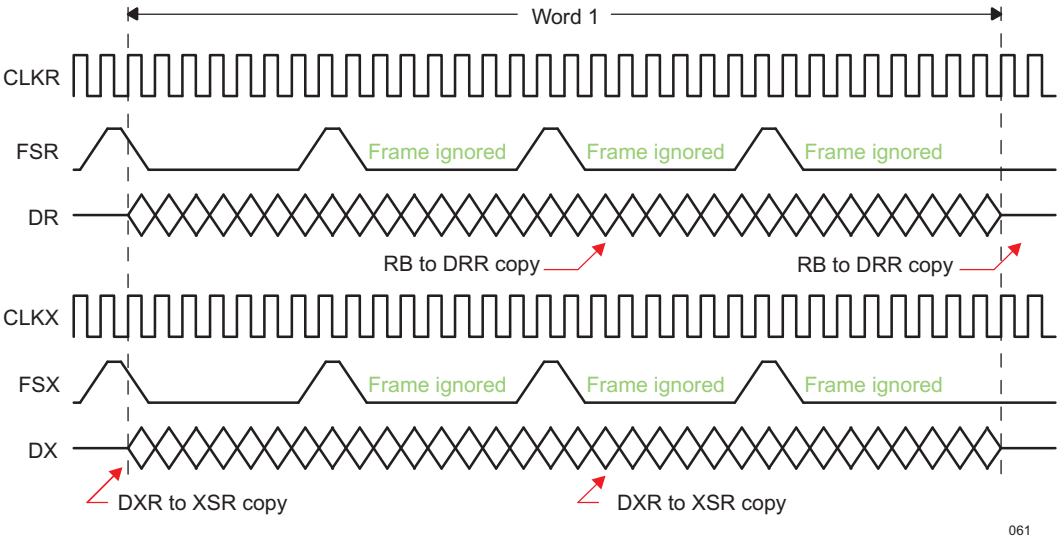


Figure 18-73 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-sync pulse. However, the McBSP ignores subsequent pulses. Only one read transfer or one write transfer is required every 32 bits. This configuration effectively reduces the required bus bandwidth to one-fourth the bandwidth needed to transfer four 8-bit words.

Figure 18-73. Configuring the Data Stream as a Continuous 32-bit Word



18.5.2 SIDETONE Feature

18.5.2.1 SIDETONE Activation Procedure

Before you enable a SIDETONE selection mode, make sure you properly configure the data frame for multichannel mode of the McBSP module:

- Select a single–phase frame (McBSPi.MCBSPLP_RCR2_REG[15] RPHASE bit and McBSPi.MCBSPLP_XCR2_REG[15] XPHASE bit=0). Each frame represents a TDM data stream.
- Set to 1 the McBSPi.MCBSPLP_MCR1_REG[0] RMCM bit, to select multichannel mode enable.
- Set a frame length (McBSPi.MCBSPLP_RCR1_REG[14:8] RFRLEN1 bit field and McBSPi.MCBSPLP_XCR1_REG[14:8] XFRLEN1 bit field) that includes the highest–numbered channel to be used (a maximum of 4 channels can be used in this configuration).
- Set a word length (McBSPi.MCBSPLP_RCR1_REG[7:5] RWDLEN1 bit field and McBSPi.MCBSPLP_XCR1_REG[7:5] XWDLEN1 bit field) to be either 16, 24 or 32 (see the note

below).

- Select the input/output channels configured as SIDETONE channels by setting the following bit fields listed in Table 18-35:

Table 18-35. Selection of the SIDETONE Input and Output Channels

Bit field	Description
McBSPi.MCBSPLP_SSELCR_REG[9:7] OCH1ASSIGN	Map the CH1 data for the speaker out channels to one of the McBSP channels (1 out of 8 channels)
McBSPi.MCBSPLP_SSELCR_REG[6:4] OCH0ASSIGN	Map the CH0 data for the speaker out channels to one of the McBSP channels (1 out of 8 channels)
McBSPi.MCBSPLP_SSELCR_REG[3:2] ICH1ASSIGN	Map the CH1 data from the digital microphone channels to one of the McBSP channels (1 out of 4 channels)
McBSPi.MCBSPLP_SSELCR_REG[1:0] ICH0ASSIGN	Map the CH0 data from the digital microphone channels to one of the McBSP channels (1 out of 4 channels)

- Enable SIDETONE by setting 2 bits to 1:
  1. In McBSP module, the McBSPi.MCBSPLP_SSELCR_REG[10] SIDETONEEN bit
  2. In SIDETONE core, the McBSPi.ST_SSELCR_REG[0] SIDETONEEN bit

NOTE: Word width in the loop is 24 bits. If input channel word width is less than 24 bits, LSBs of the samples are zero padded. If input channel word width is more than 24 bits, samples are truncated.

18.5.2.2 SIDETONE Initialization Procedure

The SIDETONE core initialization procedure is as follows:

1. Write 1 in McBSPi.ST_SSELCR_REG[1] COEFFWREN bit to enable loading of the FIR coefficients.
2. Load one by one the FIR coefficients performing 128 write accesses to McBSPi.ST_SFIRCR_REG[15:0] FIRCOEFF bit field, the McBSPi.ST_SFIRCR_REG[0] FIRCOEFF bit is loaded first. To ensure the completion of loading, check the status bit-field, McBSPi.ST_SSELCR_REG[2] COEFFWRDONE bit.
3. Set-up gain values for both channels writing desired values inside McBSPi.ST_SGAINCR_REG[31:16] CH1GAIN bit field for the second sidetone channel and McBSPi.ST_SGAINCR_REG[15:0] CH0GAIN bit field for the first sidetone channel.

18.5.2.3 SIDETONE FIR Coefficients Writing

Writing the coefficients is only possible when the SIDETONE is disabled.

1. Write 1 in McBSPi.ST_SSELCR_REG[1] COEFFWREN bit to enable writing of the FIR coefficients, or write 0, and then write 1 in the COEFFWREN bit to reset the write process.
2. Perform 128 write accesses in 32/16 LSB bit mode on the McBSPi.ST_SFIRCR_REG[15:0] FIRCOEFF bit field. The first write action following the previous step sets the coefficient index 0.
3. Check that the write is done by reading McBSPi.ST_SSELCR_REG[2] COEFFWRDONE bit (it becomes 1 after the 128th coefficient is written).

18.5.2.4 SIDETONE FIR Coefficients Reading

Reading the coefficients is only possible when the SIDETONE is disabled.

- Write 0 in McBSPi.ST_SSELCR_REG[1] COEFFWREN bit to enable reading of the FIR coefficients, or write 1, and then write 0 in the COEFFWREN bit to reset the read process.
- Perform 128 read accesses from McBSPi.ST_SFIRCR_REG[15:0] FIRCOEFF bit field. Each read returns coefficients one by one. The first read following previous step returns coefficient index 0.

18.6 McBSP Register Manual

Table 18-36 shows the base address and address space for the device module instances.

Table 18-36. Device Instance Summary

Module Name	Base Address (hex)	Size
McBSP1	0x4807 4000	4K bytes
McBSP5	0x4809 6000	4K bytes
McBSP2	0x4902 2000	4K bytes
McBSP3	0x4902 4000	4K bytes
McBSP4	0x4902 6000	4K bytes
SIDETONE_McBSP2	0x4902 8000	4K bytes
SIDETONE_McBSP3	0x4902 A000	4K bytes

18.6.1 McBSP Register Mapping Summary

CAUTION

The McBSP data registers (that is, DXR_REG for data transmit and DRR_REG for data receive) support 8/16/32-bit data accesses. All other McBSP registers are limited to 32-bit data accesses. 16-bit and 8-bit data accesses are not allowed and can corrupt register contents.

Table 18-37. McBSP1 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_DRR_REG	R	32	0x0000 0000	0x4807 4000
MCBSPLP_DXR_REG	W	32	0x0000 0008	0x4807 4008
MCBSPLP_SPCR2_REG	RW	32	0x0000 0010	0x4807 4010
MCBSPLP_SPCR1_REG	RW	32	0x0000 0014	0x4807 4014
MCBSPLP_RCR2_REG	RW	32	0x0000 0018	0x4807 4018
MCBSPLP_RCR1_REG	RW	32	0x0000 001C	0x4807 401C
MCBSPLP_XCR2_REG	RW	32	0x0000 0020	0x4807 4020
MCBSPLP_XCR1_REG	RW	32	0x0000 0024	0x4807 4024
MCBSPLP_SRGR2_REG	RW	32	0x0000 0028	0x4807 4028
MCBSPLP_SRGR1_REG	RW	32	0x0000 002C	0x4807 402C
MCBSPLP_MCR2_REG	RW	32	0x0000 0030	0x4807 4030
MCBSPLP_MCR1_REG	RW	32	0x0000 0034	0x4807 4034
MCBSPLP_RCERA_REG	RW	32	0x0000 0038	0x4807 4038
MCBSPLP_RCERB_REG	RW	32	0x0000 003C	0x4807 403C
MCBSPLP_XCERA_REG	RW	32	0x0000 0040	0x4807 4040
MCBSPLP_XCERB_REG	RW	32	0x0000 0044	0x4807 4044
MCBSPLP_PCR_REG	RW	32	0x0000 0048	0x4807 4048
MCBSPLP_RCERC_REG	RW	32	0x0000 004C	0x4807 404C
MCBSPLP_RCERD_REG	RW	32	0x0000 0050	0x4807 4050
MCBSPLP_XCERC_REG	RW	32	0x0000 0054	0x4807 4054
MCBSPLP_XCERD_REG	RW	32	0x0000 0058	0x4807 4058
MCBSPLP_RCERE_REG	RW	32	0x0000 005C	0x4807 405C
MCBSPLP_RCERF_REG	RW	32	0x0000 0060	0x4807 4060

Table 18-37. McBSP1 Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_XCERE_REG	RW	32	0x0000 0064	0x4807 4064
MCBSPLP_XCERF_REG	RW	32	0x0000 0068	0x4807 4068
MCBSPLP_RCERG_REG	RW	32	0x0000 006C	0x4807 406C
MCBSPLP_RCERH_REG	RW	32	0x0000 0070	0x4807 4070
MCBSPLP_XCERG_REG	RW	32	0x0000 0074	0x4807 4074
MCBSPLP_XCERH_REG	RW	32	0x0000 0078	0x4807 4078
MCBSPLP_REV_REG	R	32	0x0000 007C	0x4807 407C
MCBSPLP_RINTCLR_REG	RW	32	0x0000 0080	0x4807 4080
MCBSPLP_XINTCLR_REG	RW	32	0x0000 0084	0x4807 4084
MCBSPLP_ROVFLCLR_REG	RW	32	0x0000 0088	0x4807 4088
MCBSPLP_SYSCONFIG_REG	RW	32	0x0000 008C	0x4807 408C
MCBSPLP_THRSH2_REG	RW	32	0x0000 0090	0x4807 4090
MCBSPLP_THRSH1_REG	RW	32	0x0000 0094	0x4807 4094
MCBSPLP_IRQSTATUS_REG	RW	32	0x0000 00A0	0x4807 40A0
MCBSPLP_IRQENABLE_REG	RW	32	0x0000 00A4	0x4807 40A4
MCBSPLP_WAKEUPEN_REG	RW	32	0x0000 00A8	0x4807 40A8
MCBSPLP_XCCR_REG	RW	32	0x0000 00AC	0x4807 40AC
MCBSPLP_RCCR_REG	RW	32	0x0000 00B0	0x4807 40B0
MCBSPLP_XBUFFSTAT_REG	R	32	0x0000 00B4	0x4807 40B4
MCBSPLP_RBUFFSTAT_REG	R	32	0x0000 00B8	0x4807 40B8
MCBSPLP_SSELCR_REG	RW	32	0x0000 00BC	0x4807 40BC
MCBSPLP_STATUS_REG	R	32	0x0000 00C0	0x4807 40C0

Table 18-38. McBSP5 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_DRR_REG	R	32	0x0000 0000	0x4809 6000
MCBSPLP_DXR_REG	W	32	0x0000 0008	0x4809 6008
MCBSPLP_SPCR2_REG	RW	32	0x0000 0010	0x4809 6010
MCBSPLP_SPCR1_REG	RW	32	0x0000 0014	0x4809 6014
MCBSPLP_RCR2_REG	RW	32	0x0000 0018	0x4809 6018
MCBSPLP_RCR1_REG	RW	32	0x0000 001C	0x4809 601C
MCBSPLP_XCR2_REG	RW	32	0x0000 0020	0x4809 6020
MCBSPLP_XCR1_REG	RW	32	0x0000 0024	0x4809 6024
MCBSPLP_SRGR2_REG	RW	32	0x0000 0028	0x4809 6028
MCBSPLP_SRGR1_REG	RW	32	0x0000 002C	0x4809 602C
MCBSPLP_MCR2_REG	RW	32	0x0000 0030	0x4809 6030
MCBSPLP_MCR1_REG	RW	32	0x0000 0034	0x4809 6034
MCBSPLP_RCERA_REG	RW	32	0x0000 0038	0x4809 6038
MCBSPLP_RCERB_REG	RW	32	0x0000 003C	0x4809 603C
MCBSPLP_XCERA_REG	RW	32	0x0000 0040	0x4809 6040
MCBSPLP_XCERB_REG	RW	32	0x0000 0044	0x4809 6044
MCBSPLP_PCR_REG	RW	32	0x0000 0048	0x4809 6048
MCBSPLP_RCERC_REG	RW	32	0x0000 004C	0x4809 604C
MCBSPLP_RCERD_REG	RW	32	0x0000 0050	0x4809 6050
MCBSPLP_XCERC_REG	RW	32	0x0000 0054	0x4809 6054

Table 18-38. McBSP5 Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_XCERD_REG	RW	32	0x0000 0058	0x4809 6058
MCBSPLP_RCERE_REG	RW	32	0x0000 005C	0x4809 605C
MCBSPLP_RCERF_REG	RW	32	0x0000 0060	0x4809 6060
MCBSPLP_XCERE_REG	RW	32	0x0000 0064	0x4809 6064
MCBSPLP_XCERF_REG	RW	32	0x0000 0068	0x4809 6068
MCBSPLP_RCERG_REG	RW	32	0x0000 006C	0x4809 606C
MCBSPLP_RCERH_REG	RW	32	0x0000 0070	0x4809 6070
MCBSPLP_XCERG_REG	RW	32	0x0000 0074	0x4809 6074
MCBSPLP_XCERH_REG	RW	32	0x0000 0078	0x4809 6078
MCBSPLP_REV_REG	R	32	0x0000 007C	0x4809 607C
MCBSPLP_RINTCLR_REG	RW	32	0x0000 0080	0x4809 6080
MCBSPLP_XINTCLR_REG	RW	32	0x0000 0084	0x4809 6084
MCBSPLP_ROVFLCLR_REG	RW	32	0x0000 0088	0x4809 6088
MCBSPLP_SYSCONFIG_REG	RW	32	0x0000 008C	0x4809 608C
MCBSPLP_THRSH2_REG	RW	32	0x0000 0090	0x4809 6090
MCBSPLP_THRSH1_REG	RW	32	0x0000 0094	0x4809 6094
MCBSPLP_IRQSTATUS_REG	RW	32	0x0000 00A0	0x4809 60A0
MCBSPLP_IRQENABLE_REG	RW	32	0x0000 00A4	0x4809 60A4
MCBSPLP_WAKEUPEN_REG	RW	32	0x0000 00A8	0x4809 60A8
MCBSPLP_XCCR_REG	RW	32	0x0000 00AC	0x4809 60AC
MCBSPLP_RCCR_REG	RW	32	0x0000 00B0	0x4809 60B0
MCBSPLP_XBUFFSTAT_REG	R	32	0x0000 00B4	0x4809 60B4
MCBSPLP_RBUFFSTAT_REG	R	32	0x0000 00B8	0x4809 60B8
MCBSPLP_SSELCR_REG	RW	32	0x0000 00BC	0x4809 60BC
MCBSPLP_STATUS_REG	R	32	0x0000 00C0	0x4809 60C0

Table 18-39. McBSP2 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_DRR_REG	R	32	0x0000 0000	0x4902 2000
MCBSPLP_DXR_REG	W	32	0x0000 0008	0x4902 2008
MCBSPLP_SPCR2_REG	RW	32	0x0000 0010	0x4902 2010
MCBSPLP_SPCR1_REG	RW	32	0x0000 0014	0x4902 2014
MCBSPLP_RCR2_REG	RW	32	0x0000 0018	0x4902 2018
MCBSPLP_RCR1_REG	RW	32	0x0000 001C	0x4902 201C
MCBSPLP_XCR2_REG	RW	32	0x0000 0020	0x4902 2020
MCBSPLP_XCR1_REG	RW	32	0x0000 0024	0x4902 2024
MCBSPLP_SRGR2_REG	RW	32	0x0000 0028	0x4902 2028
MCBSPLP_SRGR1_REG	RW	32	0x0000 002C	0x4902 202C
MCBSPLP_MCR2_REG	RW	32	0x0000 0030	0x4902 2030
MCBSPLP_MCR1_REG	RW	32	0x0000 0034	0x4902 2034
MCBSPLP_RCERA_REG	RW	32	0x0000 0038	0x4902 2038
MCBSPLP_RCERB_REG	RW	32	0x0000 003C	0x4902 203C
MCBSPLP_XCERA_REG	RW	32	0x0000 0040	0x4902 2040
MCBSPLP_XCERB_REG	RW	32	0x0000 0044	0x4902 2044
MCBSPLP_PCR_REG	RW	32	0x0000 0048	0x4902 2048

Table 18-39. McBSP2 Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_RCERC_REG	RW	32	0x0000 004C	0x4902 204C
MCBSPLP_RCERD_REG	RW	32	0x0000 0050	0x4902 2050
MCBSPLP_XCERC_REG	RW	32	0x0000 0054	0x4902 2054
MCBSPLP_XCERD_REG	RW	32	0x0000 0058	0x4902 2058
MCBSPLP_RCERE_REG	RW	32	0x0000 005C	0x4902 205C
MCBSPLP_RCERF_REG	RW	32	0x0000 0060	0x4902 2060
MCBSPLP_XCERE_REG	RW	32	0x0000 0064	0x4902 2064
MCBSPLP_XCERF_REG	RW	32	0x0000 0068	0x4902 2068
MCBSPLP_RCERG_REG	RW	32	0x0000 006C	0x4902 206C
MCBSPLP_RCERH_REG	RW	32	0x0000 0070	0x4902 2070
MCBSPLP_XCERG_REG	RW	32	0x0000 0074	0x4902 2074
MCBSPLP_XCERH_REG	RW	32	0x0000 0078	0x4902 2078
MCBSPLP_REV_REG	R	32	0x0000 007C	0x4902 207C
MCBSPLP_RINTCLR_REG	RW	32	0x0000 0080	0x4902 2080
MCBSPLP_XINTCLR_REG	RW	32	0x0000 0084	0x4902 2084
MCBSPLP_ROVFLCLR_REG	RW	32	0x0000 0088	0x4902 2088
MCBSPLP_SYSCONFIG_REG	RW	32	0x0000 008C	0x4902 208C
MCBSPLP_THRSH2_REG	RW	32	0x0000 0090	0x4902 2090
MCBSPLP_THRSH1_REG	RW	32	0x0000 0094	0x4902 2094
MCBSPLP_IRQSTATUS_REG	RW	32	0x0000 00A0	0x4902 20A0
MCBSPLP_IRQENABLE_REG	RW	32	0x0000 00A4	0x4902 20A4
MCBSPLP_WAKEUPEN_REG	RW	32	0x0000 00A8	0x4902 20A8
MCBSPLP_XCCR_REG	RW	32	0x0000 00AC	0x4902 20AC
MCBSPLP_RCCR_REG	RW	32	0x0000 00B0	0x4902 20B0
MCBSPLP_XBUFFSTAT_REG	R	32	0x0000 00B4	0x4902 20B4
MCBSPLP_RBUFFSTAT_REG	R	32	0x0000 00B8	0x4902 20B8
MCBSPLP_SSELCR_REG	RW	32	0x0000 00BC	0x4902 20BC
MCBSPLP_STATUS_REG	R	32	0x0000 00C0	0x4902 20C0

Table 18-40. McBSP3 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_DRR_REG	R	32	0x0000 0000	0x4902 4000
MCBSPLP_DXR_REG	W	32	0x0000 0008	0x4902 4008
MCBSPLP_SPCR2_REG	RW	32	0x0000 0010	0x4902 4010
MCBSPLP_SPCR1_REG	RW	32	0x0000 0014	0x4902 4014
MCBSPLP_RCR2_REG	RW	32	0x0000 0018	0x4902 4018
MCBSPLP_RCR1_REG	RW	32	0x0000 001C	0x4902 401C
MCBSPLP_XCR2_REG	RW	32	0x0000 0020	0x4902 4020
MCBSPLP_XCR1_REG	RW	32	0x0000 0024	0x4902 4024
MCBSPLP_SRGR2_REG	RW	32	0x0000 0028	0x4902 4028
MCBSPLP_SRGR1_REG	RW	32	0x0000 002C	0x4902 402C
MCBSPLP_MCR2_REG	RW	32	0x0000 0030	0x4902 4030
MCBSPLP_MCR1_REG	RW	32	0x0000 0034	0x4902 4034
MCBSPLP_RCERA_REG	RW	32	0x0000 0038	0x4902 4038
MCBSPLP_RCERB_REG	RW	32	0x0000 003C	0x4902 403C

Table 18-40. McBSP3 Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_XCERA_REG	RW	32	0x0000 0040	0x4902 4040
MCBSPLP_XCERB_REG	RW	32	0x0000 0044	0x4902 4044
MCBSPLP_PCR_REG	RW	32	0x0000 0048	0x4902 4048
MCBSPLP_RCERC_REG	RW	32	0x0000 004C	0x4902 404C
MCBSPLP_RCERD_REG	RW	32	0x0000 0050	0x4902 4050
MCBSPLP_XCERC_REG	RW	32	0x0000 0054	0x4902 4054
MCBSPLP_XCERD_REG	RW	32	0x0000 0058	0x4902 4058
MCBSPLP_RCERE_REG	RW	32	0x0000 005C	0x4902 405C
MCBSPLP_RCERF_REG	RW	32	0x0000 0060	0x4902 4060
MCBSPLP_XCERE_REG	RW	32	0x0000 0064	0x4902 4064
MCBSPLP_XCERF_REG	RW	32	0x0000 0068	0x4902 4068
MCBSPLP_RCERG_REG	RW	32	0x0000 006C	0x4902 406C
MCBSPLP_RCERH_REG	RW	32	0x0000 0070	0x4902 4070
MCBSPLP_XCERG_REG	RW	32	0x0000 0074	0x4902 4074
MCBSPLP_XCERH_REG	RW	32	0x0000 0078	0x4902 4078
MCBSPLP_REV_REG	R	32	0x0000 007C	0x4902 407C
MCBSPLP_RINTCLR_REG	RW	32	0x0000 0080	0x4902 4080
MCBSPLP_XINTCLR_REG	RW	32	0x0000 0084	0x4902 4084
MCBSPLP_ROVFLCLR_REG	RW	32	0x0000 0088	0x4902 4088
MCBSPLP_SYSCONFIG_REG	RW	32	0x0000 008C	0x4902 408C
MCBSPLP_THRSH2_REG	RW	32	0x0000 0090	0x4902 4090
MCBSPLP_THRSH1_REG	RW	32	0x0000 0094	0x4902 4094
MCBSPLP_IRQSTATUS_REG	RW	32	0x0000 00A0	0x4902 40A0
MCBSPLP_IRQENABLE_REG	RW	32	0x0000 00A4	0x4902 40A4
MCBSPLP_WAKEUPEN_REG	RW	32	0x0000 00A8	0x4902 40A8
MCBSPLP_XCCR_REG	RW	32	0x0000 00AC	0x4902 40AC
MCBSPLP_RCCR_REG	RW	32	0x0000 00B0	0x4902 40B0
MCBSPLP_XBUFFSTAT_REG	R	32	0x0000 00B4	0x4902 40B4
MCBSPLP_RBUFFSTAT_REG	R	32	0x0000 00B8	0x4902 40B8
MCBSPLP_SSELCR_REG	RW	32	0x0000 00BC	0x4902 40BC
MCBSPLP_STATUS_REG	R	32	0x0000 00C0	0x4902 40C0

Table 18-41. McBSP4 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_DRR_REG	R	32	0x0000 0000	0x4902 6000
MCBSPLP_DXR_REG	W	32	0x0000 0008	0x4902 6008
MCBSPLP_SPCR2_REG	RW	32	0x0000 0010	0x4902 6010
MCBSPLP_SPCR1_REG	RW	32	0x0000 0014	0x4902 6014
MCBSPLP_RCR2_REG	RW	32	0x0000 0018	0x4902 6018
MCBSPLP_RCR1_REG	RW	32	0x0000 001C	0x4902 601C
MCBSPLP_XCR2_REG	RW	32	0x0000 0020	0x4902 6020
MCBSPLP_XCR1_REG	RW	32	0x0000 0024	0x4902 6024
MCBSPLP_SRGR2_REG	RW	32	0x0000 0028	0x4902 6028
MCBSPLP_SRGR1_REG	RW	32	0x0000 002C	0x4902 602C
MCBSPLP_MCR2_REG	RW	32	0x0000 0030	0x4902 6030

Table 18-41. McBSP4 Registers Mapping Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
MCBSPLP_MCR1_REG	RW	32	0x0000 0034	0x4902 6034
MCBSPLP_RCERA_REG	RW	32	0x0000 0038	0x4902 6038
MCBSPLP_RCERB_REG	RW	32	0x0000 003C	0x4902 603C
MCBSPLP_XCERA_REG	RW	32	0x0000 0040	0x4902 6040
MCBSPLP_XCERB_REG	RW	32	0x0000 0044	0x4902 6044
MCBSPLP_PCR_REG	RW	32	0x0000 0048	0x4902 6048
MCBSPLP_RCERC_REG	RW	32	0x0000 004C	0x4902 604C
MCBSPLP_RCERD_REG	RW	32	0x0000 0050	0x4902 6050
MCBSPLP_XCERC_REG	RW	32	0x0000 0054	0x4902 6054
MCBSPLP_XCERD_REG	RW	32	0x0000 0058	0x4902 6058
MCBSPLP_RCERE_REG	RW	32	0x0000 005C	0x4902 605C
MCBSPLP_RCERF_REG	RW	32	0x0000 0060	0x4902 6060
MCBSPLP_XCERE_REG	RW	32	0x0000 0064	0x4902 6064
MCBSPLP_XCERF_REG	RW	32	0x0000 0068	0x4902 6068
MCBSPLP_RCERG_REG	RW	32	0x0000 006C	0x4902 606C
MCBSPLP_RCERH_REG	RW	32	0x0000 0070	0x4902 6070
MCBSPLP_XCERG_REG	RW	32	0x0000 0074	0x4902 6074
MCBSPLP_XCERH_REG	RW	32	0x0000 0078	0x4902 6078
MCBSPLP_REV_REG	R	32	0x0000 007C	0x4902 607C
MCBSPLP_RINTCLR_REG	RW	32	0x0000 0080	0x4902 6080
MCBSPLP_XINTCLR_REG	RW	32	0x0000 0084	0x4902 6084
MCBSPLP_ROVFLCLR_REG	RW	32	0x0000 0088	0x4902 6088
MCBSPLP_SYSCONFIG_REG	RW	32	0x0000 008C	0x4902 608C
MCBSPLP_THRSH2_REG	RW	32	0x0000 0090	0x4902 6090
MCBSPLP_THRSH1_REG	RW	32	0x0000 0094	0x4902 6094
MCBSPLP_IRQSTATUS_REG	RW	32	0x0000 00A0	0x4902 60A0
MCBSPLP_IRQENABLE_REG	RW	32	0x0000 00A4	0x4902 60A4
MCBSPLP_WAKEUPEN_REG	RW	32	0x0000 00A8	0x4902 60A8
MCBSPLP_XCCR_REG	RW	32	0x0000 00AC	0x4902 60AC
MCBSPLP_RCCR_REG	RW	32	0x0000 00B0	0x4902 60B0
MCBSPLP_XBUFFSTAT_REG	R	32	0x0000 00B4	0x4902 60B4
MCBSPLP_RBUFFSTAT_REG	R	32	0x0000 00B8	0x4902 60B8
MCBSPLP_SSELCR_REG	RW	32	0x0000 00BC	0x4902 60BC
MCBSPLP_STATUS_REG	R	32	0x0000 00C0	0x4902 60C0

18.6.2 SIDETONE Register Mapping Summary

Table 18-42. SIDETONE_McBSP2 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
ST_REV_REG	R	32	0x0000 0000	0x4902 8000
ST_SYSCONFIG_REG	RW	32	0x0000 0010	0x4902 8010
ST_IRQSTATUS_REG	RW	32	0x0000 0018	0x4902 8018
ST_IRQENABLE_REG	RW	32	0x0000 001C	0x4902 801C
ST_SGAINCR_REG	RW	32	0x0000 0024	0x4902 8024
ST_SFIRCR_REG	RW	32	0x0000 0028	0x4902 8028
ST_SSELCR_REG	RW	32	0x0000 002C	0x4902 802C

Table 18-43. SIDETONE_McBSP3 Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
ST_REV_REG	R	32	0x0000 0000	0x4902 A000
ST_SYSCONFIG_REG	RW	32	0x0000 0010	0x4902 A010
ST_IRQSTATUS_REG	RW	32	0x0000 0018	0x4902 A018
ST_IRQENABLE_REG	RW	32	0x0000 001C	0x4902 A01C
ST_SGAINCR_REG	RW	32	0x0000 0024	0x4902 A024
ST_SFIRCR_REG	RW	32	0x0000 0028	0x4902 A028
ST_SSELCR_REG	RW	32	0x0000 002C	0x4902 A02C

18.6.3 McBSP Register Description

Table 18-44. MCBSPLP_DRR_REG

Address Offset	0x0000 0000		
Physical Address	0x4807 4000 0x4809 6000 0x4902 2000 0x4902 4000 0x4902 6000	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6000		McBSP5
	0x4902 2000		McBSP2
	0x4902 4000		McBSP3
	0x4902 6000		McBSP4
Description	McBSPLP data receive register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRR																															

Bits	Field Name	Description	Type	Reset
31:0	DRR	Data receive register	R	0x00000000

Table 18-45. Register Call Summary for Register MCBSPLP_DRR_REG

McBSP Functional Description
- Data Transfer Process for 8- / 12- / 16- / 20- / 24- / 32-bits Long Words: [0]
- Serial Words: [1]
- McBSP Reception: [2] [3]
- Introduction: [4]
- Overrun in the Receiver: [5] [6]

McBSP Basic Programming Model
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [7] [8]
- Data Packing Using Frame Length and Word Length: [9] [10]

McBSP Register Manual
- McBSP Register Mapping Summary: [13] [14] [15] [16] [17]

Table 18-46. MCBSPLP_DXR_REG

Address Offset	0x0000 0008		
Physical Address	0x4807 4008 0x4809 6008 0x4902 2008 0x4902 4008 0x4902 6008	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6008		McBSP5
	0x4902 2008		McBSP2
	0x4902 4008		McBSP3
	0x4902 6008		McBSP4
Description	McBSPLP data transmit register		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXR																															

Bits	Field Name	Description	Type	Reset
31:0	DXR	Data transmit register	W	0x00000000

Table 18-47. Register Call Summary for Register MCBSPLP_DXR_REG

McBSP Functional Description
- Data Transfer Process for 8- / 12- / 16- / 20- / 24- / 32-bits Long Words: [0]
- McBSP Transmission: [1] [2]
- Introduction: [3]
- Underflow in the Transmitter: [4] [5] [6] [7]
- Disabling/Enabling Versus Masking/Unmasking: [8]

McBSP Basic Programming Model
- McBSP Initialization Procedure: [9]
- Data Packing Using Frame Length and Word Length: [10] [11]

McBSP Register Manual
- McBSP Register Mapping Summary: [12] [13] [14] [15] [16]

Table 18-48. MCBSPLP_SPCR2_REG

Address Offset	0x0000 0010		
Physical Address	0x4807 4010 0x4809 6010 0x4902 2010 0x4902 4010 0x4902 6010	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6010		McBSP5
	0x4902 2010		McBSP2
	0x4902 4010		McBSP3
	0x4902 6010		McBSP4
Description	McBSPLP serial port control register 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						FREE	SOFT	FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Read returns 0x0.	R	0x000000
9	FREE	Free Running Mode (When this bit is set, the module ignores the Msuspend input) 0x0: Free running mode is disabled 0x1: Free running mode is enabled	RW	0x0
8	SOFT	Soft Bit 0x0: SOFT Mode is disabled: the McBSP module stops its activity immediately following MSuspend assertion. 0x1: SOFT Mode is enabled: the McBSP module freezes its state after completion of the current operation when MSuspend is asserted.	RW	0x0
7	FRST	Frame-Sync Generator Reset 0x0: Frame-synchronization logic is reset. Frame-sync signal FSG is not generated by the sample-rate generator 0x1: Frame-sync signal FSG is generated after (FPER+1) number of CLKG clocks; i.e., all frame counters are loaded with their programmed values	RW	0x0
6	GRST	Sample-Rate Generator Reset 0x0: SRG is reset 0x1: SRG is pulled out of reset. CLKG is driven as per programmed value in SRG registers (SRGR[1,2])	RW	0x0
5:4	XINTM	Transmit Interrupt Mode (legacy) 0x0: Transmit interrupt is driven by XRDY 0x1: Transmit interrupt generated by end-of-frame 0x2: Transmit interrupt generated by a new frame synchronization 0x3: Transmit interrupt generated by XSYNCERR	RW	0x0
3	XSYNCERR	Transmit Synchronization Error (writing 0 to this bit clear the legacy transmit interrupt if asserted due to XSYNCERROR condition) 0x0: No synchronization error 0x1: Synchronization error detected by McBSP	RW	0x0
2	XEMPTY	Transmit Shift Register XSR Empty 0x0: XSR is empty 0x1: XSR is not empty	R	0x0
1	XRDY	Transmitter ready 0x0: Transmitter is not ready. 0x1: Transmitter is ready for new data in DXR	R	0x0
0	XRST	Transmitter reset. This resets and enables the transmitter. 0x0: The serial port transmitter is disabled and in reset state. 0x1: The serial port transmitter is enabled.	RW	0x0

Table 18-49. Register Call Summary for Register MCBSPLP_SPCR2_REG

McBSP Integration • Hardware and Software Reset: [0] [1] [2]
McBSP Functional Description • Data Transfer Process for 8- / 12- / 16- / 20- / 24- / 32-bits Long Words: [3] • McBSP Transmission: [4] [5] • Clock Generation in the SRG: [6] • Frame Sync Generation in the SRG: [7] [8] • Introduction: [9] [10] [11] • Underflow in the Transmitter: [12] [13] [14] [15] [16] • Possible Responses to Transmit Frame-sync Pulses: [17] [18] [19] • McBSP DMA Configuration: [20] [21] • Disabling/Enabling Versus Masking/Unmasking: [22] [23] [24]
McBSP Basic Programming Model • McBSP Initialization Procedure: [25] [26] [27] [28] • Reset and Initialization Procedure for the Sample Rate Generator: [29] [30] [31] [32] [33] • Legacy Interrupt Line: [34] [35] [36] • Resetting (Step 1) and Enabling (Step 3) the Receiver: [37] [38] [39] [40] [41] • Resetting (Step 1) and Enabling (Step 3) the Transmitter: [42] [43] [44] [45] [46] [47] • Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [48] • General-Purpose I/O on the McBSP Pins (Legacy Only): [49] [50]
McBSP Register Manual • McBSP Register Mapping Summary: [53] [54] [55] [56] [57]

Table 18-50. MCBSPLP_SPCR1_REG

Address Offset	0x0000 0014		
Physical Address	0x4807 4014 0x4809 6014 0x4902 2014 0x4902 4014 0x4902 6014	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6014		McBSP5
	0x4902 2014		McBSP2
	0x4902 4014		McBSP3
	0x4902 6014		McBSP4
Description	McBSPLP serial port control register 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																ALB	RJUST	RESERVED						DXENA	RESERVED	RINTM		RSYNCERR	RFULL	RRDY	RRST

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15	ALB	Analog loopback Mode	RW	0x0
		0x0: Analog loopback mode disabled		
		0x1: Analog loopback mode enabled		

Bits	Field Name	Description	Type	Reset
14:13	RJUST	Receive Sign-Extension and Justification Mode 0x0: Right-justify and zero-fill MSBs in DRR 0x1: Right-justify and sign-extend MSBs in DRR 0x2: Left-justify and zero-fill LSBs in DRR 0x3: Reserved	RW	0x0
12:8	RESERVED	Read returns 0x0.	R	0x00
7	DXENA	DX Enabler 0x0: DX enabler is off 0x1: DX enabler is on	RW	0x0
6	RESERVED	Read returns 0x0.	R	0x0
5:4	RINTM	Receive Interrupt Mode (legacy) 0x0: Receive Interrupt driven by RRDY (i.e. end of word) and end of frame in A-bis mode 0x1: Receive Interrupt generated by end-of-block or end-of-frame in multichannel operation 0x2: Receive Interrupt generated by a new frame synchronization 0x3: Receive Interrupt generated by RSYNCERR	RW	0x0
3	RSYNCERR	Receive Synchronization Error (writing 0 to this bit clear the legacy receive interrupt if asserted due to RSYNCERROR condition) 0x0: No synchronization error 0x1: Synchronization error detected by McBSP	RW	0x0
2	RFULL	Receive Shift Register (RSR) Full 0x0: RB is not in overrun condition 0x1: DRR is not read, RB is full and RSR is also full with new word	R	0x0
1	RRDY	Receiver Ready 0x0: Receiver is not ready 0x1: Receiver is ready with data to be read from DRR	R	0x0
0	RRST	Receiver reset. This resets and enables the receiver. 0x0: The serial port receiver is disabled and in reset state. 0x1: The serial port receiver is enabled.	RW	0x0

Table 18-51. Register Call Summary for Register MCBSPLP_SPCR1_REG

McBSP Integration
- Hardware and Software Reset: [0]

McBSP Functional Description
- McBSP Reception: [1] [2] [3]
- Clock Generation in the SRG: [4]
- Introduction: [5] [6] [7] [8]
- Overrun in the Receiver: [9] [10] [11]
- Possible Responses to Receive Frame-sync Pulses: [12] [13] [14]
- McBSP DMA Configuration: [15] [16]
- Receive Multichannel Selection Mode: [17]

Table 18-51. Register Call Summary for Register MCBSPLP_SPCR1_REG (continued)

McBSP Basic Programming Model

- McBSP Initialization Procedure: [18] [19]
- Reset and Initialization Procedure for the Sample Rate Generator: [20] [21] [22]
- Legacy Interrupt Line: [23] [24] [25] [26]
- Resetting (Step 1) and Enabling (Step 3) the Receiver: [27] [28]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [29] [30] [31] [32] [33]
- Resetting (Step 1) and Enabling (Step 3) the Transmitter: [34]
- Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [35]
- General-Purpose I/O on the McBSP Pins (Legacy Only): [36] [37]

McBSP Register Manual

- McBSP Register Mapping Summary: [42] [43] [44] [45] [46]
- McBSP Register Description: [47]

Table 18-52. MCBSPLP_RCR2_REG

Address Offset	0x0000 0018		
Physical Address	0x4807 4018	**Instance**	McBSP1
	0x4809 6018		McBSP5
	0x4902 2018		McBSP2
	0x4902 4018		McBSP3
	0x4902 6018		McBSP4
	0x4809 6018		McBSP5
	0x4902 2018		McBSP2
	0x4902 4018		McBSP3
	0x4902 6018		McBSP4
Description	McBSPLP receive control register 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RPHASE	RFRLEN2							RWDLEN2			RREVERSE		RESERVED		RDATDLY

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15	RPHASE	Receive Phases	RW	0x0
		0x0: Single-phase frame		
		0x1: Dual-phase frame		
14:8	RFRLEN2	Receive Frame Length 2 Single-phase frame selected: RFRLEN2=don't care Dual-phase frame selected: RFRLEN2=000 0000 - 1 word per second phase (other values are reserved)	RW	0x00
7:5	RWDLEN2	Receive Word Length 2	RW	0x0
		0x0: 8 bits		
		0x1: 12 bits		
		0x2: 16 bits		
		0x3: 20 bits		
		0x4: 24 bits		
		0x5: 32 bits		
		0x6: Reserved (do not use)		
		0x7: Reserved (do not use)		

Bits	Field Name	Description	Type	Reset
4:3	RREVERSE	Receive reverse mode. 0x0: Data transfer starts with MSB first. 0x1: Data transfer starts with LSB first. 0x2: Reserved (do not use) 0x3: Reserved (do not use)	RW	0x0
2	RESERVED	Read returns 0x0.	R	0x0
1:0	RDATDLY	Receive Data Delay 0x0: 0-bit data delay 0x1: 1-bit data delay 0x2: 2-bit data delay 0x3: Reserved	RW	0x0

Table 18-53. Register Call Summary for Register MCBSPLP_RCR2_REG

McBSP Environment
- Words or Channels: [0]
- Frames: [1] [2]

McBSP Functional Description
- Bit Reordering (Option to Transfer LSB First): [3]
- Serial Words: [4]
- Frames and Frame Synchronization: [5]
- Number of Phases, Words, and Bits per Frame: [6] [7] [8] [9]
- Single-Phase Frame Example: [10] [11] [12]
- Dual-Phase Frame Example: [13] [14] [15]
- McBSP Reception: [16]
- MCBSP Data Transfer Mode: [17]
- Preventing Unexpected Receive Frame-sync Pulses: [18]
- Configuring a Frame for Multichannel Selection: [19]
- SIDETONE Interface: [20]

McBSP Basic Programming Model
- McBSP Initialization Procedure: [21]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [22] [23] [24] [25] [26] [27]
- SIDETONE Activation Procedure: [28]

McBSP Register Manual
- McBSP Register Mapping Summary: [32] [33] [34] [35] [36]

Table 18-54. MCBSPLP_RCR1_REG

Address Offset	0x0000 001C		
Physical Address	0x4807 401C 0x4809 601C 0x4902 201C 0x4902 401C 0x4902 601C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 601C		McBSP5
	0x4902 201C		McBSP2
	0x4902 401C		McBSP3
	0x4902 601C		McBSP4
Description	McBSPLP receive control register 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RFRLEN1								RWDLEN1			RESERVED				

Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns 0x0.	R	0x00000
14:8	RFRLEN1	Receive Frame Length 1 Single-phase frame selected: RFRLEN1=000 0000 - 1 word per frame RFRLEN1=000 0001 - 2 words per frame RFRLEN1=111 1111 - 128 words per frame Dual-phase frame selected: RFRLEN1=000 0000 - 1 word per phase (other values are reserved)	RW	0x00
7:5	RWDLEN1	Receive Word Length 1 0x0: 8 bits 0x1: 12 bits 0x2: 16 bits 0x3: 20 bits 0x4: 24 bits 0x5: 32 bits 0x6: Reserved (do not use) 0x7: Reserved (do not use)	RW	0x0
4:0	RESERVED	Read returns 0x0.	R	0x00

Table 18-55. Register Call Summary for Register MCBSPLP_RCR1_REG

McBSP Environment
- Words or Channels: [0]
- Frames: [1]

McBSP Functional Description
- Serial Words: [2]
- Frames and Frame Synchronization: [3]
- Number of Phases, Words, and Bits per Frame: [4] [5] [6]
- Single-Phase Frame Example: [7] [8]
- Dual-Phase Frame Example: [9] [10]
- Configuring a Frame for Multichannel Selection: [11]
- SIDETONE Interface: [12] [13]

McBSP Basic Programming Model
- McBSP Initialization Procedure: [14]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [15] [16]
- SIDETONE Activation Procedure: [17] [18]

McBSP Register Manual
- McBSP Register Mapping Summary: [22] [23] [24] [25] [26]

Table 18-56. MCBSPLP_XCR2_REG

Address Offset	0x0000 0020		
Physical Address	0x4807 4020	**Instance**	McBSP1
	0x4809 6020		McBSP5
	0x4902 2020		McBSP2
	0x4902 4020		McBSP3
	0x4902 6020		McBSP4
	0x4809 6020		McBSP5
	0x4902 2020		McBSP2
	0x4902 4020		McBSP3
	0x4902 6020		McBSP4
Description	McBSPLP transmit control register 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XPHASE	XFRLEN2							XWDLEN2			XREVERSE		RESERVED	XDATDLY	

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15	XPHASE	Transmit Phases	RW	0x0
		0x0: Single-phase frame		
		0x1: Dual-phase frame		
14:8	XFRLEN2	Transmit Frame Length 2 Single-phase frame selected: XFRLEN2=don't care Dual-phase frame selected: XFRLEN2=000 0000 - 1 word per second phase (other values are reserved)	RW	0x00
7:5	XWDLEN2	Transmit Word Length 2	RW	0x0
		0x0: 8 bits		
		0x1: 12 bits		
		0x2: 16 bits		
		0x3: 20 bits		
		0x4: 24 bits		
		0x5: 32 bits		
		0x6: Reserved (do not use)		
		0x7: Reserved (do not use)		
4:3	XREVERSE	Transmit reverse mode.	RW	0x0
		0x0: Data transfer starts with MSB first.		
		0x1: Data transfer starts with LSB first.		
		0x2: Reserved (do not use)		
		0x3: Reserved (do not use)		
2	RESERVED	Read returns 0x0.	R	0x0
1:0	XDATDLY	Transmit Data Delay	RW	0x0
		0x0: 0-bit data delay		
		0x1: 1-bit data delay		
		0x2: 2-bit data delay		
		0x3: Reserved		

Table 18-57. Register Call Summary for Register MCBSPLP_XCR2_REG

McBSP Environment • Words or Channels: [0] • Frames: [1] [2]
McBSP Functional Description • Bit Reordering (Option to Transfer LSB First): [3] • Serial Words: [4] • Frames and Frame Synchronization: [5] • Maximum Frame Frequency: [6] • Number of Phases, Words, and Bits per Frame: [7] [8] [9] [10] • Single-Phase Frame Example: [11] [12] [13] • Dual-Phase Frame Example: [14] [15] [16] • McBSP Transmission: [17] • MCBSP Data Transfer Mode: [18] • Preventing Unexpected Transmit Frame-sync Pulses: [19] • Configuring a Frame for Multichannel Selection: [20] • SIDETONE Interface: [21]
McBSP Basic Programming Model • McBSP Initialization Procedure: [22] • Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [23] [24] [25] [26] [27] • SIDETONE Activation Procedure: [28]
McBSP Register Manual • McBSP Register Mapping Summary: [29] [30] [31] [32] [33]

Table 18-58. MCBSPLP_XCR1_REG

Address Offset	0x0000 0024		
Physical Address	0x4807 4024 0x4809 6024 0x4902 2024 0x4902 4024 0x4902 6024	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6024		McBSP5
	0x4902 2024		McBSP2
	0x4902 4024		McBSP3
	0x4902 6024		McBSP4
Description	McBSPLP transmit control register 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XFRLEN1								XWDLEN1			RESERVED				

Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns 0x0.	R	0x00000
14:8	XFRLEN1	Transmit Frame Length 1 Single-phase frame selected: XFRLEN1=000 0000 - 1 word per frame XFRLEN1=000 0001 - 2 words per frame XFRLEN1=111 1111 - 128 words per frame Dual-phase frame selected: XFRLEN1=000 0000 - 1 word per phase (other values are reserved)	RW	0x00

Bits	Field Name	Description	Type	Reset
7:5	XWDLEN1	Transmit Word Length 1 0x0: 8 bits 0x1: 12 bits 0x2: 16 bits 0x3: 20 bits 0x4: 24 bits 0x5: 32 bits 0x6: Reserved (do not use) 0x7: Reserved (do not use)	RW	0x0
4:0	RESERVED	Read returns 0x0.	R	0x00

Table 18-59. Register Call Summary for Register MCBSPLP_XCR1_REG

McBSP Environment
- Words or Channels: [0]
- Frames: [1]

McBSP Functional Description
- Serial Words: [2]
- Frames and Frame Synchronization: [3]
- Number of Phases, Words, and Bits per Frame: [4] [5] [6]
- Single-Phase Frame Example: [7] [8]
- Dual-Phase Frame Example: [9] [10]
- Configuring a Frame for Multichannel Selection: [11]
- SIDETONE Interface: [12] [13]

McBSP Basic Programming Model
- McBSP Initialization Procedure: [14]
- Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [15] [16]
- SIDETONE Activation Procedure: [17] [18]

McBSP Register Manual
- McBSP Register Mapping Summary: [19] [20] [21] [22] [23]

Table 18-60. MCBSPLP_SRGR2_REG

Address Offset	0x0000 0028		
Physical Address	0x4807 4028 0x4809 6028 0x4902 2028 0x4902 4028 0x4902 6028	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6028		McBSP5
	0x4902 2028		McBSP2
	0x4902 4028		McBSP3
	0x4902 6028		McBSP4
Description	McBSPLP SRG register 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																GSYNC	CLKSP	CLKSM	FSGM	FPER											

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15	GSYNC	Sample Rate Generator Synchronization Only used when the external clock (CLKS) drives the SRG clock (CLKSM=0) 0x0: The SRG clock (CLKG) is free running. 0x1: The SRG clock (CLKG) is running. But 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.	RW	0x0
14	CLKSP	CLKS Polarity Clock Edge Select Only used when the external clock CLKS drives the SRG clock (CLKSM=0). 0x0: Rising edge of CLKG and FSG. 0x1: Falling edge of CLKG and FSG.	RW	0x0
13	CLKSM	McBSP SRG Clock Mode 0x0: SCLKME=0: SRG clock derived from the CLKS pin. SCLKME=1: SRG clock derived from the CLKR input pin. 0x1: SCLKME=0: SRG clock derived from the McBSPi_ICLK clock. SCLKME=1: SRG clock derived from the CLKX input pin.	RW	0x1
12	FSGM	Sample Rate Generator Transmit Frame-Synchronization Mode Used when FSXM=1 in the PCR. 0x0: Transmit frame-sync signal (FSX) is generated when transmit buffer is not empty. When FSGM=0, FPER and FWID are used to determine the frame synchronization period and width (external FSX is gated by the buffer empty condition). 0x1: Transmit frame-sync signal driven by the SRG frame-sync signal, FSG.	RW	0x0
11:0	FPER	Frame Period. This value + 1 determines when the next frame-sync signal becomes active. Range: 1 to 4096 CLKG periods	RW	0x000

Table 18-61. Register Call Summary for Register MCBSPLP_SRGR2_REG

McBSP Integration

- McBSP1 Clocks: [0]
- McBSP2 Clocks: [1]
- McBSP3 Clocks: [2]
- McBSP4 Clocks: [3]
- McBSP5 Clocks: [4]

McBSP Functional Description

- Clocking and Framing Data: [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16]
- Frames and Frame Synchronization: [17] [18]
- McBSP SRG: [19] [20] [21] [22]
- Frame Sync Generation in the SRG: [23] [24]
- Controlling the Period Between the Starting Edges of Frame Sync Pulses: [25] [26]
- Keeping FSG Synchronized to an External Clock: [27] [28] [29]
- Synchronizing SRG Outputs to an External Clock: [30] [31] [32] [33] [34]
- Operating the Transmitter Synchronously with the Receiver: [35] [36]
- Synchronization Examples: [37] [38]
- Underflow in the Transmitter: [39] [40] [41]

Table 18-61. Register Call Summary for Register MCBSPLP_SRGR2_REG (continued)

McBSP Basic Programming Model • McBSP Initialization Procedure: [42] • Reset and Initialization Procedure for the Sample Rate Generator: [43] • Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [44] [45] [46] [47] [48] [49] • Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [50] [51]
McBSP Register Manual • McBSP Register Mapping Summary: [52] [53] [54] [55] [56]

Table 18-62. MCBSPLP_SRGR1_REG

Address Offset	0x0000 002C		
Physical Address	0x4807 402C 0x4809 602C 0x4902 202C 0x4902 402C 0x4902 602C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 602C		McBSP5
	0x4902 202C		McBSP2
	0x4902 402C		McBSP3
	0x4902 602C		McBSP4
Description	McBSPLP SRG register 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																FWID								CLKGDV							

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:8	FWID	Frame Width. This value + 1 determines the width of the frame-sync pulse, FSG, during its active period. Range: 1 to 256 CLKG periods.	RW	0x00
7:0	CLKGDV	Sample Rate Generator Clock Divider This value is used as the divide-down number to generate the required SRG clock frequency. Default value is 1.	RW	0x01

Table 18-63. Register Call Summary for Register MCBSPLP_SRGR1_REG

McBSP Functional Description • Clocking: [0] • McBSP SRG: [1] [2] [3] • Frame Sync Generation in the SRG: [4] • Choosing the Width of the Frame-sync Pulse: [5] • Synchronization Examples: [6] [7] • Underflow in the Transmitter: [8]
McBSP Basic Programming Model • McBSP Initialization Procedure: [9] • Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [10] [11]
McBSP Register Manual • McBSP Register Mapping Summary: [12] [13] [14] [15] [16]

Table 18-64. MCBSPLP_MCR2_REG

Address Offset	0x0000 0030		
Physical Address	0x4807 4030 0x4809 6030 0x4902 2030 0x4902 4030 0x4902 6030	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6030		McBSP5
	0x4902 2030		McBSP2
	0x4902 4030		McBSP3
	0x4902 6030		McBSP4
Description	McBSPLP multi channel register 2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						XMCME	XPBBLK	XPABLK				RESERVED		XMCM	

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Read returns 0x0.	R	0x000000
9	XMCME	Transmit multichannel partition mode determines whether only 32 channels or all 128 channels are to be individually selectable. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). 0x0: 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 bits. If XMCM = 01b or 10b, assign 16 channels to partition A with the XPABLK bits. Assign 16 channels to partition B with the XPBBLK bits. If XMCM = 11b (for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bits. Assign 16 channels to receive partition B with the RPBBLK bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels in partition A XCERB: Channels in partition B 0x1: 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 bits. You control the channels with the appropriate transmit channel enable registers: XCERA: Channels 0 through 15 XCERB: Channels 16 through 31 XCERC: Channels 32 through 47 XCERD: Channels 48 through 63 XCERE: Channels 64 through 79 XCERF: Channels 80 through 95 XCERG: Channels 96 through 111 XCERH: Channels 112 through 127	RW	0x0

Bits	Field Name	Description	Type	Reset
8:7	XPBBLK	Transmit Partition B Block (legacy) 0x0: Block 1. Channel 16 to channel 31 0x1: Block 3. Channel 48 to channel 63 0x2: Block 5. Channel 80 to channel 95 0x3: Block 7. Channel 112 to channel 127	RW	0x0
6:5	XPABLK	Transmit Partition A Block (legacy) 0x0: Block 0. Channel 0 to channel 15 0x1: Block 2. Channel 32 to channel 47 0x2: Block 4. Channel 64 to channel 79 0x3: Block 6. Channel 96 to channel 111	RW	0x0
4:2	RESERVED	Reserved	R	0x0
1:0	XMCM	Transmit Multichannel Selection Enable 0x0: All channels enabled without masking (DX is always driven during transmission of data). 0x1: All channels disabled and therefore masked by default. Required channels are selected by enabling XP(A/B)BLK and XCER(A/B) appropriately. Also, these selected channels are not masked and therefore DX is always driven. 0x2: All channels enabled, but masked. Selected channels enabled via XP(A/B)BLK and XCER(A/B) are unmasked. 0x3: All channels disabled and therefore masked by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately. Selected channels can be unmasked by RP(A/B)BLK and XCER(A/B). This mode is used for symmetric transmit and receive operation.	RW	0x0

Table 18-65. Register Call Summary for Register MCBSPLP_MCR2_REG

McBSP Functional Description • Using Eight Partitions: [0] [1] [2] • Using Two Partitions (Legacy Only): [3] [4] • Transmit Multichannel Selection Modes: [5] [6] [7] [8] • Activity on McBSP Pins for Different Values of XMCM: [9] [10]
McBSP Basic Programming Model • McBSP Initialization Procedure: [11] • Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [12]
McBSP Register Manual • McBSP Register Mapping Summary: [13] [14] [15] [16] [17]

Table 18-66. MCBSPLP_MCR1_REG

Address Offset	0x0000 0034		
Physical Address	0x4807 4034 0x4809 6034 0x4902 2034 0x4902 4034 0x4902 6034	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6034		McBSP5
	0x4902 2034		McBSP2
	0x4902 4034		McBSP3
	0x4902 6034		McBSP4
Description	McBSPLP multi channel register 1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						RMCME	RPBBLK	RPABLK	RESERVED						RMCM

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Read returns 0x0.	R	0x000000
9	RMCME	Receive multichannel partition mode determines whether only 32 channels or all 128 channels are to be individually selectable. RMCME is only applicable if channels can be individually enabled or disabled for reception (RMCM = 1). 0x0: 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 RPABLK bits. Assign 16 channels to partition B with the RPBBLK bits. You control the channels with the appropriate receive channel enable registers: RCERA: Channels in partition A RCERB: Channels in partition B 0x1: 8-partition mode: All partitions (A through H) are used. You can control up to 128 channels in the receive multichannel selection mode. You control the channels with the appropriate receive channel enable registers: RCERA: Channels 0 through 15 RCERB: Channels 16 through 31 RCERC: Channels 32 through 47 RCERD: Channels 48 through 63 RCERE: Channels 64 through 79 RCERF: Channels 80 through 95 RCERG: Channels 96 through 111 RCERH: Channels 112 through 127	RW	0x0
8:7	RPBBLK	Receive Partition B Block (legacy) 0x0: Block 1. Channel 16 to channel 31 0x1: Block 3. Channel 48 to channel 63 0x2: Block 5. Channel 80 to channel 95 0x3: Block 7. Channel 112 to channel 127	RW	0x0

Bits	Field Name	Description	Type	Reset
6:5	RPABLK	Receive Partition A Block (legacy) 0x0: Block 0. Channel 0 to channel 15 0x1: Block 2. Channel 32 to channel 47 0x2: Block 4. Channel 64 to channel 79 0x3: Block 6. Channel 96 to channel 111	RW	0x0
4:1	RESERVED	Read returns 0x0.	R	0x0
0	RMCM	Receive Multichannel Selection Enable 0x0: All 128 channels 0x1: All channels disabled by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately	RW	0x0

Table 18-67. Register Call Summary for Register MCBSPLP_MCR1_REG

McBSP Functional Description
- Using Eight Partitions: [0] [1] [2]
- Receive Multichannel Selection Mode: [3] [4]
- Using Two Partitions (Legacy Only): [5] [6] [7] [8]
- SIDETONE Interface: [9]

McBSP Basic Programming Model
- McBSP Initialization Procedure: [10]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [11]
- SIDETONE Activation Procedure: [12]

McBSP Register Manual
- McBSP Register Mapping Summary: [13] [14] [15] [16] [17]

Table 18-68. MCBSPLP_RCERA_REG

Address Offset	0x0000 0038		
Physical Address	0x4807 4038 0x4809 6038 0x4902 2038 0x4902 4038 0x4902 6038 0x4809 6038 0x4902 2038 0x4902 4038 0x4902 6038	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4 McBSP5 McBSP2 McBSP3 McBSP4
Description	McBSPLP receive channel enable register partition A		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERA															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERA	Receive Channel Enable RCERA n=0 Disables reception of n-th channel in an even-numbered block in partition A RCERA n=1 Enables reception of n-th channel in an even-numbered block in partition A	RW	0x0000

Table 18-69. Register Call Summary for Register MCBSPLP_RCERA_REG

McBSP Functional Description
- Using Eight Partitions: [0]
- Receive Multichannel Selection Mode: [1] [2]
- Using Two Partitions (Legacy Only): [3]
- Transmit Multichannel Selection Modes: [4] [5]

McBSP Register Manual
- McBSP Register Mapping Summary: [6] [7] [8] [9] [10]

Table 18-70. MCBSPLP_RCERB_REG

Address Offset	0x0000 003C		
Physical Address	0x4807 403C 0x4809 603C 0x4902 203C 0x4902 403C 0x4902 603C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 603C		McBSP5
	0x4902 203C		McBSP2
	0x4902 403C		McBSP3
	0x4902 603C		McBSP4
Description	McBSPLP receive channel enable register partition B		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERB															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERB	Receive Channel Enable RCERB n=0 Disables reception of n-th channel in a even-numbered block in partition B RCERB n=1 Enables reception of n-th channel in a even-numbered block in partition B	RW	0x0000

Table 18-71. Register Call Summary for Register MCBSPLP_RCERB_REG

McBSP Functional Description
- Using Eight Partitions: [0]
- Using Two Partitions (Legacy Only): [1]

McBSP Register Manual
- McBSP Register Mapping Summary: [2] [3] [4] [5] [6]

Table 18-72. MCBSPLP_XCERA_REG

Address Offset	0x0000 0040		
Physical Address	0x4807 4040 0x4809 6040 0x4902 2040 0x4902 4040 0x4902 6040	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6040		McBSP5
	0x4902 2040		McBSP2
	0x4902 4040		McBSP3
	0x4902 6040		McBSP4
Description	McBSPLP transmit channel enable register partition A		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERA															
R-0																RW-0															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERA	Transmit Channel Enable	RW	0x0000
		XCERA n=0 Disables transmission of n-th channel in an event-numbered block in partition A		
		XCERA n=1 Enables transmission of n-th channel in an event-numbered block in partition A		

Table 18-73. Register Call Summary for Register MCBSPLP_XCERA_REG

McBSP Functional Description

- Using Eight Partitions: [0]
- Using Two Partitions (Legacy Only): [1]
- Transmit Multichannel Selection Modes: [2] [3] [4] [5] [6] [7]

McBSP Register Manual

- McBSP Register Mapping Summary: [8] [9] [10] [11] [12]

Table 18-74. MCBSPLP_XCERB_REG

Address Offset	0x0000 0044		
Physical Address	0x4807 4044 0x4809 6044 0x4902 2044 0x4902 4044 0x4902 6044	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6044		McBSP5
	0x4902 2044		McBSP2
	0x4902 4044		McBSP3
	0x4902 6044		McBSP4
Description	McBSPLP transmit channel enable register partition B		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERB															
R-0																RW-0															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERB	Transmit Channel Enable XCERB n=0 Disables transmission of n-th channel in an even-numbered block in partition B XCERB n=1 Enables transmission of n-th channel in an even-numbered block in partition B	RW	0x0000

Table 18-75. Register Call Summary for Register MCBSPLP_XCERB_REG

McBSP Functional Description
- Using Eight Partitions: [0]
- Using Two Partitions (Legacy Only): [1]

McBSP Register Manual
- McBSP Register Mapping Summary: [2] [3] [4] [5] [6]

Table 18-76. MCBSPLP_PCR_REG

Address Offset	0x0000 0048		
Physical Address	0x4807 4048 0x4809 6048 0x4902 2048 0x4902 4048 0x4902 6048 0x4809 6048 0x4902 2048 0x4902 4048 0x4902 6048	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4 McBSP5 McBSP2 McBSP3 McBSP4
Description	McBSPLP pin control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																IDLE_EN	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM	SCLKME	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP	

Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns 0x0.	R	0x00000
14	IDLE_EN	Idle enable. This bit allows stopping all the clocks in the MCBSP module. (legacy) 0x0: The McBSP is running 0x1: The clocks in the McBSP are shut off when both IDLE_EN=1 and his power domain is in idle mode (Force idle or Smart idle)	RW	0x0
13	XIOEN	Transmit General Purpose I/O Mode only when XRST=0 in SPCR[1,2] (legacy) 0x0: DX, FSX and CLKX are configured as serial port pins and do not function as general-purpose I/Os. 0x1: DX pin is a general purpose output. FSX and CLKX are general purpose I/Os. These serial port pins do not perform serial port operation.	RW	0x0

Bits	Field Name	Description	Type	Reset
12	RIOEN	Receive General Purpose I/O Mode when RRST=0 in SPCR[1,2] (legacy) 0x0: DR, FSR, CLKR and CLKS are configured as serial port pins and do not function as general-purpose I/Os. 0x1: DR and CLKS pins are general purpose inputs; FSR and CLKR are general purpose I/Os. These serial port pins do not perform serial port operation. The CLKS pin is affected by a combination of RRST and RIOEN signals of the receiver.	RW	0x0
11	FSXM	Transmit Frame-Synchronization Mode 0x0: Frame-synchronization signal derived from an external source 0x1: Frame synchronization is determined by the SRG frame-synchronization mode bit FSGM in SRGR2.	RW	0x0
10	FSRM	Receive Frame-Synchronization Mode 0x0: Frame-Synchronization pulses generated by an external device. FSR is an input pin. 0x1: Frame synchronization generated internally by SRG. FSR is an output pin except when GSYNC=1 in SRGR.	RW	0x0
9	CLKXM	Transmitter Clock Mode When digital loop-back mode set (McBSPi.MCBSPLP_XCCR_REG[5] DLB=1), CLKXM bit is ignored. The internal transmit clock (not the mcbspi_clkx pin) is driven by the internal SRG and mcbspi_clkx pin is in high-impedance. 0x0: Transmitter clock is driven by an external clock with CLKX as an input pin. 0x1: CLKX is an output pin and is driven by the internal SRG.	RW	0x0
8	CLKRM	Receiver Clock Mode When digital loop-back mode set (McBSPi.MCBSPLP_XCCR_REG[5] DLB=1), CLKRM bit is ignored. The internal receive lock (not the mcbsp1_clkr pin) is driven by the internal SRG and mcbsp1_clkr pin is in high-impedance. 0x0: Receive clock is an input driven by an external clock with CLKR as an input pin. 0x1: CLKR is an output pin and is driven by the internal SRG.	RW	0x0
7	SCLKME	The frequency of CLKG is: CLKG frequency = (Input clock frequency) / (CLKGDV + 1) SCLKME is used in conjunction with the CLKSM bit to select the input clock: 0x0: CLKSM = 0: Signal on CLKS pin CLKSM = 1: McBSPi_ICLK clock 0x1: CLKSM = 0: Signal on CLKR pin CLKSM = 1: Signal on CLKX pin	RW	0x0
6	CLKS_STAT	CLKS pin status. Reflects value on CLKS pin when selected as a general purpose input. (legacy) 0x0: The signal on the CLKS pin is low 0x1: The signal on the CLKS pin is high	R	0x0

Bits	Field Name	Description	Type	Reset
5	DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output. (legacy) 0x0: Drive the signal on the DX pin low 0x1: Drive the signal on the DX pin high	RW	0x0
4	DR_STAT	DR pin status. Reflects value on DR pin when selected as a general purpose input. (legacy) 0x0: The signal on DR pin is low 0x1: The signal on DR pin is high	R	0x0
3	FSXP	Transmit Frame-Synchronization Polarity 0x0: Frame-synchronization pulse FSX is active high 0x1: Frame-synchronization pulse FSX is active low	RW	0x0
2	FSRP	Receive Frame-Synchronization Polarity 0x0: Frame-synchronization pulse FSR is active high 0x1: Frame-synchronization pulse FSR is active low	RW	0x0
1	CLKXP	Transmit Clock Polarity 0x0: Transmit data driven on rising edge of CLKX 0x1: Transmit data driven on falling edge of CLKX	RW	0x0
0	CLKRP	Receive Clock Polarity 0x0: Receive data sampled on falling edge of CLKR 0x1: Receive data sampled on rising edge of CLKR	RW	0x0

Table 18-77. Register Call Summary for Register MCBSPLP_PCR_REG

McBSP Environment
- Words or Channels: [0]
- Frames: [1]

McBSP Integration
- McBSP1 Clocks: [2]
- McBSP2 Clocks: [3]
- McBSP3 Clocks: [4]
- McBSP4 Clocks: [5]
- McBSP5 Clocks: [6]
- McBSP Acknowledgment Modes: [7]
- Analysis of the Receiver Smart Idle Behavior: [8] [9]

McBSP Functional Description
- Clocking and Framing Data: [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
- Clocking: [24] [25] [26] [27] [28] [29] [30]
- Frames and Frame Synchronization: [31] [32] [33] [34] [35] [36] [37]
- Detecting Frame-Synchronization Pulses, Even in Reset State: [38] [39] [40] [41]
- Single-Phase Frame Example: [42] [43] [44] [45]
- Dual-Phase Frame Example: [46] [47] [48] [49]
- MCBSP Data Transfer Mode: [50] [51]
- McBSP SRG: [52] [53] [54] [55] [56]
- Clock Generation in the SRG: [57] [58] [59] [60]
- Frame Sync Generation in the SRG: [61] [62]
- Operating the Transmitter Synchronously with the Receiver: [63] [64] [65] [66]
- Underflow in the Transmitter: [67]

Table 18-77. Register Call Summary for Register MCBSPLP_PCR_REG (continued)

McBSP Basic Programming Model • McBSP Initialization Procedure: [68] • Reset and Initialization Procedure for the Sample Rate Generator: [69] [70] • Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] • Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [83] [84] [85] [86] [87] [88] • General-Purpose I/O on the McBSP Pins (Legacy Only): [89] [90] [91] [92] [93] [94]
McBSP Register Manual • McBSP Register Mapping Summary: [100] [101] [102] [103] [104]

Table 18-78. MCBSPLP_RCERC_REG

Address Offset	0x0000 004C		
Physical Address	0x4807 404C 0x4809 604C 0x4902 204C 0x4902 404C 0x4902 604C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 604C		McBSP5
	0x4902 204C		McBSP2
	0x4902 404C		McBSP3
	0x4902 604C		McBSP4
Description	McBSPLP receive channel enable register partition C		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERC															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERC	Receive Channel Enable RCERC n=0 Disables reception of n-th channel in an even-numbered block in partition C RCERC n=1 Enables reception of n-th channel in an even-numbered block in partition C	RW	0x0000

Table 18-79. Register Call Summary for Register MCBSPLP_RCERC_REG

McBSP Functional Description • Using Eight Partitions: [0]
McBSP Register Manual • McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-80. MCBSPLP_RCERD_REG

Address Offset	0x0000 0050		
Physical Address	0x4807 4050 0x4809 6050 0x4902 2050 0x4902 4050 0x4902 6050	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6050		McBSP5
	0x4902 2050		McBSP2
	0x4902 4050		McBSP3
	0x4902 6050		McBSP4
Description	McBSPLP receive channel enable register partition D		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERD															
R-0																R/W-0															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERD	Receive Channel Enable	RW	0x0000
		RCERD n=0 Disables reception of n-th channel in an even-numbered block in partition D		
		RCERD n=1 Enables reception of n-th channel in an even-numbered block in partition D		

Table 18-81. Register Call Summary for Register MCBSPLP_RCERD_REG

McBSP Functional Description
- Using Eight Partitions: [0]

McBSP Register Manual
- McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-82. MCBSPLP_XCERC_REG

Address Offset	0x0000 0054		
Physical Address	0x4807 4054 0x4809 6054 0x4902 2054 0x4902 4054 0x4902 6054	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6054		McBSP5
	0x4902 2054		McBSP2
	0x4902 4054		McBSP3
	0x4902 6054		McBSP4
Description	McBSPLP transmit channel enable register partition C		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERC															
R-0																R/W-0															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERC	Transmit Channel Enable XCERC n=0 Disables transmission of n-th channel in an event-numbered block in partition C XCERC n=1 Enables transmission of n-th channel in an event-numbered block in partition C	RW	0x0000

Table 18-83. Register Call Summary for Register MCBSPLP_XCERC_REG

McBSP Functional Description • Using Eight Partitions: [0]
McBSP Register Manual • McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-84. MCBSPLP_XCERD_REG

Address Offset	0x0000 0058		
Physical Address	0x4807 4058 0x4809 6058 0x4902 2058 0x4902 4058 0x4902 6058 0x4809 6058 0x4902 2058 0x4902 4058 0x4902 6058	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4 McBSP5 McBSP2 McBSP3 McBSP4
Description	McBSPLP transmit channel enable register partition D		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERD															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERD	Transmit Channel Enable XCERD n=0 Disables transmission of n-th channel in an even-numbered block in partition D XCERD n=1 Enables transmission of n-th channel in an even-numbered block in partition D	RW	0x0000

Table 18-85. Register Call Summary for Register MCBSPLP_XCERD_REG

McBSP Functional Description • Using Eight Partitions: [0]
McBSP Register Manual • McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-86. MCBSPLP_RCERE_REG

Address Offset	0x0000 005C		
Physical Address	0x4807 405C 0x4809 605C 0x4902 205C 0x4902 405C 0x4902 605C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 605C		McBSP5
	0x4902 205C		McBSP2
	0x4902 405C		McBSP3
	0x4902 605C		McBSP4
Description	McBSPLP receive channel enable register partition E		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERE															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERE	Receive Channel Enable RCERE n=0 Disables reception of n-th channel in an even-numbered block in partition E RCERE n=1 Enables reception of n-th channel in an even-numbered block in partition E	RW	0x0000

Table 18-87. Register Call Summary for Register MCBSPLP_RCERE_REG

McBSP Functional Description
- Using Eight Partitions: [0]

McBSP Register Manual
- McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-88. MCBSPLP_RCERF_REG

Address Offset	0x0000 0060		
Physical Address	0x4807 4060 0x4809 6060 0x4902 2060 0x4902 4060 0x4902 6060	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6060		McBSP5
	0x4902 2060		McBSP2
	0x4902 4060		McBSP3
	0x4902 6060		McBSP4
Description	McBSPLP receive channel enable register partition F		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERF															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERF	Receive Channel Enable RCERF n=0 Disables reception of n-th channel in an even-numbered block in partition F RCERF n=1 Enables reception of n-th channel in an even-numbered block in partition F	RW	0x0000

Table 18-89. Register Call Summary for Register MCBSPLP_RCERF_REG

McBSP Functional Description • Using Eight Partitions: [0]
McBSP Register Manual • McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-90. MCBSPLP_XCERE_REG

Address Offset	0x0000 0064		
Physical Address	0x4807 4064 0x4809 6064 0x4902 2064 0x4902 4064 0x4902 6064 0x4809 6064 0x4902 2064 0x4902 4064 0x4902 6064	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4 McBSP5 McBSP2 McBSP3 McBSP4
Description	McBSPLP transmit channel enable register partition E		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERE															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERE	Transmit Channel Enable XCERE n=0 Disables transmission of n-th channel in an event-numbered block in partition E XCERE n=1 Enables transmission of n-th channel in an event-numbered block in partition E	RW	0x0000

Table 18-91. Register Call Summary for Register MCBSPLP_XCERE_REG

McBSP Functional Description • Using Eight Partitions: [0]
McBSP Register Manual • McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-92. MCBSPLP_XCERF_REG

Address Offset	0x0000 0068		
Physical Address	0x4807 4068 0x4809 6068 0x4902 2068 0x4902 4068 0x4902 6068	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6068		McBSP5
	0x4902 2068		McBSP2
	0x4902 4068		McBSP3
	0x4902 6068		McBSP4
Description	McBSPLP transmit channel enable register partition F		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERF															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERF	Transmit Channel Enable	RW	0x0000
		XCERF n=0 Disables transmission of n-th channel in an even-numbered block in partition F		
		XCERF n=1 Enables transmission of n-th channel in an even-numbered block in partition F		

Table 18-93. Register Call Summary for Register MCBSPLP_XCERF_REG

McBSP Functional Description
- Using Eight Partitions: [0]

McBSP Register Manual
- McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-94. MCBSPLP_RCERG_REG

Address Offset	0x0000 006C		
Physical Address	0x4807 406C 0x4809 606C 0x4902 206C 0x4902 406C 0x4902 606C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 606C		McBSP5
	0x4902 206C		McBSP2
	0x4902 406C		McBSP3
	0x4902 606C		McBSP4
Description	McBSPLP receive channel enable register partition G		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERG															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERG	Receive Channel Enable RCERG n=0 Disables reception of n-th channel in an even-numbered block in partition G RCERG n=1 Enables reception of n-th channel in an even-numbered block in partition G	RW	0x0000

Table 18-95. Register Call Summary for Register MCBSPLP_RCERG_REG

McBSP Functional Description • Using Eight Partitions: [0]
McBSP Register Manual • McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-96. MCBSPLP_RCERH_REG

Address Offset	0x0000 0070		
Physical Address	0x4807 4070 0x4809 6070 0x4902 2070 0x4902 4070 0x4902 6070	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6070		McBSP5
	0x4902 2070		McBSP2
	0x4902 4070		McBSP3
	0x4902 6070		McBSP4
Description	McBSPLP receive channel enable register partition H		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																RCERH															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	RCERH	Receive Channel Enable RCERH n=0 Disables reception of n-th channel in an even-numbered block in partition H RCERH n=1 Enables reception of n-th channel in an even-numbered block in partition H	RW	0x0000

Table 18-97. Register Call Summary for Register MCBSPLP_RCERH_REG

McBSP Functional Description • Using Eight Partitions: [0] • Receive Multichannel Selection Mode: [1] [2] • Transmit Multichannel Selection Modes: [3] [4]
McBSP Register Manual • McBSP Register Mapping Summary: [5] [6] [7] [8] [9]

Table 18-98. MCBSPLP_XCERG_REG

Address Offset	0x0000 0074		
Physical Address	0x4807 4074	**Instance**	McBSP1
	0x4809 6074		McBSP5
	0x4902 2074		McBSP2
	0x4902 4074		McBSP3
	0x4902 6074		McBSP4
	0x4809 6074		McBSP5
	0x4902 2074		McBSP2
	0x4902 4074		McBSP3
	0x4902 6074		McBSP4
Description	McBSPLP transmit channel enable register partition G		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERG															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERG	Transmit Channel Enable	RW	0x0000
		XCERG n=0 Disables transmission of n-th channel in an event-numbered block in partition G		
		XCERG n=1 Enables transmission of n-th channel in an event-numbered block in partition G		

Table 18-99. Register Call Summary for Register MCBSPLP_XCERG_REG

McBSP Functional Description

- Using Eight Partitions: [0]

McBSP Register Manual

- McBSP Register Mapping Summary: [1] [2] [3] [4] [5]

Table 18-100. MCBSPLP_XCERH_REG

Address Offset	0x0000 0078		
Physical Address	0x4807 4078	**Instance**	McBSP1
	0x4809 6078		McBSP5
	0x4902 2078		McBSP2
	0x4902 4078		McBSP3
	0x4902 6078		McBSP4
	0x4809 6078		McBSP5
	0x4902 2078		McBSP2
	0x4902 4078		McBSP3
	0x4902 6078		McBSP4
Description	McBSPLP transmit channel enable register partition H		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XCERH															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	XCERH	Transmit Channel Enable XCERH n=0 Disables transmission of n-th channel in an even-numbered block in partition H XCERH n=1 Enables transmission of n-th channel in an even-numbered block in partition H	RW	0x0000

Table 18-101. Register Call Summary for Register MCBSPLP_XCERH_REG

McBSP Functional Description • Using Eight Partitions: [0] • Transmit Multichannel Selection Modes: [1] [2] [3] [4] [5] [6]
McBSP Register Manual • McBSP Register Mapping Summary: [7] [8] [9] [10] [11]

Table 18-102. MCBSPLP_REV_REG

Address Offset	0x0000 007C		
Physical Address	0x4807 407C 0x4809 607C 0x4902 207C 0x4902 407C 0x4902 607C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 607C		McBSP5
	0x4902 207C		McBSP2
	0x4902 407C		McBSP3
	0x4902 607C		McBSP4
Description	MCBSPLP Revision number register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Read returns 0x0.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision examples: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 18-103. Register Call Summary for Register MCBSPLP_REV_REG

McBSP Register Manual • McBSP Register Mapping Summary: [0] [1] [2] [3] [4]

Table 18-104. MCBSPLP_RINTCLR_REG

Address Offset	0x0000 0080		
Physical Address	0x4807 4080 0x4809 6080 0x4902 2080 0x4902 4080 0x4902 6080	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6080		McBSP5
	0x4902 2080		McBSP2
	0x4902 4080		McBSP3
	0x4902 6080		McBSP4
Description	McBSPLP receive interrupt clear		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RINTCLR																															

Bits	Field Name	Description	Type	Reset
31:0	RINTCLR	Read from this register will clear the IRQ generated by receive end-of-frame indication or mcbsp1_fsr detection. Write to this register has no effect. (legacy)	RW	0x00000000

Table 18-105. Register Call Summary for Register MCBSPLP_RINTCLR_REG

McBSP Register Manual

- McBSP Register Mapping Summary: [0] [1] [2] [3] [4]

Table 18-106. MCBSPLP_XINTCLR_REG

Address Offset	0x0000 0084		
Physical Address	0x4807 4084 0x4809 6084 0x4902 2084 0x4902 4084 0x4902 6084	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6084		McBSP5
	0x4902 2084		McBSP2
	0x4902 4084		McBSP3
	0x4902 6084		McBSP4
Description	McBSPLP transmit interrupt clear (legacy)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XINTCLR																															

Bits	Field Name	Description	Type	Reset
31:0	XINTCLR	Read from this register will clear the IRQ generated by transmit end-of-frame indication or mcbspi_fsx detection (I=1:5). Write to this register has no effect. (legacy)	RW	0x00000000

Table 18-107. Register Call Summary for Register MCBSPLP_XINTCLR_REG

McBSP Register Manual

- McBSP Register Mapping Summary: [0] [1] [2] [3] [4]

Table 18-108. MCBSPLP_ROVFLCLR_REG

Address Offset	0x0000 0088		
Physical Address	0x4807 4088 0x4809 6088 0x4902 2088 0x4902 4088 0x4902 6088 0x4809 6088 0x4902 2088 0x4902 4088 0x4902 6088	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4 McBSP5 McBSP2 McBSP3 McBSP4
Description	McBSPLP receive overflow interrupt clear		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROVFLCLR																															

Bits	Field Name	Description	Type	Reset
31:0	ROVFLCLR	Read from this register will clear the IRQ generated by the receive overflow condition. Write to this register has no effect. ((legacy)	RW	0x00000000

Table 18-109. Register Call Summary for Register MCBSPLP_ROVFLCLR_REG

McBSP Register Manual

- McBSP Register Mapping Summary: [0] [1] [2] [3] [4]

Table 18-110. MCBSPLP_SYSCONFIG_REG

Address Offset	0x0000 008C		
Physical Address	0x4807 408C 0x4809 608C 0x4902 208C 0x4902 408C 0x4902 608C	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 608C		McBSP5
	0x4902 208C		McBSP2
	0x4902 408C		McBSP3
	0x4902 608C		McBSP4
Description	McBSPLP System Configuration register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																						CLOCKACTIVITY		RESERVED				SIDLEMODE		ENAWAKEUP	SOFTRESET	RESERVED

Bits	Field Name	Description	Type	Reset
31:10	RESERVED	Read returns 0x0.	R	0x000000
9:8	CLOCKACTIVITY	0x0: The McBSPi_ICLK clock can be switched off. The PRCM functional clock can be switched off. 0x1: The McBSPi_ICLK clock must be maintained during wakeup. The PRCM functional clock can be switched off. 0x2: The McBSPi_ICLK clock can be switched off. The PRCM functional clock must be maintained during wakeup. 0x3: The McBSPi_ICLK clock must be maintained during wakeup. The PRCM functional clock must be maintained during wakeup.	RW	0x0
7:5	RESERVED	Read returns 0x0.	R	0x0
4:3	SIDLEMODE	Slave interface power management, idle request / acknowledge control: 0x0: Force-idle. An idle request is acknowledged unconditionally. 0x1: No-idle. An idle request is never acknowledged. 0x2: Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module 0x3: Reserved	RW	0x0
2	ENAWAKEUP	Wake-up feature control: 0x0: Wake-up is disabled 0x1: Wake-up capability is enabled	RW	0x0
1	SOFTRESET	McBSP global software reset 0x0: NO soft reset 0x1: Soft reset triggered	RW	0x0
0	RESERVED	Read returns 0x0.	R	0x0

Table 18-111. Register Call Summary for Register MCBSPLP_SYSCONFIG_REG

McBSP Integration • Hardware and Software Reset: [0] • McBSP Acknowledgment Modes: [1] [2] [3] [4] • Wake-Up Capability: [5] • Analysis of the Receiver Smart Idle Behavior: [6]
McBSP Basic Programming Model • McBSP Initialization Procedure: [7]
McBSP Register Manual • McBSP Register Mapping Summary: [8] [9] [10] [11] [12]

Table 18-112. MCBSPLP_THRSH2_REG

Address Offset	0x0000 0090		
Physical Address	0x4807 4090 0x4809 6090 0x4902 2090 0x4902 4090 0x4902 6090	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 6090		McBSP5
	0x4902 2090		McBSP2
	0x4902 4090		McBSP3
	0x4902 6090		McBSP4
Description	McBSPLP transmit buffer threshold (DMA or IRQ trigger)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								XTHRESHOLD							

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Read returns 0x0.	R	0x000000
10:0[(1)]	XTHRESHOLD	Transmit buffer threshold value. The DMA request (if enabled) of interrupt assertion (if enabled) is triggered if the number of free locations in the transmit buffer are above or equal to the XTHRESHOLD value + 1. Also, this value (XTHRESHOLD value + 1) indicates the number of words transferred during a transmit data DMA request, if transmit DMA is enabled.	RW	0x00

(1) XTHRESHOLD is an 11-bit field for McBSP2 only. For other McBSPs, XTHRESHOLD is an 8-bit field (bits 7 to 10 are reserved). In other words, the other McBSPs are limited to a FIFO width of 0x7F.

Table 18-113. Register Call Summary for Register MCBSPLP_THRSH2_REG

McBSP Integration • Wake-Up Capability: [0]
McBSP Functional Description • McBSP Transmission: [1] • McBSP DMA Configuration: [2]
McBSP Basic Programming Model • McBSP Initialization Procedure: [3] • Data Transfer DMA Request Configuration: [4] • Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [5]
McBSP Register Manual • McBSP Register Mapping Summary: [6] [7] [8] [9] [10]

Table 18-114. MCBSPLP_THRSH1_REG

Address Offset	0x0000 0094		
Physical Address	0x4807 4094	**Instance**	McBSP1
	0x4809 6094		McBSP5
	0x4902 2094		McBSP2
	0x4902 4094		McBSP3
	0x4902 6094		McBSP4
	0x4809 6094		McBSP5
	0x4902 2094		McBSP2
	0x4902 4094		McBSP3
	0x4902 6094		McBSP4
Description	McBSPLP receive buffer threshold (DMA or IRQ trigger)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					RTHRESHOLD										

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Read returns 0x0.	R	0x000000
10:0[(1)]	RTHRESHOLD	Receive buffer threshold value. The DMA request (if enabled) of interrupt assertion (if enabled)is triggered if the number of occupied locations in the receive buffer are above or equal to the RTHRESHOLD value + 1. Also, this value (RTHRESHOLD value + 1) indicates the number of words transferred during a receive data DMA request, if receive DMA is enabled.	RW	0x00

(1) RTHRESHOLD is an 11-bit field for McBSP2 only. For other McBSPs, RTHRESHOLD is an 8-bit field (bits 7 to 10 are reserved). In other words, the other McBSPs are limited to a FIFO width of 0x7F.

Table 18-115. Register Call Summary for Register MCBSPLP_THRSH1_REG

McBSP Integration
- Wake-Up Capability: [0]

McBSP Functional Description
- McBSP Reception: [1]
- McBSP DMA Configuration: [2]

McBSP Basic Programming Model
- Data Transfer DMA Request Configuration: [3]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [4]

McBSP Register Manual
- McBSP Register Mapping Summary: [7] [8] [9] [10] [11]

Table 18-116. MCBSPLP_IRQSTATUS_REG

Address Offset	0x0000 00A0		
Physical Address	0x4807 40A0	**Instance**	McBSP1
	0x4809 60A0		McBSP5
	0x4902 20A0		McBSP2
	0x4902 40A0		McBSP3
	0x4902 60A0		McBSP4
	0x4809 60A0		McBSP5
	0x4902 20A0		McBSP2
	0x4902 40A0		McBSP3
	0x4902 60A0		McBSP4
Description	McBSPLP Interrupt Status register (OCP compliant IRQ line)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																	XEMPTYEOF	RESERVED	XOVFLSTAT	XUNDFLSTAT	XRDY	XEOF	XFSX	XSYNCERR	RESERVED	ROVFLSTAT	RUNDFLSTAT	RRDY	REOF	RFSR	RSYNCERR

Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns 0x0.	R	0x00000
14	XEMPTYEOF	Transmit Buffer Empty at end of frame (XEMPTYEOF is set to one when a complete frame was transmitted and the transmit buffer is empty). Writing 1 to this bit clears the bit. 0x0: Transmit buffer NOT Empty at end of frame 0x1: Transmit buffer Empty at end of frame; Writing 1 to this bit clears the bit.	RW	0x0
13	RESERVED	Read returns 0x0.	R	0x0
12	XOVFLSTAT	Transmit Buffer Overflow (XOVFLSTAT bit is set to one when transmit buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit. 0x0: Transmit buffer NOT overflow 0x1: Transmit buffer overflow; Writing 1 to this bit clears the bit.	RW	0x0
11	XUNDFLSTAT	Transmit Buffer Underflow (XUNDFLSTAT bit is set to one when the transmit data buffer is empty new data is required to be transmitted). Writing 1 to this bit clears the bit. 0x0: the transmit data buffer is NOT empty new data is required to be transmitted. 0x1: the transmit data buffer is empty new data is required to be transmitted. Writing 1 to this bit clears the bit.	RW	0x0
10	XRDY	Transmit Buffer Threshold Reached (XRDY bit is set to one when the transmit buffer free locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit. 0x0: Transmit buffer occupied locations are below the THRSH2_REG value). 0x1: Transmit buffer occupied locations are equal or above the THRSH2_REG value). Writing 1 to this bit clears the bit.	RW	0x0
9	XEOF	Transmit End Of Frame (XEOF is set to one when a complete frame was transmitted). Writing 1 to this bit clears the bit. 0x0: complete frame was NOT transmitted 0x1: complete frame was transmitted; Writing 1 to this bit clears the bit.	RW	0x0
8	XFSX	Transmit Frame Synchronization (XFSX bit is set to one when a new transmit frame synchronization is asserted). Writing 1 to this bit clears the bit. 0x0: new transmit frame synchronization is NOT asserted 0x1: new transmit frame synchronization is asserted; Writing 1 to this bit clears the bit.	RW	0x0

Bits	Field Name	Description	Type	Reset
7	XSYNCERR	Transmit Frame Synchronization Error (XSYNCERR is set to one when a transmit frame synchronization error is detected). Writing 1 to this bit clears the bit. 0x0: Transmit frame synchronization error is NOT detected 0x1: Transmit frame synchronization error is detected. Writing 1 to this bit clears the bit.	RW	0x0
6	RESERVED	Read returns 0x0.	R	0x0
5	ROVFLSTAT	Receive Buffer Overflow (ROVFLSTAT bit is set to one when receive buffer overflow; the data which is written while overflow condition is discarded). Writing 1 to this bit clears the bit. 0x0: receive buffer NOT overflow 0x1: receive buffer overflow; Writing 1 to this bit clears the bit.	RW	0x0
4	RUNDFLSTAT	Receive Buffer Underflow (RUNDFLSTAT bit is set to one when read operation is performed to the receive data register while receive buffer is empty; data read while underflow condition is undefined). Writing 1 to this bit clears the bit. 0x0: read operation is performed to the receive data register while receive buffer is NOT empty 0x1: read operation is performed to the receive data register while receive buffer is empty; Writing 1 to this bit clears the bit.	RW	0x0
3	RRDY	Receive Buffer Threshold Reached (RRDY bit is set to one when the receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit. 0x0: receive buffer occupied locations are below the THRSH1_REG value). 0x1: receive buffer occupied locations are equal or above the THRSH1_REG value). Writing 1 to this bit clears the bit.	RW	0x0
2	REOF	Receive End Of Frame (REOF is set to one when a complete frame was received). Writing 1 to this bit clears the bit. 0x0: complete frame was NOT received 0x1: complete frame was received; Writing 1 to this bit clears the bit.	RW	0x0
1	RFSR	Receive Frame Synchronization (RFSR bit is set to one when a new receive frame synchronization is asserted). Writing 1 to this bit clears the bit. 0x0: new receive frame synchronization is NOT asserted 0x1: new receive frame synchronization is asserted; Writing 1 to this bit clears the bit.	RW	0x0
0	RSYNCERR	Receive Frame Synchronization Error (RSYNCERR is set to one when a receive frame synchronization error is detected). Writing 1 to this bit clears the bit. 0x0: receive frame synchronization error is NOT detected 0x1: receive frame synchronization error is detected. Writing 1 to this bit clears the bit.	RW	0x0

Table 18-117. Register Call Summary for Register MCBSPLP_IRQSTATUS_REG

McBSP Integration

- Wake-Up Capability: [0] [1] [2] [3]
- McBSP Interrupt Requests: [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17]

Table 18-117. Register Call Summary for Register MCBSPLP_IRQSTATUS_REG (continued)

McBSP Functional Description • Enable/Disable the Transmit and Receive Processes: [18] [19] [20] [21] • Introduction: [22] [23] [24] [25] [26] [27] [28] • Overrun in the Receiver: [29] [30] [31] • Possible Responses to Receive Frame-sync Pulses: [32] [33] • Underflow in the Receiver: [34] [35] • Underflow in the Transmitter: [36] [37] [38] • Possible Responses to Transmit Frame-sync Pulses: [39] • Overflow in the Transmitter: [40]
McBSP Basic Programming Model • L4-Compliant Interrupt Line: [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] • Legacy Interrupt Line: [56] [57]
McBSP Register Manual • McBSP Register Mapping Summary: [58] [59] [60] [61] [62]

Table 18-118. MCBSPLP_IRQENABLE_REG

Address Offset	0x0000 00A4		
Physical Address	0x4807 40A4 0x4809 60A4 0x4902 20A4 0x4902 40A4 0x4902 60A4	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60A4		McBSP5
	0x4902 20A4		McBSP2
	0x4902 40A4		McBSP3
	0x4902 60A4		McBSP4
Description	McBSPLP Interrupt Enable register (OCP compliant IRQ line)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XEMPTYEOFEN	RESERVED	XOVFLEN	XUNDFLEN	XRDYEN	XEOFEN	XFSXEN	XSYNCERREN	RESERVED	ROVFLEN	RUNDFLEN	RRDYEN	REOFEN	RFSREN	RSYNCERREN	

Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns 0x0.	R	0x00000
14	XEMPTYEOFEN	Transmit buffer empty at end of frame enable bit. 0x0: Transmit buffer Empty at end of frame NOT enabled 0x1: Transmit buffer Empty at end of frame enabled	RW	0x0
13	RESERVED	Read returns 0x0.	R	0x0
12	XOVFLEN	Transmit Buffer Overflow enable bit. 0x0: Transmit Buffer Overflow NOT enabled 0x1: Transmit Buffer Overflow enabled	RW	0x0
11	XUNDFLEN	Transmit Buffer Underflow enable bit. 0x0: Transmit Buffer Underflow NOT enabled 0x1: Transmit Buffer Underflow enabled	RW	0x0

Bits	Field Name	Description	Type	Reset
10	XRDYEN	Transmit Buffer Threshold Reached enable bit. 0x0: Transmit Buffer Threshold Reached NOT enabled 0x1: Transmit Buffer Threshold Reached enabled	RW	0x0
9	XEOFEN	Transmit End Of Frame enable bit. 0x0: Transmit End Of Frame NOT enabled 0x1: Transmit End Of Frame enabled	RW	0x0
8	XFSXEN	Transmit Frame Synchronization enable bit. 0x0: Transmit Frame Synchronization NOT enabled 0x1: Transmit Frame Synchronization enabled	RW	0x0
7	XSYNCERREN	Transmit Frame Synchronization Error enable bit. 0x0: Transmit Frame Synchronization Error NOT enabled 0x1: Transmit Frame Synchronization Error enabled	RW	0x0
6	RESERVED	Read returns 0x0.	R	0x0
5	ROVFLEN	Receive Buffer Overflow enable bit. 0x0: Receive Buffer Overflow NOT enabled 0x1: Receive Buffer Overflow enabled	RW	0x0
4	RUNDFLEN	Receive Buffer Underflow enable bit. 0x0: Receive Buffer Underflow NOT enabled 0x1: Receive Buffer Underflow enabled	RW	0x0
3	RRDYEN	Receive Buffer Threshold enable bit. 0x0: Receive Buffer Threshold NOT enabled 0x1: Receive Buffer Threshold enabled	RW	0x0
2	REOFEN	Receive End Of Frame enable bit. 0x0: Receive End Of Frame NOT enabled 0x1: Receive End Of Frame enabled	RW	0x0
1	RFSREN	Receive Frame Synchronization enable bit. RW 0x0: Receive Frame Synchronization NOT enabled 0x1: Receive Frame Synchronization enabled	RW	0x0
0	RSYNCERREN	Receive Frame Synchronization Error enable bit. 0x0: Receive Frame Synchronization Error NOT enabled 0x1: Receive Frame Synchronization Error enabled	RW	0x0

Table 18-119. Register Call Summary for Register MCBSPLP_IRQENABLE_REG

McBSP Integration
- Wake-Up Capability: [0] [1] [2] [3] [4] [5] [6] [7] [8]
- McBSP Interrupt Requests: [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22]

McBSP Functional Description
- Overrun in the Receiver: [23]
- Possible Responses to Receive Frame-sync Pulses: [24]
- Underflow in the Receiver: [25]
- Underflow in the Transmitter: [26]
- Possible Responses to Transmit Frame-sync Pulses: [27]
- Overflow in the Transmitter: [28]

McBSP Basic Programming Model
- Data Transfer DMA Request Configuration: [29] [30]
- L4-Compliant Interrupt Line: [31] [32]

McBSP Register Manual
- McBSP Register Mapping Summary: [33] [34] [35] [36] [37]

Table 18-120. MCBSPLP_WAKEUPEN_REG

Address Offset	0x0000 00A8		
Physical Address	0x4807 40A8 0x4809 60A8 0x4902 20A8 0x4902 40A8 0x4902 60A8	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60A8		McBSP5
	0x4902 20A8		McBSP2
	0x4902 40A8		McBSP3
	0x4902 60A8		McBSP4
Description	McBSPLP Wakeup Enable register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																XEMPTYEOFEN	RESERVED				XRDYEN	XEOFEN	XFSXEN	XSYNCERREN	RESERVED			RRDYEN	REOFEN	RFSREN	RSYNCERREN

Bits	Field Name	Description	Type	Reset
31:15	RESERVED	Read returns 0x0.	R	0x00000
14	XEMPTYEOFEN	Transmit buffer empty at end of frame WK enable bit. 0x0: Transmit buffer Empty at end of frame WK enable is NOT active 0x1: Transmit buffer Empty at end of frame WK enable is active	RW	0x0
13:11	RESERVED	Read returns 0x0.	R	0x0
10	XRDYEN	Transmit Buffer Threshold Reached WK enable bit. 0x0: Transmit Buffer Threshold WK enable is NOT active 0x1: Transmit Buffer Threshold WK enable is active	RW	0x0
9	XEOFEN	Transmit End Of Frame WK enable bit. 0x0: Transmit End Of Frame WK enable is NOT active 0x1: Transmit End Of Frame WK enable is active	RW	0x0
8	XFSXEN	Transmit Frame Synchronization WK enable bit. 0x0: Transmit Frame Synchronization WK enable is NOT active 0x1: Transmit Frame Synchronization WK enable is active	RW	0x0
7	XSYNCERREN	Transmit Frame Synchronization Error WK enable bit. 0x0: Transmit Frame Synchronization Error WK enable is NOT active 0x1: Transmit Frame Synchronization Error WK enable is active	RW	0x0
6:4	RESERVED	Read returns 0x0.	R	0x0
3	RRDYEN	Receive Buffer Threshold wake-up enable bit. 0x0: Receive Buffer Threshold WK enable is NOT active 0x1: Receive Buffer Threshold WK enable is active	RW	0x0
2	REOFEN	Receive End Of Frame WK enable bit. 0x0: Receive End Of Frame WK enable is NOT active 0x1: Receive End Of Frame WK enable is active	RW	0x0

Bits	Field Name	Description	Type	Reset
1	RFSREN	Receive Frame Synchronization WK enable bit. 0x0: Receive Frame Synchronization WK enable is NOT active 0x1: Receive Frame Synchronization WK enable is active	RW	0x0
0	RSYNCERREN	Receive Frame Synchronization Error WK enable bit. 0x0: Receive Frame Synchronization Error WK enable is NOT active 0x1: Receive Frame Synchronization Error WK enable is active	RW	0x0

Table 18-121. Register Call Summary for Register MCBSPLP_WAKEUPEN_REG

McBSP Integration
- Wake-Up Capability: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

McBSP Register Manual
- McBSP Register Mapping Summary: [10] [11] [12] [13] [14]

Table 18-122. MCBSPLP_XCCR_REG

Address Offset	0x0000 00AC		
Physical Address	0x4807 40AC 0x4809 60AC 0x4902 20AC 0x4902 40AC 0x4902 60AC	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60AC		McBSP5
	0x4902 20AC		McBSP2
	0x4902 40AC		McBSP3
	0x4902 60AC		McBSP4
Description	McBSPLP transmit configuration control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EXTCLKGATE	PPCONNECT	DXENDLY		XFULL_CYCLE	RESERVED			DLB	RESERVED	XDMAEN	RESERVED				XDISABLE

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15	EXTCLKGATE	External clock gating enable (CLKX and FSX master only). When this bit is set and the transmit clock and FSX are set as output, the CLKX is enabled when FSX is active plus 3 clock cycles after (clock is provided for FWID + 4 clock cycles, assuming that the FSX width, active, is FWID + 1 clock cycles); outside this window the external transmit clock is gated. The receive use the same gated transmit clock and transmit frame synchronization signals regardless of the CLKRM/FSRM settings. When using this mode the frame synchronization signal must be active during reception of the entire frame (FWID must be programmed accordingly) to ensure the proper receive process, which requires at least 3 cycles after the frame complete to transfer the data into the receive buffer. 0x0: External clock gating disabled. 0x1: External clock gating enable.	RW	0x0
14	PPCONNECT	Pair to pair connection. When set the DXENO pin is always set to 0, regardless of the frame boundary, setting the tree state buffer as output. 0x0: non Pair-to-pair connection. The DX pin will go to high-impedance state when there is no frame to transmit. 0x1: Pair-to-pair connection. When set, the DXENO pin is always set to 0, regardless of the frame boundary, setting the tree state buffer as output. This means the DX pin will be driven outside valid frame window. In that case, data sent by McBSP module during inactive channel are not guaranteed.	RW	0x0
13:12	DXENDLY	When McBSPi.MCBSPLP_SPCR1_REG[7] DXENA bit is set to one, this field selects the added delay as follow: 0x0: 13 ns 0x1: 18 ns (default) 0x2: 24 ns 0x3: 30 ns	RW	0x1
11	XFULL_CYCLE	Transmit full cycle mode select: 0x0: McBSP module operates in transmit half-cycle mode (transmit frame synchronization is sampled by the opposite edge of the clock used to drive transmit data) 0x1: McBSP module operates in transmit full-cycle mode (transmit frame synchronization is sampled by the same edge of the clock used to drive transmit data)	RW	0x0
10:6	RESERVED	Read returns 0x0.	R	0x00
5	DLB	Digital Loop-Back 0x0: No DLB 0x1: DLB	RW	0x0
4	RESERVED	Read returns 0x0.	R	0x0
3	XDMAEN	Transmit DMA Enable bit. When set to zero this bit will gate the external transmit DMA request, without resetting the DMA state machine. It is recommended to change this bit value only during transmit reset. 0x0: When set to zero this bit will gate the external transmit DMA request, 0x1: When set to one this bit will NOT gate the external transmit DMA request,	RW	0x1
2:1	RESERVED	Read returns 0x0.	R	0x0

Bits	Field Name	Description	Type	Reset
0	XDISABLE	Transmit Disable bit. When this bit is set the transmit process will stop at the next frame boundary. 0x0: The transmit process will NOT stop at the next frame boundary. 0x1: The transmit process will stop at the next frame boundary.	RW	0x0

Table 18-123. Register Call Summary for Register MCBSPLP_XCCR_REG

McBSP Functional Description
- Clocking and Framing Data: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]
- Clocking: [14] [15] [16] [17]
- Enable/Disable the Transmit and Receive Processes: [18] [19] [20]
- Transmit Full Cycle Mode: [21]
- Transmit Half Cycle Mode: [22]
- Clock Generation in the SRG: [23]
- McBSP DMA Configuration: [24]

McBSP Basic Programming Model
- McBSP Core: [25]
- McBSP Initialization Procedure: [26]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [27] [28] [29]
- Programming the McBSP Registers for the Desired Transmitter Operation (Step 2): [30] [31]

McBSP Register Manual
- McBSP Register Mapping Summary: [32] [33] [34] [35] [36]
- McBSP Register Description: [37] [38]

Table 18-124. MCBSPLP_RCCR_REG

Address Offset	0x0000 00B0		
Physical Address	0x4807 40B0	**Instance**	McBSP1
	0x4809 60B0		McBSP5
	0x4902 20B0		McBSP2
	0x4902 40B0		McBSP3
	0x4902 60B0		McBSP4
	0x4809 60B0		McBSP5
	0x4902 20B0		McBSP2
	0x4902 40B0		McBSP3
	0x4902 60B0		McBSP4
Description	McBSPLP receive configuration control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				RFULL_CYCLE	RESERVED							RDMAEN	RESERVED		RDISABLE

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Read returns 0x0.	R	0x0000000
11	RFULL_CYCLE	Receive full cycle mode select: 0x0: McBSP module operates in receive half-cycle mode (receive frame synchronization is sampled by the opposite edge of the clock used to sample receive data) 0x1: McBSP module operates in receive full-cycle mode (receive frame synchronization is sampled by the same edge of the clock used to sample receive data)	RW	0x1
10:4	RESERVED	Read returns 0x0.	R	0x0000000
3	RDMAEN	Receive DMA Enable bit. When set to zero this bit will gate the external transmit DMA request, without resetting the DMA state machine. It is recommended to change this bit value only during receive reset. 0x0: When set to zero this bit will gate the external transmit DMA request 0x1: When set to one this bit will NOT gate the external transmit DMA request	RW	0x1
2:1	RESERVED	Read returns 0x0.	R	0x0
0	RDISABLE	Receive Disable bit. When this bit is set the receive process will stop at the next frame boundary. 0x0: the receive process will NOT stop at the next frame boundary. 0x1: When this bit is set the receive process will stop at the next frame boundary.	RW	0x0

Table 18-125. Register Call Summary for Register MCBSPLP_RCCR_REG

McBSP Functional Description
- Enable/Disable the Transmit and Receive Processes: [0] [1]
- Receive Full Cycle Mode: [2]
- Receive Half Cycle Mode: [3]
- McBSP DMA Configuration: [4]

McBSP Basic Programming Model
- McBSP Core: [5]
- Programming the McBSP Registers for the Desired Receiver Configuration (Step 2): [6]

McBSP Register Manual
- McBSP Register Mapping Summary: [7] [8] [9] [10] [11]

Table 18-126. MCBSPLP_XBUFFSTAT_REG

Address Offset	0x0000 00B4		
Physical Address	0x4807 40B4 0x4809 60B4 0x4902 20B4 0x4902 40B4 0x4902 60B4	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60B4		McBSP5
	0x4902 20B4		McBSP2
	0x4902 40B4		McBSP3
	0x4902 60B4		McBSP4
Description	McBSPLP transmit buffer status		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					XBUFFSTAT										

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Read returns 0x0.	R	0x000000
10:0[1]	XBUFFSTAT	Transmit Buffer Status (indicates the number of free locations in the transmit buffer). The XBUFFSTAT value reflects the buffer status on the L4 clock domain and it can be bigger than the real number of the free locations which are seen by the transmit state machine.	R	0x500[2]

(1) XBUFFSTAT is an 11-bit field for McBSP2 only. For other McBSPs, XBUFFSTAT is an 8-bit field (bits 8 to 10 are reserved).
(2) The reset value of XBUFFSTAT for other McBSPs is 0x080.

Table 18-127. Register Call Summary for Register MCBSPLP_XBUFFSTAT_REG

McBSP Functional Description
- Enable/Disable the Transmit and Receive Processes: [0]
- McBSP DMA Configuration: [1]

McBSP Register Manual
- McBSP Register Mapping Summary: [2] [3] [4] [5] [6]

Table 18-128. MCBSPLP_RBUFFSTAT_REG

Address Offset	0x0000 00B8		
Physical Address	0x4807 40B8 0x4809 60B8 0x4902 20B8 0x4902 40B8 0x4902 60B8	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60B8		McBSP5
	0x4902 20B8		McBSP2
	0x4902 40B8		McBSP3
	0x4902 60B8		McBSP4
Description	McBSPLP receive buffer status		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					RBUFFSTAT										

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Read returns 0x0.	R	0x000000
10:0[1]	RBUFFSTAT	Receive Buffer Status (indicates the number of occupied locations in the receive buffer). The RBUFFSTAT value reflects the buffer status on the L4 clock domain and it can be smaller than the real number of the occupied locations which are seen by the receive state machine.	R	0x00

(1) RBUFFSTAT is an 11-bit field in McBSP2. For other McBSPs, RBUFFSTAT is an 8-bit field (bits 8 to 10 are reserved).

Table 18-129. Register Call Summary for Register MCBSPLP_RBUFFSTAT_REG

McBSP Functional Description
- Enable/Disable the Transmit and Receive Processes: [0]
- McBSP DMA Configuration: [1]

McBSP Register Manual
- McBSP Register Mapping Summary: [2] [3] [4] [5] [6]

Table 18-130. MCBSPLP_SSELCR_REG

Address Offset	0x0000 00BC		
Physical Address	0x4807 40BC 0x4809 60BC 0x4902 20BC 0x4902 40BC 0x4902 60BC	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60BC		McBSP5
	0x4902 20BC		McBSP2
	0x4902 40BC		McBSP3
	0x4902 60BC		McBSP4
Description	McBSPLP sidetone select register.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					SIDETONEEN	OCH1ASSIGN		OCH0ASSIGN		ICH1ASSIGN		ICH0ASSIGN			

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Read returns 0x0.	R	0x000000
10	SIDETONEEN	Sidetone mode enable. 0x0: Sidetone disabled. 0x1: Sidetone enabled.	RW	0x0
9:7	OCH1ASSIGN	Map the data for the speaker out channels to one of the McBSP channels (1 out of 8 channels)	RW	0x1
6:4	OCH0ASSIGN	Map the data for the speaker out channels to one of the McBSP channels (1 out of 8 channels)	RW	0x0
3:2	ICH1ASSIGN	Map the data from digital microphone channels to one of the McBSP channels (1 out of 4 channels)	RW	0x1
1:0	ICH0ASSIGN	Map the data from digital microphone channels to one of the McBSP channels (1out of 4 channels)	RW	0x0

Table 18-131. Register Call Summary for Register MCBSPLP_SSELCR_REG

McBSP Functional Description
- SIDETONE Interface: [0] [1] [2] [3] [4] [5] [6]

McBSP Basic Programming Model
- SIDETONE Activation Procedure: [7] [8] [9] [10] [11]

McBSP Register Manual
- McBSP Register Mapping Summary: [12] [13] [14] [15] [16]

Table 18-132. MCBSPLP_STATUS_REG

Address Offset	0x0000 00C0		
Physical Address	0x4807 40C0 0x4809 60C0 0x4902 20C0 0x4902 40C0 0x4902 60C0	**Instance**	McBSP1 McBSP5 McBSP2 McBSP3 McBSP4
	0x4809 60C0		McBSP5
	0x4902 20C0		McBSP2
	0x4902 40C0		McBSP3
	0x4902 60C0		McBSP4
Description	McBSPLP status register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															CLKMUXSTATUS

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Read returns 0x0.	R	0x00000000
0	CLKMUXSTATUS	When going to/exiting from idle mode, the clock for the interface domain is switched between McBSP_ICLK and master serial clock to allow functioning in idle mode. This bit indicates that the status of clock switching and accesses to the McBSP registers are delayed during clock switching. To avoid such a situation, polling can be performed to the status register to evaluate when McBSPLP is ready. This information is relevant only for the McBSPLP having operating in slave mode (serial clock provided by external component). 0: McBSP registers can be accessed. 1: The response to a different register access is delayed until the muxing process is done. Only the MCBSPLP_STATUS_REG[CLKMUXSTATUS] register can be accessed under this condition. The McBSP cannot exit from IDLE state (the external clock must be restarted).	R	0x0

Table 18-133. Register Call Summary for Register MCBSPLP_STATUS_REG

McBSP Register Manual
- McBSP Register Mapping Summary: [0] [1] [2] [3] [4]
- McBSP Register Description: [5]

18.6.4 SIDETONE Register Description

Table 18-134. ST_REV_REG

Address Offset	0x0000 0000		
Physical Address	0x4902 8000 0x4902 A000	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A000		SIDETONE_McBSP3
Description	SIDETONE Revision number register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Read returns 0x0.	R	0x000000
7:0	REV	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 18-135. Register Call Summary for Register ST_REV_REG

McBSP Register Manual
- SIDETONE Register Mapping Summary: [0] [1]

Table 18-136. ST_SYSCONFIG_REG

Address Offset	0x0000 0010		
Physical Address	0x4902 8010 0x4902 A010	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A010		SIDETONE_McBSP3
Description	SIDETONE System Configuration register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Read returns 0x0.	R	0x00000000
0	AUTOIDLE	Automatic McBSPi_ICLK clock gating 0x0: McBSPi_ICLK clock auto-gating disabled. 0x1: McBSPi_ICLK clock auto-gating enabled.	RW	0x1

Table 18-137. Register Call Summary for Register ST_SYSCONFIG_REG

McBSP Integration
- SIDETONE Clock: [0] [1]

McBSP Register Manual
- SIDETONE Register Mapping Summary: [2] [3]

Table 18-138. ST_IRQSTATUS_REG

Address Offset	0x0000 0018		
Physical Address	0x4902 8018 0x4902 A018	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A018		SIDETONE_McBSP3
Description	SIDETONE Interrupt Status Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															OVRRERROR

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Read returns 0x0.	R	0x00000000
0	OVRRERROR	Over-run error has occurred. New data to be processed has arrived before the previous one has ended. Writing 1 to this bit clears the bit.	RW	0x0

Table 18-139. Register Call Summary for Register ST_IRQSTATUS_REG

McBSP Integration
- SIDETONE_McBSP Interrupt Requests: [0] [1] [2] [3]

McBSP Functional Description
- Interrupt Operation: [4]

McBSP Register Manual
- SIDETONE Register Mapping Summary: [5] [6]

Table 18-140. ST_IRQENABLE_REG

Address Offset	0x0000 001C		
Physical Address	0x4902 801C 0x4902 A01C	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A01C		SIDETONE_McBSP3
Description	SIDETONE Interrupt enable register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															OVRRERROREN

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Read returns 0x0.	R	0x00000000
0	OVRRERROREN	Over-run error interrupt enable bit.	RW	0x0

Table 18-141. Register Call Summary for Register ST_IRQENABLE_REG

McBSP Integration • SIDETONE_McBSP Interrupt Requests: [0] [1] [2]
McBSP Functional Description • Interrupt Operation: [3]
McBSP Register Manual • SIDETONE Register Mapping Summary: [4] [5]

Table 18-142. ST_SGAINCR_REG

Address Offset	0x0000 0024		
Physical Address	0x4902 8024 0x4902 A024	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A024		SIDETONE_McBSP3
Description	Sidetone gain control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1GAIN																CH0GAIN															

Bits	Field Name	Description	Type	Reset
31:16	CH1GAIN	Second sidetone channel gain	RW	0x0000
15:0	CH0GAIN	First sidetone channel gain	RW	0x0000

Table 18-143. Register Call Summary for Register ST_SGAINCR_REG

McBSP Functional Description • Applying Gain: [0]
McBSP Basic Programming Model • SIDETONE Initialization Procedure: [1] [2]
McBSP Register Manual • SIDETONE Register Mapping Summary: [3] [4]

Table 18-144. ST_SFIRCR_REG

Address Offset	0x0000 0028		
Physical Address	0x4902 8028 0x4902 A028	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A028		SIDETONE_McBSP3
Description	Sidetone FIR coefficients control register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																FIRCOEFF															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Read returns 0x0.	R	0x0000
15:0	FIRCOEFF	FIR coefficients control register (the coefficients are programmed by successive write sequence of all 128 FIR coefficients) The write sequence should start with the coefficient 0. In order to enable the write to this register the COEFFWREN bit in SSELCR_REG should be set to one. When this bit is set the read operation will return only the last written value. After a complete FIR coefficients write sequence the COEFFWREN should be set to zero. A read sequence from SFIRCR_REG while COEFFWREN is set to zero will return the coefficients values starting from 0 to 127. The write coefficient address is set to 0 by the change of COEFFWREN from 0 to 1. The read coefficient address is set to 0 by the change of COEFFWREN from 1 to 0	RW	0x0000

Table 18-145. Register Call Summary for Register ST_SFIRCR_REG

McBSP Basic Programming Model
- SIDETONE Initialization Procedure: [0] [1]
- SIDETONE FIR Coefficients Writing: [2]
- SIDETONE FIR Coefficients Reading: [3]

McBSP Register Manual
- SIDETONE Register Mapping Summary: [4] [5]

Table 18-146. ST_SSELCR_REG

Address Offset	0x0000 002C		
Physical Address	0x4902 802C 0x4902 A02C	**Instance**	SIDETONE_McBSP2 SIDETONE_McBSP3
	0x4902 A02C		SIDETONE_McBSP3
Description	Sidetone select register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													COEFFWRDONE	COEFFWREN	SIDETONEEN

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Read returns 0x0.	R	0x00000000
2	COEFFWRDONE	Write FIR coefficients completed. 0x0: FIR coefficients not loaded 0x1: FIR coefficients loaded	R	0x0

Bits	Field Name	Description	Type	Reset
1	COEFFWREN	Write enable FIR coefficients. 0x1: If a 0 to 1 transition on this bit occurs, the write coefficient index is reset. When this bit is 1, all coefficients can be written in SFIRCR_REG performing 128 write accesses with SIDETONEEN 1. First access writes coefficient index 0 Read access in this case returns the last written value. 0x0: If a 1 to 0 transition on this bit occurs, the read coefficient index is reset. When this bit is 0, all coefficients can be read from SFIRCR_REG by performing 128 read accesses with SIDETONEEN 0. First access reads coefficient index 0.	RW	0x0
0	SIDETONEEN	Sidetone mode enable. 0x0: Sidetone disabled 0x1: Sidetone enabled	RW	0x0

Table 18-147. Register Call Summary for Register ST_SSELCR_REG

McBSP Functional Description • Data Processing Path: [0] • Enabling SIDETONE: [1] [2] [3]
McBSP Basic Programming Model • SIDETONE Activation Procedure: [4] • SIDETONE Initialization Procedure: [5] [6] • SIDETONE FIR Coefficients Writing: [7] [8] • SIDETONE FIR Coefficients Reading: [9]
McBSP Register Manual • SIDETONE Register Mapping Summary: [10] [11]

MMC/SD/SDIO Card Interface

This chapter describes the features and functions of the multimedia card/SD/SD I/O (MMC/SDIO) card interface.

Topic **Page**

19.1 MMC/SD/SDIO Overview

The multimedia card high-speed/SD/SD I/O (MMC/SD/SDIO) host controller provides an interface between a local host (LH) such as a microprocessor unit (MPU) or digital signal processor (DSP) and either MMC, SD memory cards, or SDIO cards and handles MMC/SD/SDIO transactions with minimal LH intervention.

The application interface manages transaction semantics. The MMC/SD/SDIO host controller deals with MMC/SD/SDIO protocol at transmission level, data packing, adding cyclic redundancy checks (CRC), start/end bit, and checking for syntactical correctness.

The application interface can send every MMC/SD/SDIO command and either poll for the status of the adapter or wait for an interrupt request, which is sent back in case of exceptions or to warn of end of operation.

The application interface can read card responses or flag registers. It can also mask individual interrupt sources. All these operations can be performed by reading and writing control registers. The MMC/SD/SDIO host controller also supports two DMA channels.

There are three MMC/SD/SDIO host controllers inside the device: Figure 19-1 gives an overview of the MMC/SD/SDIO controller instances 1 and 3, and Figure 19-2 gives an overview of the MMC/SD/SDIO2 controller instance.

Figure 19-1. MMC/SD/SDIO1 and 3 Overview

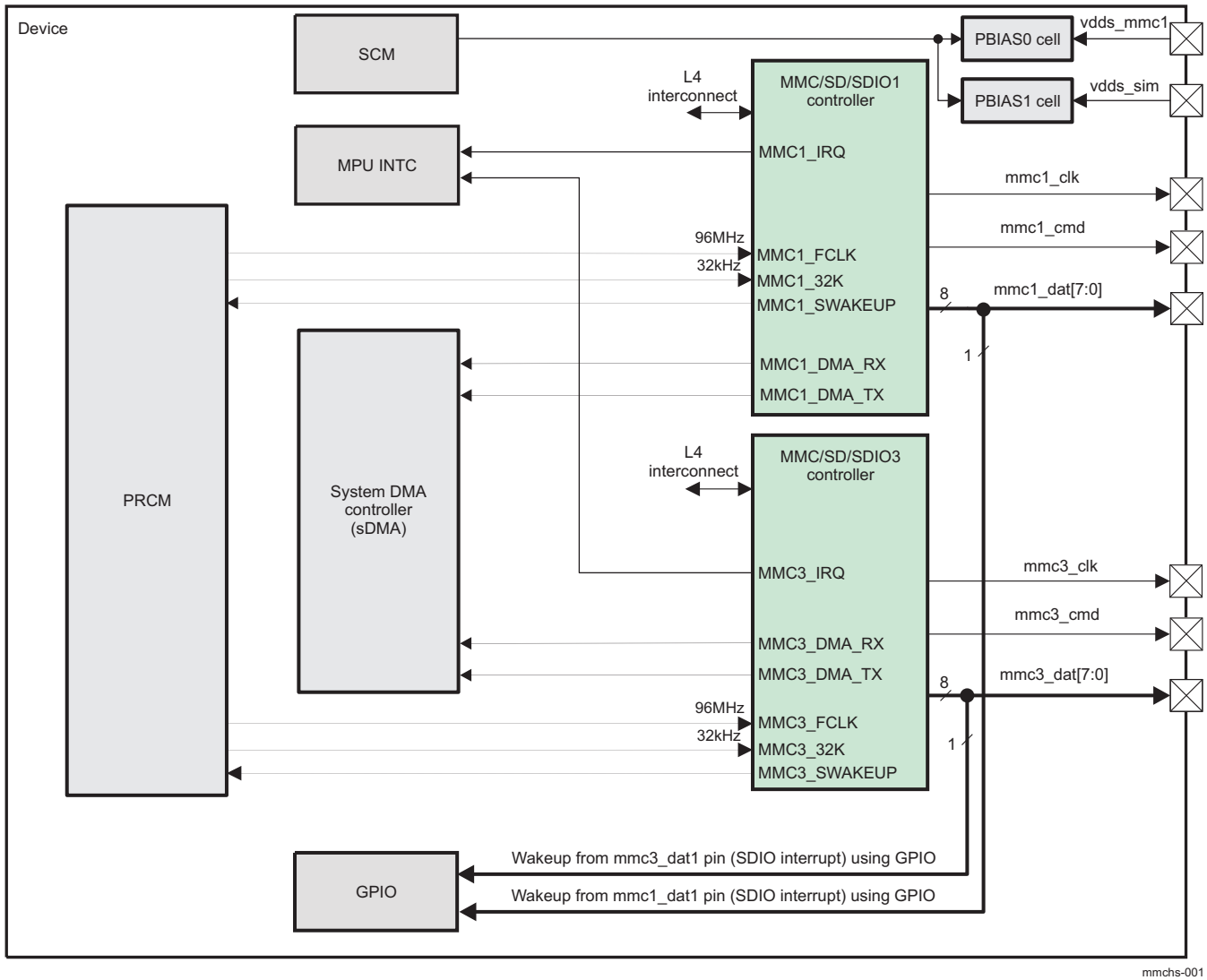
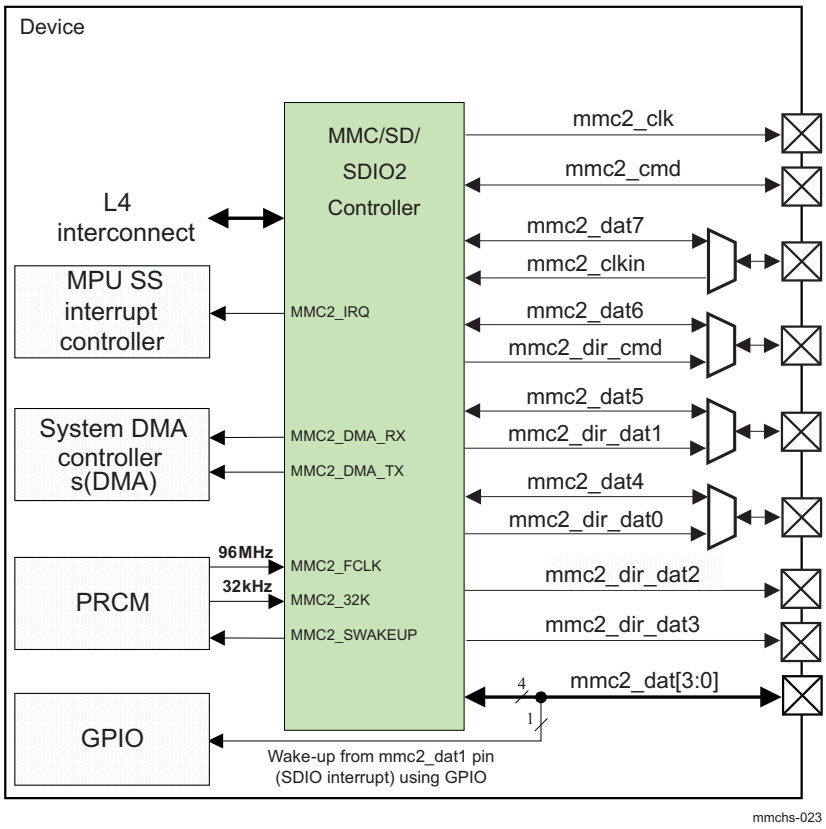


Figure 19-2. MMC/SD/SDIO2 Overview



19.1.1 MMC/SD/SDIO Features

The main features of the MMC/SD/SDIO host controller are:

- Full compliance with MMC command/response sets as defined in the *Multimedia Card System Specification,* v4.2 including high-capacity (size > 2GB) cards HC MMC.
- Full compliance with SD command/response sets as defined in the *SD Memory Card Specifications*, v2.0 including high-capacity SDHC cards up to 32GB.
- Full compliance with SDIO command/response sets and interrupt/read-wait mode as defined in the *SDIO Card Specification, Part E1*, v2.00
- Full compliance with sets as defined in the *SD Card Specification, Part A2, SD Host Controller Standard Specification*, v2.00
- Full compliance with MMC bus testing procedure as defined in the *Multimedia Card System Specification*, v4.2
- Full compliance with CE-ATA command/response sets as defined in the *CE-ATA Standard Specification*
- Full compliance with ATA for MMCA specification
- Flexible architecture allowing support for new command structure
- Support:
  - 1-bit or 4-bit transfer mode specifications for SD and SDIO cards
  - 1-bit, 4-bit, or 8-bit transfer mode specifications for MMC cards
- Built-in 1024-byte buffer for read or write
- 32-bit-wide access bus to maximize bus throughput
- Single interrupt line for multiple interrupt source events
- Two slave DMA channels (1 for TX, 1 for RX)

- Designed for low power
- Programmable clock generation
- Support SDIO Read Wait and Suspend/Resume functions
- Support Stop at block gap
- Support command completion signal (CCS) and command completion signal disable (CCSD) management as specified in the *CE-ATA Standard Specification*

The known limitations are as follows:

- No built-in hardware support for error correction codes (ECC). See the *Multimedia Card System Specification*, v4.2, and the *SD Memory Card Specifications*, v2.0, for details about ECC.
- The maximum block size defined in the *SD Memory Card Specifications*, v2.0, that the host driver can read and write to the buffer in the host controller is 2048 bytes. MMC supports a maximum block size of 1024 bytes. Up to 512 byte transfers, the buffer in MMC is considered as a double buffering with ping-pong management; half of the buffer can be written while the other part is read. For 512 to 1024 byte transfers, the entire buffer is dedicated to the transfer (read only or write only).

NOTE: MMC/e.MMC compliancy versus the *Multimedia Card System Specification*, v4.3 (or v4.4):

Full compliance with the MMC command/response sets and MMC bus testing procedure, as defined in the *Multimedia Card System Specification*, v4.2, does not prevent the use of e.MMC version 4.3 (or 4.4) devices with the OMAP34xx, because e.MMC devices are backward compatible. Only the e.MMC version 4.3 (or 4.4) boot operation is not supported in the OMAP34xx. Boot operation in the OMAP34xx follows the TI booting scheme, ID booting from standard memory array.

The differences between the MMC/SD/SDIO host controllers and a standard SD host controller are defined by the *SD Card Specification, Part A2, SD Host Controller Standard Specification*, v1.00, as follows:

- The MMC/SD/SDIO host controllers support MMC cards.
- The MMC/SD/SDIO host controller is defined as a DMA slave device. A standard SD host controller is defined as a DMA master controller that can start and stop a DMA transfer. MMC/SD/SDIO host controllers support DMA transfers through slave DMA requests.
- The clock divider in MMC/SD/SDIO host controller supports a wider range of frequency than specified in the *SD Memory Card Specifications*, v2.0. The MMC/SD/SDIO host controller supports odd and even clock ratio.
- The MMC/SD/SDIO host controller supports configurable busy time-out.

19.2 MMC/SD/SDIO Environment

One MMC/SD/SDIO host controller can support one MMC memory card, one SD memory card, or one SDIO card.

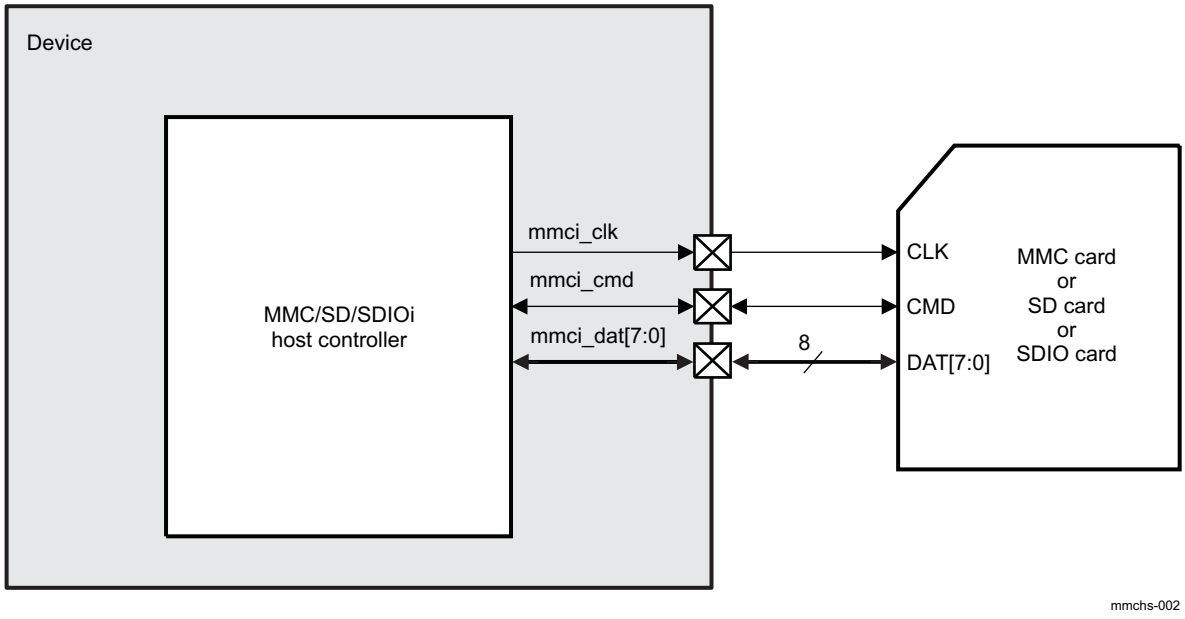
Other combinations (for example, two SD cards, one MMC card, and one SD card) are not supported through a single controller.

- The first controller (MMC/SD/SDIO1) integrates an internal transceiver that allows a direct connection to the MMC/SD/SDIO card (1.8 and 3V), without external transceiver.
- The second controller (MMC/SD/SDIO2) allows connecting MMC/SD/SDIO cards (only 1.8V cards) or an external device that uses the MMC/SD/SDIO interface (WLAN device for example). The second instance also supports an external transceiver and provides direction signals for data and command. Using an external transceiver device precludes 8-bit transfer mode.
- The third controller (MMC/SD/SDIO3) allows connecting MMC/SD/SDIO cards (only 1.8V cards) or an external device that uses the MMC/SD/SDIO interface (Wireless USB card for example). This interface is used without external transceiver.

19.2.1 MMC/SD/SDIO Connected to an MMC, an SD, or an SDIO Card

Figure 19-3 shows MMC/SD/SDIOi host controller, instance 1, 2 or 3, connected to an MMC, an SD, or an SDIO card and its related external connections.

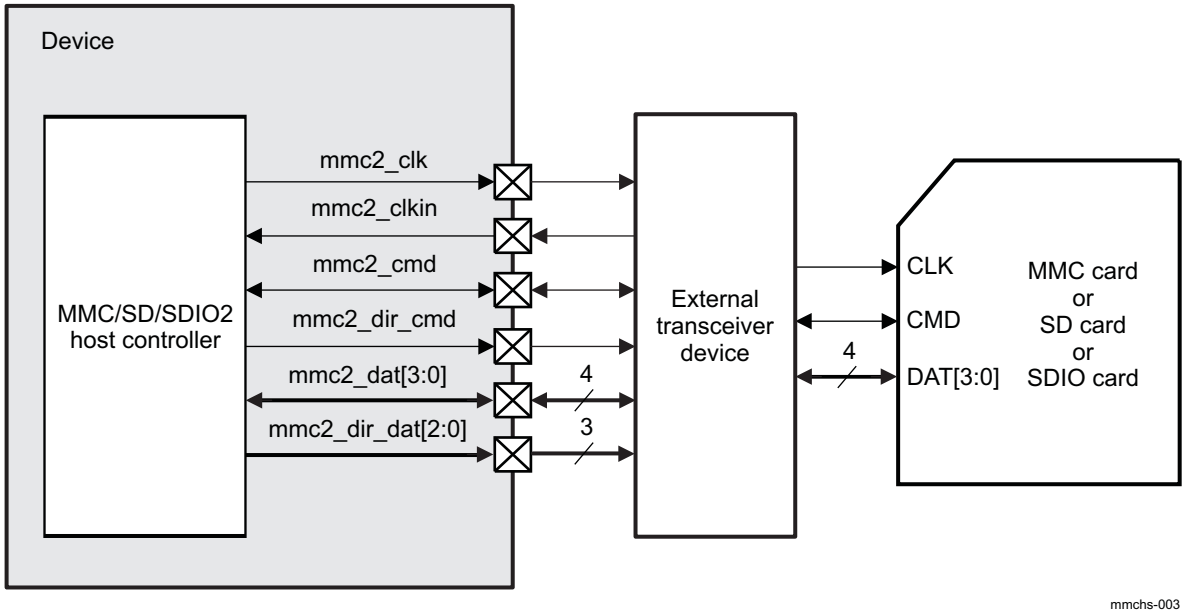
Figure 19-3. MMC/SD/SDIO Connected to an MMC, an SD, or an SDIO Card Without External Transceiver



19.2.2 MMC/SD/SDIO Connected to an MMC, an SD, or an SDIO Card Through an External Transceiver Device

This connection is supported only by the MMC/SD/SDIO2 host controller. Figure 19-4 shows the MMC/SD/SDIO2 host controller connected to an MMC, an SD, or an SDIO card through an external transceiver device.

Figure 19-4. MMC/SD/SDIO2 Connected to an MMC, an SD, or an SDIO Card with External Transceiver



19.2.3 MMC/SD/SDIO Functional Interfaces

19.2.3.1 Basic MMC/SD/SDIOi Pins Without External Transceiver

Figure 19-5 shows the MMC/SD/SDIOi host controller interface signals (instance 1, 2 or 3).

Figure 19-5. MMC/SD/SDIOi Interface Signals

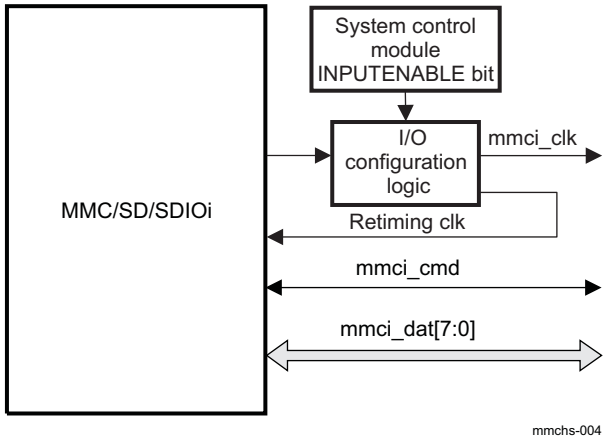


Table 19-1 describes the MMC/SD/SDIOi inputs/outputs.

Table 19-1. MMC/SD/SDIOi I/O Description

Signal Name	I/O[1]	Description	Reset Value
mmci_clk	O	External clock for MMC/SD/SDIO card[2]	0
mmci_cmd	I/O	Command signal	0
mmci_dat[7:0]	I/O	Data signals	0

(1) I = Input; O = Output

(2) This output signal is also used as retiming input (the CONTROL_PADCONF_x.INPUTENABLE bit must be set to 1).

19.2.3.2 Basic MMC/SD/SDIO2 Pins with External Transceiver

Figure 19-6 shows the MMC/SD/SDIO2 host controller interface signals.

Figure 19-6. MMC/SD/SDIO2 Interface Signals

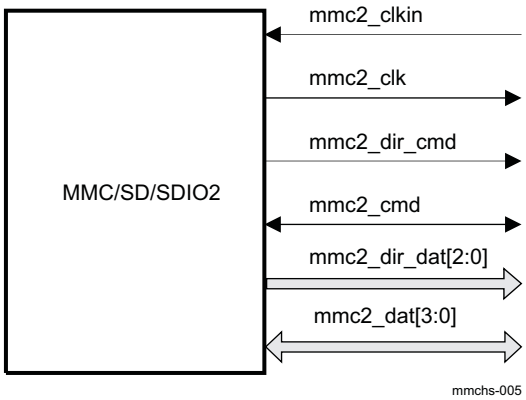


Table 19-2 describes the MMC/SD/SDIO2 inputs/outputs.

Table 19-2. MMC/SD/SDIO2 I/O Description

Signal Name	I/O	Description	Reset Value
mmc2_clk	O	External clock for MMC/SD/SDIO card	0
mmc2_clkin	I	Input clock from MMC/SD/SDIO card	0
mmc2_cmd	I/O	Command signal	0
mmc2_dir_cmd	O	Direction control for mmc2_cmd signal case an external transceiver is used (high when transmit, low when receive)	0
mmc2_dat[3:0]	I/O	Data signals	0
mmc2_dir_dat0	O	Direction control for mmc2_dat0 signal when an external transceiver is used (high when transmit, low when receive)	0
mmc2_dir_dat1	O	Direction control for mmc2_dat1 and mmc2_dat3 signal when an external transceiver is used (high when transmit, low when receive)	0
mmc2_dir_dat2	O	Direction control for mmc2_dat2 signal when an external transceiver is used (high when transmit, low when receive)	0
mmc2_dir_dat3	O	Direction control for mmc2_dat[7:4] signal when an external transceiver is used (high when transmit, low when receive). Unusable on the device because mmci_dat[7:4] are muxed with other direction control signals. See Chapter 6, *System Control Module* for further details on the pin multiplexing.	0

19.2.3.3 MMC/SD/SDIO Protocol and Data Format

The bus protocol between the MMC/SD/SDIOi host controller and the card is message-based. Each message is represented by one of the following parts:

Command: a command starts an operation. The command is transferred serially from the MMC/SD/SDIO host controller to the card on the mmci_cmd line.

Response: a response is an answer to a command. The response is sent from the card to the MMC/SD/SDIO host controller. It is transferred serially on the mmci_cmd line.

Data: data are transferred from the MMC/SD/SDIOi host controller to the card or from a card to the MMC/SD/SDIO host controller using the DATA lines.

Busy: the mmci_dat0 signal is maintained low by the card as far as it is programming the data received.

CRC status: CRC result is sent by the card through the mmci_dat0 line when executing a write transfer. In the case of transmission error, occurring on any of the active data lines, the card sends a negative CRC status on mmci_dat0. In the case of successful transmission, over all active data lines, the card sends a positive CRC status on mmci_dat0 and starts the data programming procedure.

19.2.3.3.1 Protocol

There are two types of data transfer:

- Sequential operation
- Block-oriented operation

There are specific commands for each type of operation (sequential or block-oriented).

See the *Multimedia Card System Specification*, v4.2, the *SD Memory Card Specifications*, v2.0, and the *SDIO Card Specification, Part E1*, August 2004, for details about commands and programming sequences supported by the MMC, SD, and SDIO cards.

Figure 19-7 and Figure 19-8 show how sequential operations are defined. Sequential operation is only for 1-bit transfer and initiates a continuous data stream. The transfer terminates when a stop command follows on the mmci_cmd line.

CAUTION

Stream commands are supported only by MMC cards.

Figure 19-7. Sequential Read Operation (MMC Cards Only)

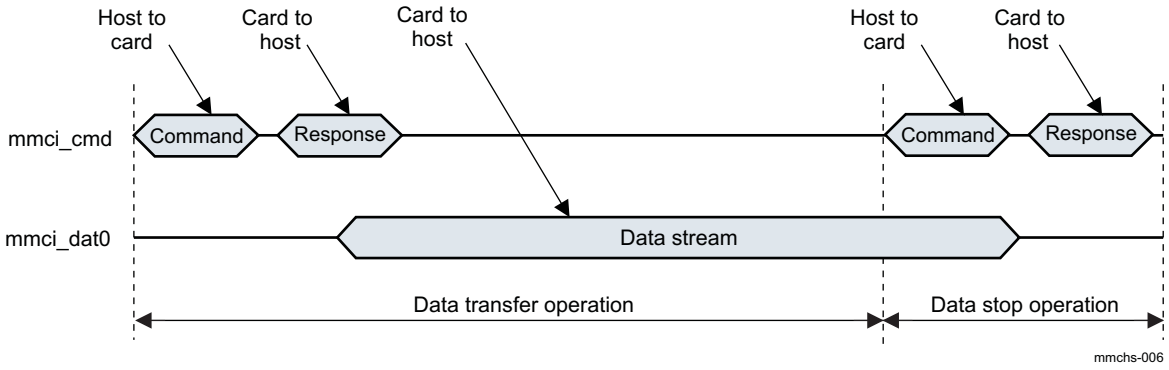


Figure 19-8. Sequential Write Operation (MMC Cards Only)

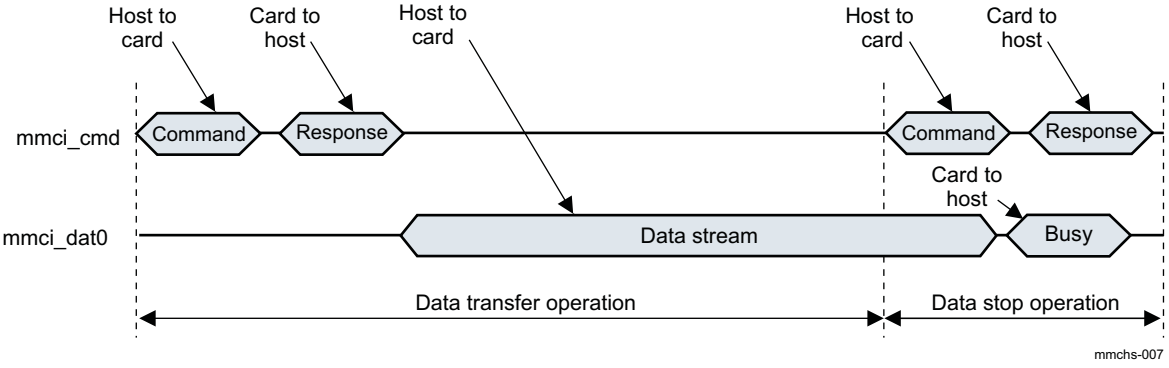


Figure 19-9 and Figure 19-10 show how multiple block-oriented operations are defined. A multiple block-oriented operation sends a data block plus CRC bits. The transfer terminates when a stop command follows on the mmci_cmd line. These operations are available for all kinds of cards.

Figure 19-9. Multiple Block Read Operation

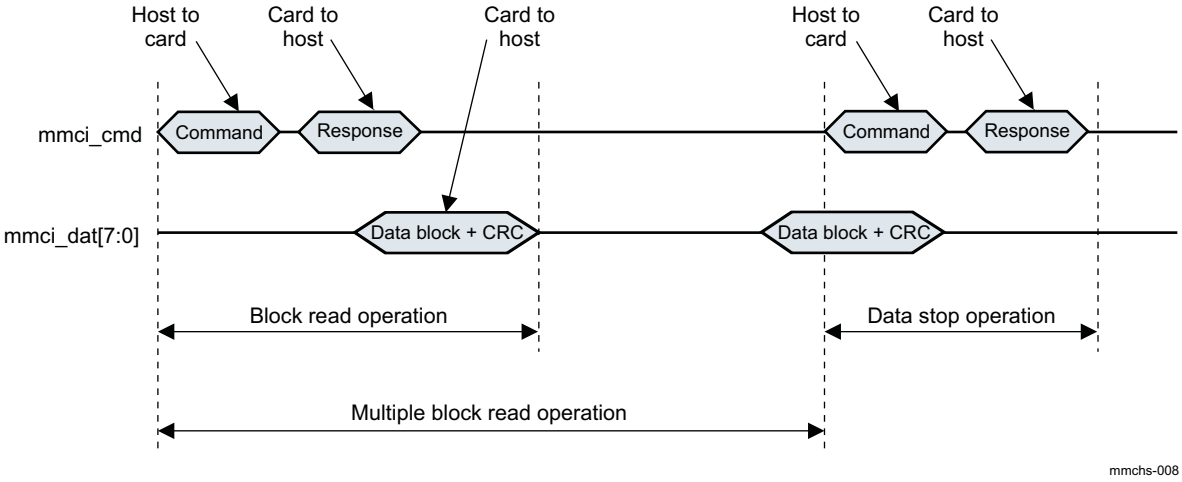
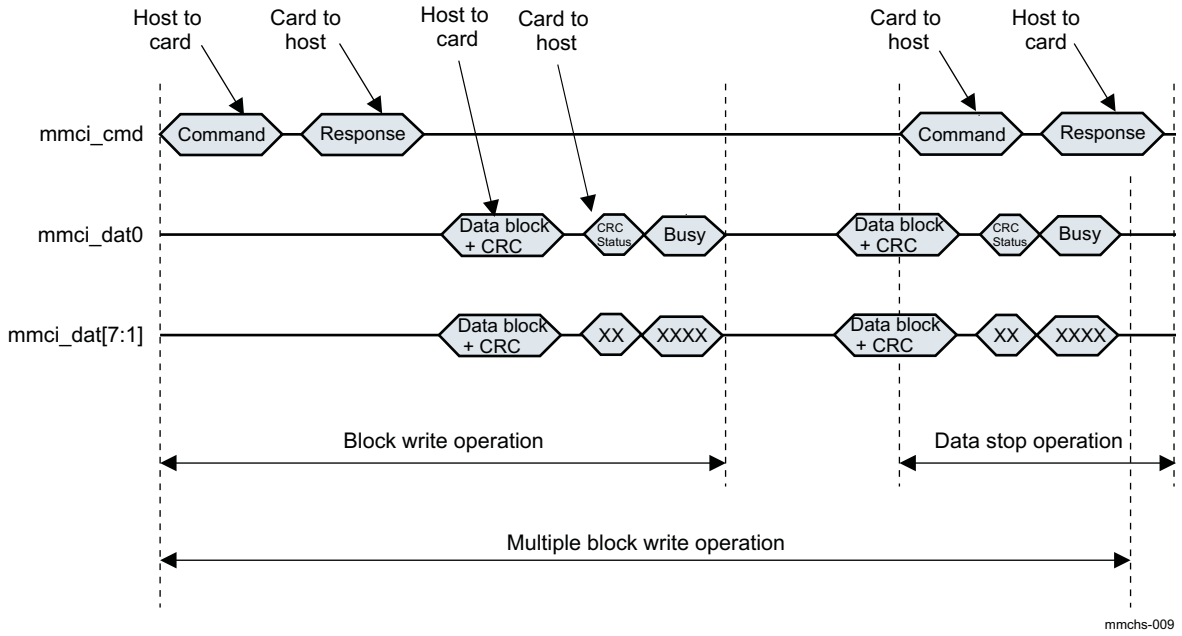


Figure 19-10. Multiple Block Write Operation with Card Busy Signal



NOTE:

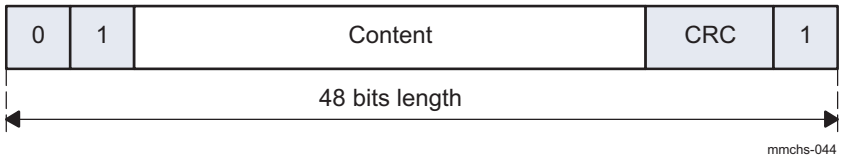
1. The card busy signal is not always generated by the card; the previous examples show a particular case.
2. It is software responsibility to do a software reset (set MMCi.MMCHS_SYSCTL[26] SRD bit to 0x1) after data timeout to ensure mmci_clk is stopped
3. For multiblock transfer, and especially for MMC cards, you can abort a transfer without using a stop command. Use a CMD23 before data transfer to define the number of blocks that will be transferred, then the transfer stops automatically after the last block (if the MMC card supports this feature).

19.2.3.3.2 Data Format

Coding Scheme for Command Token

Command tokens always start with 0 and end with 1. The second bit is a transmitter bit: 1 for a host command. The content is the command index (coded by 6 bits) and an argument (for example, an address), coded by 32 bits. The content is protected by 7-bit CRC checksum (see Figure 19-11).

Figure 19-11. Command Token Format



Coding Scheme for Response Token

Response tokens always start with 0 and end with a 1. The second bit is a transmitter bit: 0 for a card response. The content is different for each type of response (R1, R2, R3, R4, and R5 R6 [for SDIO]) and the content is protected by 7-bit CRC checksum (see Figure 19-12 and Figure 19-13). Depending on the type of commands sent to the card, the MMCHS_CMD register must be configured differently to avoid false CRC or index errors to be flagged on command response (see Table 19-3). For more details about response types, see the *Multimedia Card System Specification*, v4.2, the *SD Memory Card Specifications*, v2.0, or the *SDIO Card Specification, Part E1*, v1.10.

Figure 19-12. Response Token Format (R1, R3, R4, R5, R6)

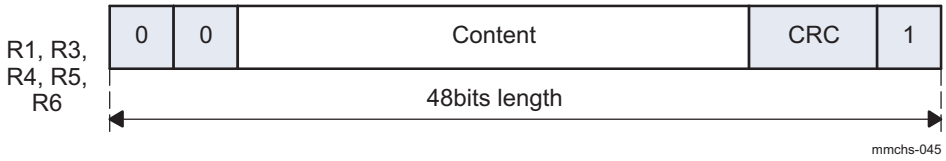


Figure 19-13. Response Token Format (R2)

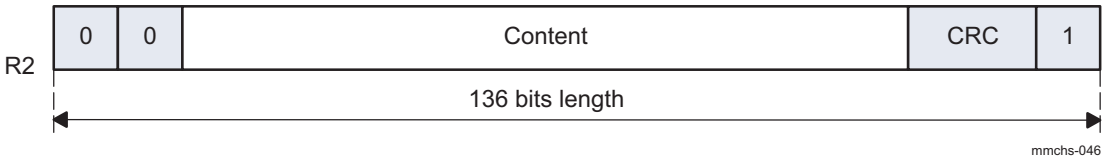


Table 19-3. Relation Between Configuration and Name of Response Type[1]

Response Type MMCi.MMCHS_CMD[17:16] RSP_TYPE	Index Check Enable MMCi.MMCHS_CMD[20] CICE	CRC Check Enable MMCi.MMCHS_CMD[19] CCCE	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3 (R4 for SD cards)
10	1	1	R1, R6, R5
11	1	1	R1b, R5b

(1) The MMC/SD/SDIOi host controller assumes that both clocks may be switched off, whatever the value set in the MMCi.MMCHS_SYSCONFIG[9:8] CLOCKACTIVITY bit.

Coding Scheme for Data Token

Data tokens always start with 0 and end with 1 (see Figure 19-14, Figure 19-15, and Figure 19-16).

Figure 19-14. Data Token Format for 1-Bit Transfers

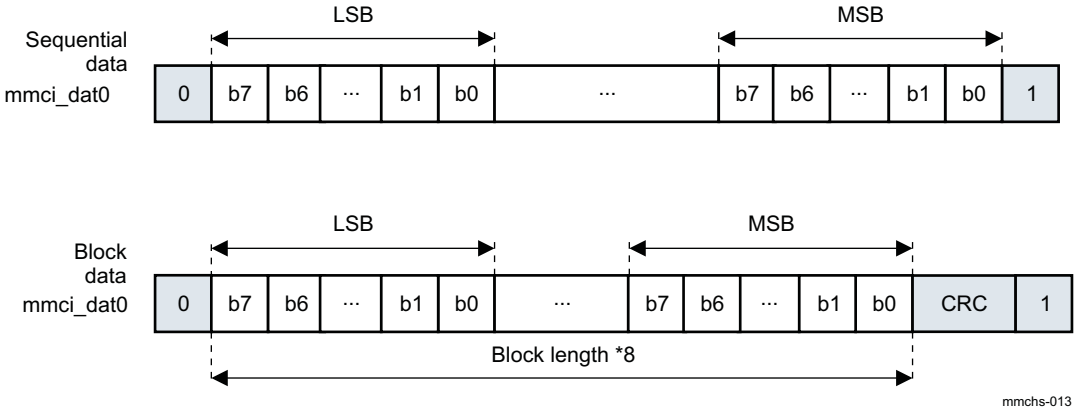


Figure 19-15. Data Token Format for 4-Bit Transfers

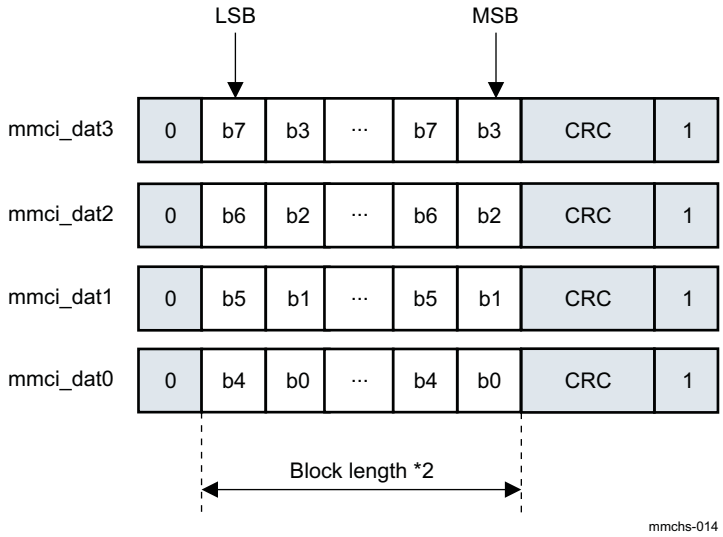
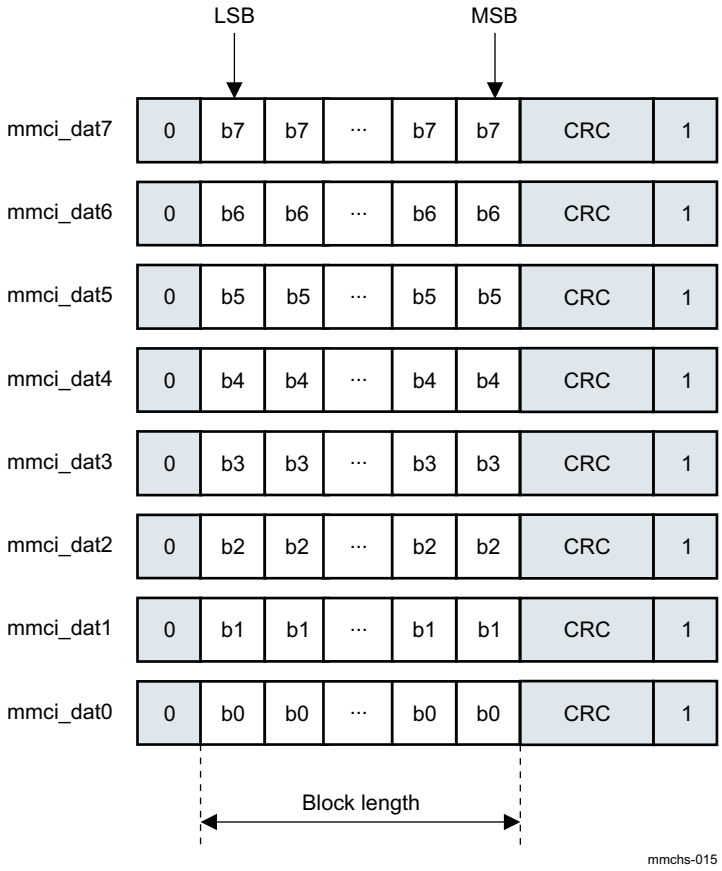


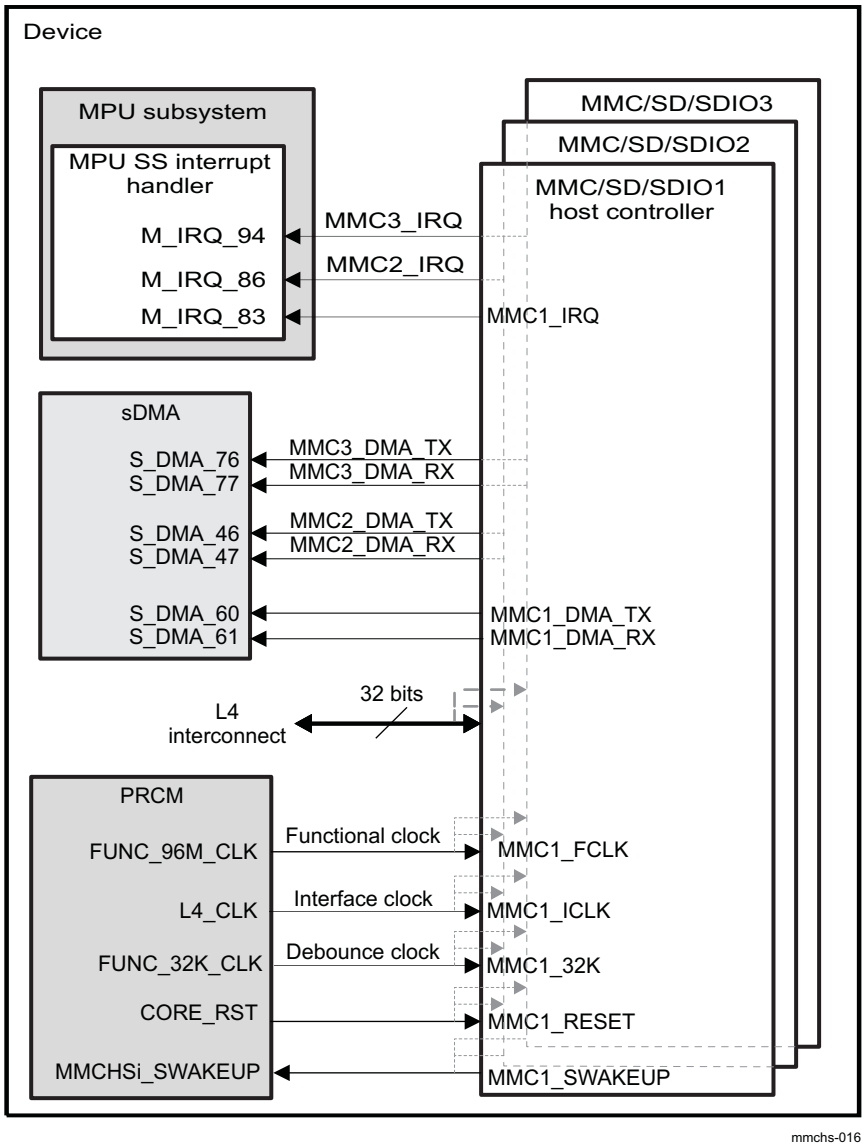
Figure 19-16. Data Token Format for 8-Bit Transfers



19.3 MMC/SD/SDIO Integration

Figure 19-17 shows the internal connections between the three instances of the MMC/SD/SDIO host controller and the other modules

Figure 19-17. MMC/SD/SDIO1 Integration



19.3.1 Clocking, Reset, and Power-Management Scheme

19.3.1.1 Clocks

19.3.1.1.1 Module Clocks

The MMC/SD/SDIO receives three clocks:

- A fixed functional clock of 96 MHz (the MMCi_FCLK) independent of the device external clock frequency
- An interface clock (the MMCi_ICLK) for interfacing with L4 interconnects (register accesses)
- A debounce clock, the MMCi_32K for reset process only

All these clocks are generated and controlled by the power, reset, and clock manager (PRCM) module (see , *Power Reset and Clock Management* for more information).

19.3.1.1.2 Power Management

The MMC/SD/SDIO host controller can enter into different modes and save power:

- Normal mode
- Idle mode

The two modes are mutually exclusive (the module can be in normal mode or in idle mode). The MMC/SD/SDIO host controller is compliant with the PRCM module handshake protocol.

Normal Mode

The autogating of interface and functional clocks occurs when the following conditions are met:

- The MMCi.MMCHS_SYSCONFIG[0] AUTOIDLE bit is set to 1 (I = 1 for MMC/SD/SDIO1, 2 for MMC/SD/SDIO2 and 3 for MMC/SD/SDIO3 instances).
- There is no transaction on the MMC interface.

The autogating of interface and functional clocks stops when the following conditions are met:

- A register access occurs through the L4 interconnect.
- A wake-up event occurs (an interrupt from a SDIO card).
- A transaction on the MMC/SD/SIO interface starts.

Then the MMC/SD/SDIO host controller enters in low-power state (MMCi_ICLK clock autogated) even if MMCi.MMCHS_SYSCONFIG[0] AUTOIDLE is set to 0.

The functional clock is internally switched off and only interconnect read and write accesses are allowed.

Idle Mode

The MMCi_ICLK and MMCi_FCLK clocks provided to MMC/SD/SDIO are switched off upon a PRCM module request. They are switched back upon module request.

The MMC/SD/SDIO host controller complies with the PRCM module handshaking protocol:

- Idle request from the system power manager
- Idle acknowledgment from the MMC/SD/SDIO host controller
- Wake-up request from the MMC/SD/SDIO host controller

The idle acknowledgment varies according to the MMCi.MMCHS_SYSCONFIG[4:3] SIDLEMODE bit field:

- 0x0: Force-idle mode. The MMC/SD/SDIO host controller acknowledges the system power manager request unconditionally.
- 0x1: No-idle mode. The MMC/SD/SDIO host controller ignores the system power manager request and behaves normally as if the request was not asserted.
- 0x2: Smart-idle mode. The MMC/SD/SDIO host controller acknowledges the system power manager request according to its internal state.

During the smart-idle mode period, the MMC/SD/SDIO host controller acknowledges that the MMCi_ICLK and MMCi_FCLK clocks may be switched off whatever the value set in the MMCi.MMCHS_SYSCONFIG[9:8] CLOCKACTIVITY field.

Transition from Normal Mode to Smart-Idle Mode

Smart-idle mode is enabled when the MMCi.MMCHS_SYSCONFIG[4:3] SIDLEMODE bit field is set to 0x2.

The MMC/SD/SDIOi host controller goes into idle mode when the PRCM issues an idle request, according to its internal activity.

During normal to idle mode transition, any access to the registers of the MMC/SD/SDIOi host controller generates an error as long as the MMCi_ICLK clock is alive. The PRCM.CM_IDLEST1_CORE[25] ST_MMC2 and (respectively the PRCM.CM_IDLEST1_CORE[24] ST_MMC1 bit) is set to 0x0 when the MMC/SD/SDIO2 module (respectively the MMC/SD/SDIO1 module) can be accessed.

The MMC/SD/SDIO host controller acknowledges the idle request from the PRCM after ensuring the following:

- The current multi/single-block transfer is completed.
- Any interrupt or DMA request is asserted.
- There is no card interrupt on mmci_dat[1] signal.

As long as the MMC/SD/SDIOi controller do not acknowledge the idle request, if an event occurs, the MMC/SD/SDIOi host controller can still generate an interrupt or a DMA request. In this case, the module ignores the idle request from the PRCM.

As soon as the MMC/SD/SDIOi controller acknowledges the idle request from the PRCM, the module does not assert any new interrupt or DMA request.

Wake-Up Event in Smart-Idle Mode

The wake-up feature is enabled when the following enable wake-up bits are set:

- MMCi.MMCHS_SYSCONFIG[2] ENAWAKEUP bit is set to 0x1
- MMCi.MMCHS_HCTL[24] IWE bit is set to 0x1
- MMCi.MMCHS_IE[8] CIRQ_ENABLE bit is set to 0x1

The wakeup is generated only in smart-idle mode only, when module is in idle mode.

Table 19-4 lists the supported cases in smart-idle mode.

Table 19-4. Smart Idle Mode and Wake-Up Capabilities

Mode	MMCi_ICLK clock	MMCi_FCLK clock	Wake-up Event
Card interrupt	May be switched off(1)	May be switched off(1)	The module sends an asynchronous wake-up request on detection of a card interrupt on mmci_dat[1] signal

(1) The MMC/SD/SDIOi host controller assumes that both clocks may be switched off, whatever the value set in the MMCi.MMCHS_SYSCONFIG[9:8] CLOCKACTIVITY bit.

Transition from Smart-Idle Mode to Normal Mode

The MMC/SD/SDIO host controller detects the end of the idle period when the PRCM deasserts the idle request.

For the wake-up event, there is a corresponding interrupt status in the MMCi.MMCHS_STAT register. The MMC/SD/SDIOi host controller operates the conversion between wake-up and interrupt (or DMA request) upon exit from smart-idle mode if the associated enable bit is set in the MMCi.MMCHS_ISE register.

Interrupts and wake-up events have independent enable/disable controls, accessible through the MMCi.MMCHS_HCTL and MMCi.MMCHS_ISE registers. The overall consistency must be ensured by software.

The interrupt status register MMCi.MMCHS_STAT is updated with the event that caused the wake-up in the CIRQ bit when the MMCi.MMCHS_IE[8] CIRQ_ENABLE associated bit is enabled.

Then, the wake-up event at the origin of the transition from smart-idle mode to normal mode is converted into its corresponding interrupt or DMA request. (The MMCi.MMCHS_STAT register is updated and the status of the interrupt signal changes.)

When the idle request from the PRCM is deasserted, the module switches back to normal mode. The module is fully operational.

Force-Idle Mode

Force-idle mode is enabled when the MMCi.MMCHS_SYSCONFIG[4:3] SIDLEMODE bit field is set to 0x0.

Force-idle mode is an idle mode where the MMC/SD/SDIOi host controller responds unconditionally to the idle request from the PRCM. Moreover, in this mode, the MMC/SD/SDIOi host controller unconditionally deasserts interrupts and DMA request lines asserted.

The transition from normal mode to force-idle mode does not affect the bits of the MMCi.MMCHS_STAT register.

In force-idle mode, the interrupt and DMA request lines are deasserted. MMCi_ICLK and MMCi_FCLK can be switched off.

CAUTION

In force-idle mode, an idle request from the PRCM during a command or a data transfer can lead to an unexpected and unpredictable result. When the module is idle, any access to the module generates an error as long as the MMCi_ICLK clock is alive.

The module exits the force-idle mode when the PRCM deasserts the idle request. Then the module switches back to normal mode. The module is fully operational. Interrupt and DMA request lines are optionally asserted one clock cycle later.

19.3.1.2 Resets

19.3.1.2.1 Hardware Reset

The module is reinitialized by the hardware when the active-low reset signal (CORE_RST), synchronous to the MMCi_ICLK clock is asserted on the input pin MMCi_RESET (see , *Power, Reset, and Clock Management* for more information).

A global status bit RESETDONE is provided in the status register MMCi.MMCHS_SYSSTATUS[0]. This bit is set to 1 when all the different clock domain resets (interface domain, functional domain, and 32K domain) have been released.

The MMCi.MMCHS_SYSSTATUS[0] RESETDONE bit can be monitored by the software to check if the module is ready-to-use after a hardware reset.

NOTE: Functional clock MMCi_FCLK, interface clock MMCi_ICLK, and debounce clock MMCi_32K must be provided to the module to allow the RESETDONE status bit to be set.

This hardware reset signal has a global reset action on the module. All configuration registers and all state-machines are reset in all clock domains.

19.3.1.2.2 Software Reset

The module is reinitialized by software through the MMCi.MMCHS_SYSCONFIG[1] SOFTRESET bit. This bit has the same action on the module logic as the hardware MMCi_RESET signal except for:

- Debounce logic
- MMCi.MMCHS_PSTATE, MMCi.MMCHS_CAPA, and MMCi.MMCHS_CUR_CAPA registers (see corresponding register descriptions)

The SOFTRESET bit is active high. The bit is automatically reinitialized to 0 by the hardware. The MMCi.MMCHS_SYSCTL[24] SRA bit has the same action as the SOFTRESET bit on the design.

The MMCi.MMCHS_SYSSTATUS[0] RESETDONE bit can be monitored by the software to check if the module is ready-to-use after a software reset.

Moreover, two partial software reset bits are provided:

- MMCi.MMCHS_SYSCTL[26] SRD bit
- MMCi.MMCHS_SYSCTL[25] SRC bit

These two reset bits are useful to reinitialize data or command processes respectively in case of line conflict. When set to 1, a reset process is automatically released when the reset completes:

- The MMCi.MMCHS_SYSCTL[26] SRD bit resets all finite state-machines and status management that

handle data transfers on both the interface and functional side.

- The MMCi.MMCHS_SYSCTL[25] SRC bit resets all finite state-machines and status management that handle command transfers on both the interface and functional side.

19.3.1.3 Power Domain

MMC/SD/SDIOi power is supplied by the CORE power domain (see , *Power Reset and Clock Management* for more information).

When the MMC/SD/SDIOi power domain is off, the only way to wake up the power domain and different MMC/SD/SDIOi clocks is to monitor mmci_dat[1] input pin state via a different GPIO line for each MMC/SD/SDIO interface (see Chapter 21, *General-Purpose Interface*, for more information).

19.3.2 Hardware Requests

19.3.2.1 DMA Requests

The MMC/SD/SDIOi host controller can be interfaced with a DMA controller. At system level, the advantage is to discharge the LH of the data transfers. The module does not support wide DMA access (above 1024 bytes) for SD cards as specified in the *SD Card Specification, Part A2, SD Host Controller Standard Specification*, v1.00.

The DMA request is issued if the three following conditions are met:

- The MMCi.MMCHS_CMD[0] DE bit is set to 1 to trigger the initial DMA request (the write must be done when running the data transfer command).
- A command was emitted on the mmci_cmd line.
- There is enough space in the buffer of the MMC/SD/SDIOi host controller to write an entire block (BLEN writes).

DMA request lines are connected on the system DMA (sDMA) inputs:

- S_DMA_60 (MMC1_DMA_TX)
- S_DMA_61 (MMC1_DMA_RX)
- S_DMA_46 (MMC2_DMA_TX)
- S_DMA_47 (MMC2_DMA_RX)
- S_DMA_76 (MMC3_DMA_TX)
- S_DMA_77 (MMC3_DMA_RX)

19.3.2.1.1 DMA Receive Mode

In a DMA block read operation (single or multiple), the request signal MMCi_DMA_RX is asserted to its active level when a complete block is written in the buffer. The block size transfer is specified in the MMCi.MMCHS_BLK[10:0] BLEN field.

The MMCi_DMA_RX signal is deasserted to its inactive level when the sDMA has read one single word from the buffer.

Only one request is sent per block; the DMA controller can make a 1-shot read access or several DMA bursts, in which case the DMA controller must manage the number of burst accesses, according to block size BLEN field.

New DMA requests are internally masked if the sDMA has not read exactly BLEN bytes and a new complete block is not ready. As DMA accesses are in 32-bit, then the number of sDMA read is Integer(BLEN/4)+1.

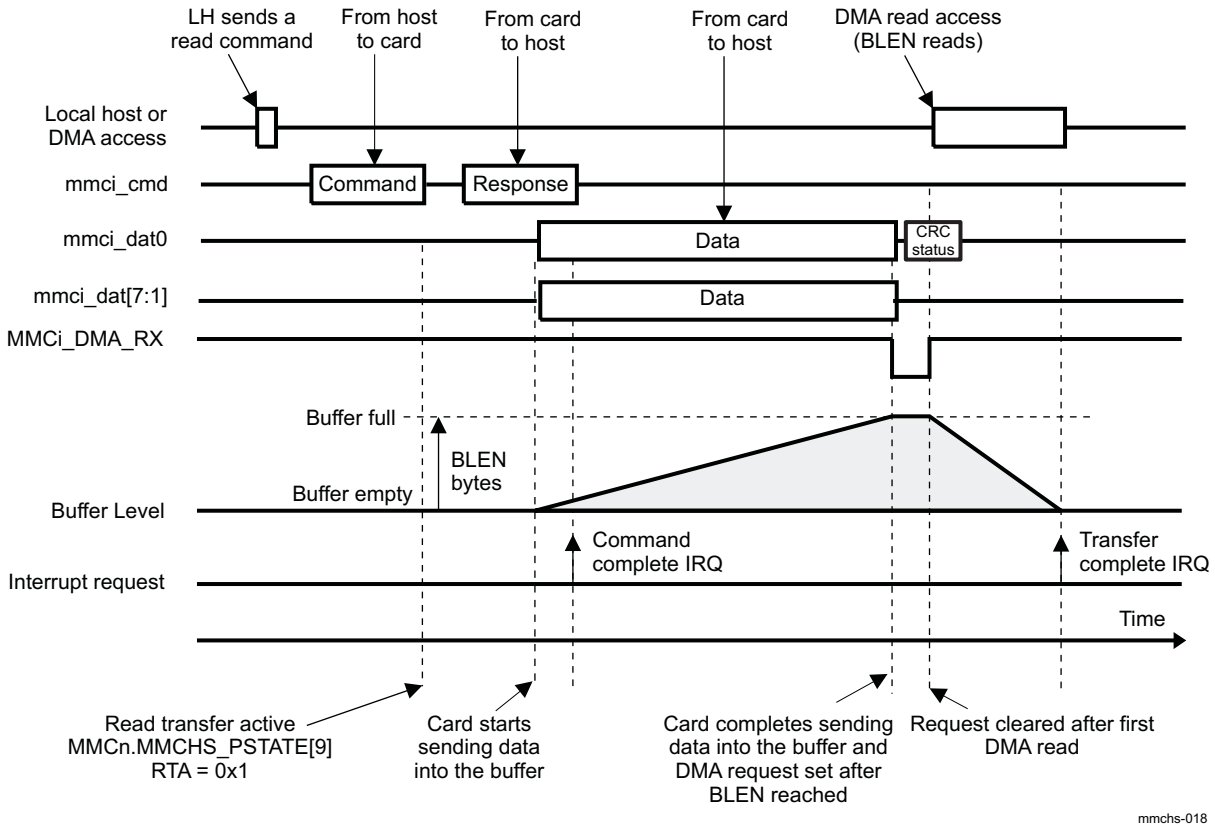
The receive buffer never overflows. In multiple block transfers for block size above 512 bytes, when the buffer gets full, the mmci_clk clock signal (provided to the card) is momentarily stopped until the sDMA or the MPU performs a read access, which reads a complete block in the buffer.

Summary (see Figure 19-18):

- DMA transfer size = BLEN buffer size (maximum 1024 32-bit words) in one shot or by burst

- One DMA request per block

Figure 19-18. DMA Receive Mode



19.3.2.1.2 DMA Transmit Mode

In a DMA block write operation (single or multiple), the request signal MMCi_DMA_TX is asserted to its active level when a complete block is to be written to the buffer. The block size transfer is specified in the MMCi.MMCHS_BLK[10:0] BLEN field.

The MMCi_DMA_TX signal is deasserted to its inactive level when the sDMA has written one single word to the buffer.

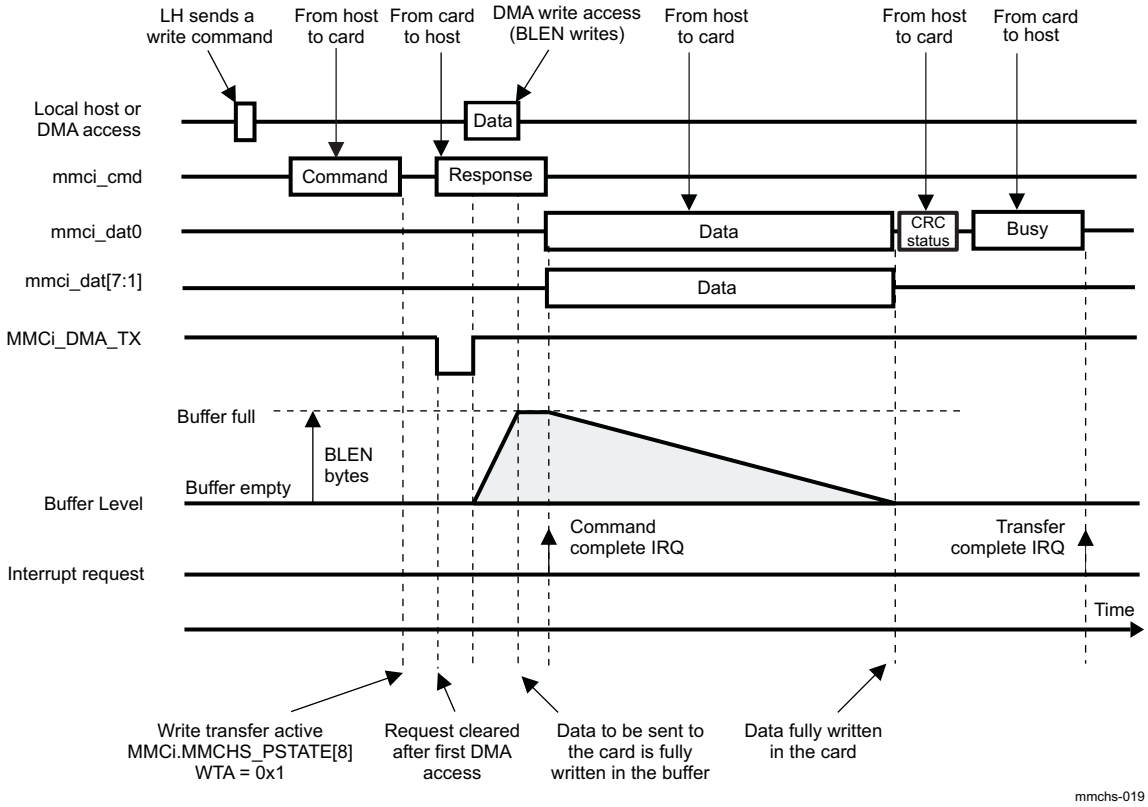
Only one request is sent per block; the DMA controller can make a 1-shot write access or multiple write DMA bursts, in which case the DMA controller must manage the number of burst accesses, according to block size BLEN field.

New DMA requests are internally masked if the sDMA has not written exactly BLEN bytes (as DMA accesses are in 32-bit, then the number of sDMA read is Integer(BLEN/4)+1) and if there is not enough memory space to write a complete block in the buffer.

Summary (see Figure 19-19):

- DMA transfer size = BLEN buffer size (maximum 1024 32-bit words) in one shot or by burst
- One DMA request per block

Figure 19-19. DMA Transmit Mode



19.3.2.2 Interrupt Requests

Several internal module events can generate an interrupt. Each interrupt has a status bit, an interrupt enable bit, and a signal status enable:

- The status of each type of interrupt is automatically updated in the MMCi.MMCHS_STAT register; it indicates which service is required.
- The interrupt status enable bits of the MMCi.MMCHS_IE register enable/disable the automatic update of the MMCi.MMCHS_STAT register on an event-by-event basis.
- The interrupt signal enable bits of the MMCi.MMCHS_ISE register enable/disable the transmission of an interrupt request on the interrupt line MMCi_IRQ (from the MMC/SD/SDIOi host controller to the MPU subsystem interrupt controller) on an event-by-event basis.

If an interrupt status is disabled in the MMCi.MMCHS_IE register, then the corresponding interrupt request is not transmitted, and the value of the corresponding interrupt signal enable in the MMCi.MMCHS_ISE register is ignored.

When an interrupt event occurs, the corresponding status bit is automatically set to 0x1 (the MMC/SD/SDIOi host controller updates the status bit) in the MMCi.MMCHS_STAT register. If later a mask is applied on the interrupt in the MMCi.MMCHS_ISE register, the interrupt request is deactivated.

When the interrupt source has not been serviced, if the interrupt status is cleared in the MMCi.MMCHS_STAT register and the corresponding mask is removed from the MMCi.MMCHS_ISE register, the interrupt status is not asserted again in the MMCi.MMCHS_STAT register and the MMC/SD/SDIOi host controller does not transmit an interrupt request.

CAUTION

If the buffer write ready interrupt (BWR) or the buffer read ready only interrupt (BRR) are not serviced and are cleared in the MMCi.MMCHS_STAT register, and the corresponding mask is removed, then the MMC/SD/SDIOi host controller will wait for the service of the interrupt without updating the status MMCi.MMCHS_STAT or transmitting an interrupt request.

The MMC/SD/SDIOi host controller supports interrupt-driven operation and polling.

There are one interrupt line for each instance of the MMC/SD/SDIOi host controller:

- MMC1_IRQ is connected to M_IRQ_83 for MMC/SD/SDIO1 instance.
- MMC2_IRQ is connected to M_IRQ_86 for MMC/SD/SDIO2 instance.
- MMC3_IRQ is connected to M_IRQ_94 for MMC/SD/SDIO3 instance.

19.3.2.2.1 Interrupt-Driven Operation

An interrupt enable bit must be set in the MMCi.MMCHS_IE register to enable the module internal source of interrupt.

When an interrupt event occurs, the single interrupt line is asserted and the LH must:

- Read the MMCi.MMCHS_STAT register to identify which event occurred.
- Write 1 into the corresponding bit of the MMCi.MMCHS_STAT register to clear the interrupt status and release the interrupt line (if a read is done after this write, this would return 0).

NOTE: In the MMCi.MMCHS_STAT register, Card Interrupt (CIRQ) and Error Interrupt (ERRI) bits cannot be cleared.
The MMCi.MMCHS_STAT[8] CIRQ status bit must be masked by disabling the MMCi.MMCHS_IE[8] CIRQ_ENABLE bit (set to 0x0), then the interrupt routine must clear SDIO interrupt source in SDIO card common control register (CCCR). See Chapter 5, *Interconnect* for more information.
The MMCi.MMCHS_STAT[15] ERRI bit is automatically cleared when all status bits in MMCi.MMCHS_STAT[31:16] are cleared.

19.3.2.2.2 Polling

When the interrupt capability of an event is disabled in the MMCi.MMCHS_ISE register, the interrupt line is not asserted:

- Software can poll the status bit in the MMCi.MMCHS_STAT register to detect when the corresponding event occurs.
- Writing 1 into the corresponding bit of the MMCi.MMCHS_STAT register clears the interrupt status and does not affect the interrupt line state.

NOTE: Refer to the previous note concerning CIRQ and ERRI bits clearing.

19.4 MMC/SD/SDIO Functional Description

19.4.1 Description

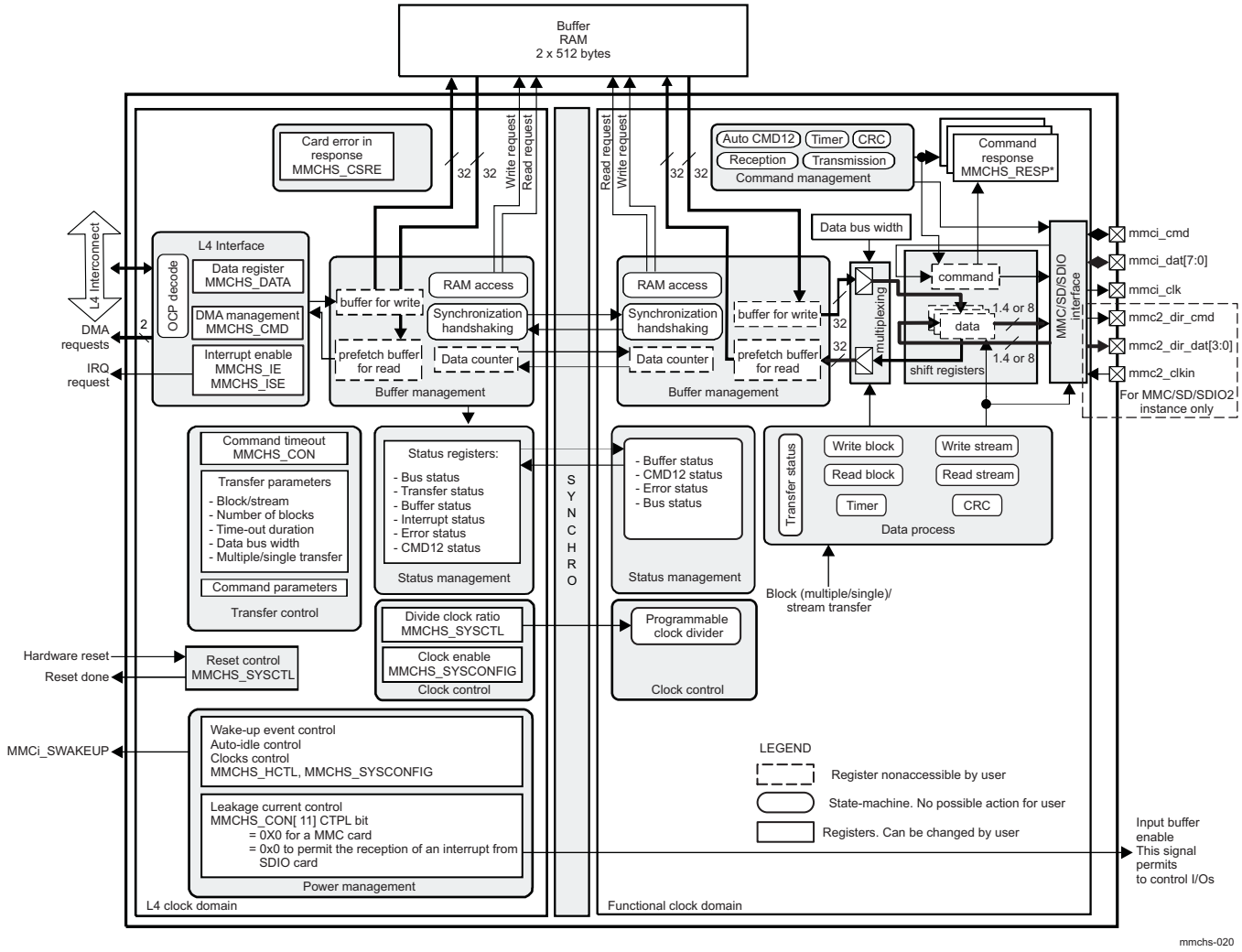
MMC/SD/SDIOi host controller is partitioned into two clock domains:

- The interface L4 clock domain
- The functional clock domain

Any two domains are considered asynchronous to each other, and exchanges between the two domains are synchronized through a synchronization stage and an asynchronous buffer. Data are transferred from one domain to other through the buffer (2*512 RAM).

Figure 19-20 shows a block diagram of the MMC/SD/SDIO host controller.

Figure 19-20. MMC/SD/SDIO Diagram



In the L4 clock domain, the user can:

- Control the transfer (configure parameters for the transfer)
- Control clock activities and reset
- Manage interrupts and hardware requests

- Consult different status:
  - Bus
  - Buffer
  - Interrupts
  - Errors
- Transmit and receive data through the buffer

In the functional clock domain, the internal mechanisms perform the transfers of data and commands.

For more detail see the command response registers (MMCi.MMCHS_RSPn registers: , , , and).

CAUTION

Read access to the command response registers is allowed only when the command process is completed.

19.4.2 Mode Selection

The MMC/SD/SDIO host controller can be use in two modes: MMC and SD/SDIO modes. It has been designed to be the most transparent with the type of card.

The type of the card connected is differentiated by the software initialization procedure. Software identifies the type of card connected during software initialization. For each given card type, there are corresponding commands. Some commands are not supported by all cards. See the *Multimedia Card System Specification*, v4.2, the *SD Memory Card Specifications*, v2.0, and the *SDIO Card Specification, Part E1*, v1.10, for more details.

The purpose of the module is to transfer commands and data, to whatever card is connected, respecting the protocol of the connected card.

Writes and reads to the card must respect the appropriate protocol of that card.

19.4.3 Buffer Management

19.4.3.1 Data Buffer

The MMC/SD/SDIOi host controller uses a data buffer divided into two 512-byte portions that are 32 bits wide by 128 words deep. This buffer transfers data from one data bus (Interconnect) to another data bus (SD SDIO or MMC card bus) and vice versa.

The buffer is the heart of the interface and ensures the transfer between the two interfaces (L4 and the card).

To enhance performance, the data buffer is completed by a prefetch register and a post-write buffer that are not accessible by the host controller.

The read access time of the prefetch register is faster than the one of the data buffer. The prefetch register allows data to be read from the data buffer at an increased speed by preloading data into the prefetch register.

The entry point of the data buffer for the two portions, the prefetch buffer and the post-write buffer, is the 32-bit register MMCi.MMCHS_DATA. A write access to the MMCi.MMCHS_DATA register followed by a read access from the MMCi.MMCHS_DATA register corresponds to a write access to the post-write buffer followed by a read access to the prefetch buffer. As a consequence, it is normal that the data of the write access to the MMCi.MMCHS_DATA register and the data of the read access to the MMCi.MMCHS_DATA register are different.

The number of 32-bit accesses to the MMCi.MMCHS_DATA register that are needed to read (or write) a data block with a size of MMCi.MMCHS_BLK[10:0] BLEN, and equals the rounded up result of BLEN divided by 4.

The maximum block size supported by the host controller is 1024 bytes. This value is hard-coded in the register MMCi.MMCHS_CAPA[17:16] MBL field and cannot be changed.

A read access to the MMCi.MMCHS_DATA register is allowed only when the buffer read enable status is set to 1 (MMCi.MMCHS_PSTATE[11] BRE); otherwise, a bad access (MMCi.MMCHS_STAT[29] BADA) is signaled.

A write access to the MMCi.MMCHS_DATA register is allowed only when the buffer write enable status is set to 1 (MMCi.MMCHS_PSTATE[10] BWE); otherwise, a bad access (MMCi.MMCHS_STAT[29] BADA) is signaled and the data is not written.

The data buffer has two modes of operation to store and read of the first and second portions of the data buffer:

- When the size of the data block to transfer is less than or equal to 512 bytes, meaning the value written in BLEN is less than or equal to 0x200, two data transfers can occur from one data bus to the other data bus and vice versa at the same time. The MMC/SD/SDIOi host controller uses the two portions of the data buffer in a ping pong manner so that storing and reading of the first and second portions of the data buffer are automatically interchanged from time to time so that data may be read from one portion (for instance, through a DMA read access on the interconnect bus) while data (for instance, from the card) is being stored into the other portion and vice versa. When BLEN is less than, or equal to 0x200 (that is, less than, or equal to 512 bytes), each of the two portions of the buffer that can be used have a size of BLEN (that is, 32 bits x BLEN div by 4). Do not use a size greater than 2 times this value.

CAUTION

The MMCi.MMCHS_CMD[4] DDIR bit must be configured before a transfer to indicate the direction of the transfer.

Figure 19-21 and Figure 19-22 show the buffer management for a write and for a read, respectively.

Figure 19-21. Buffer Management for a Write

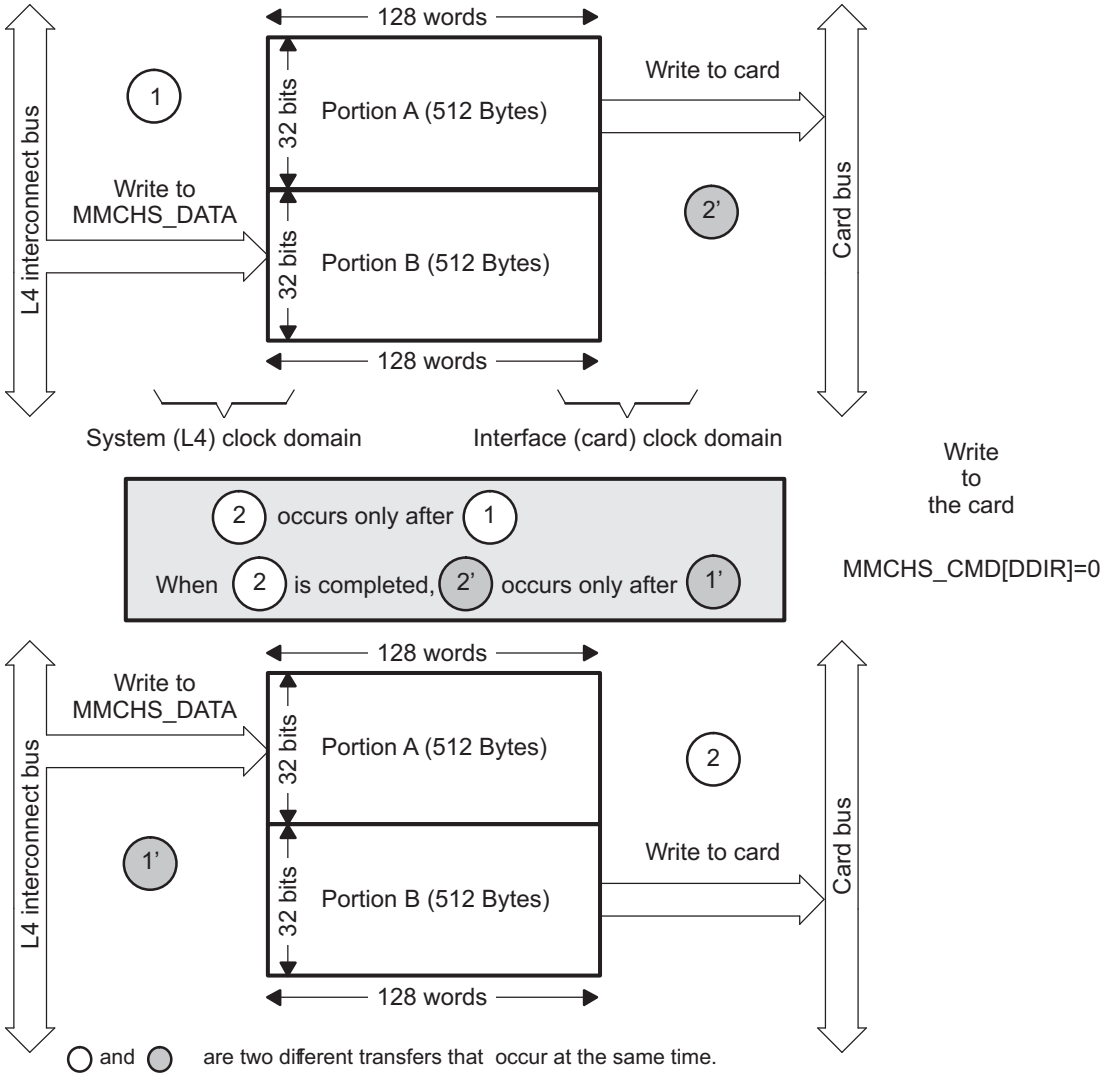
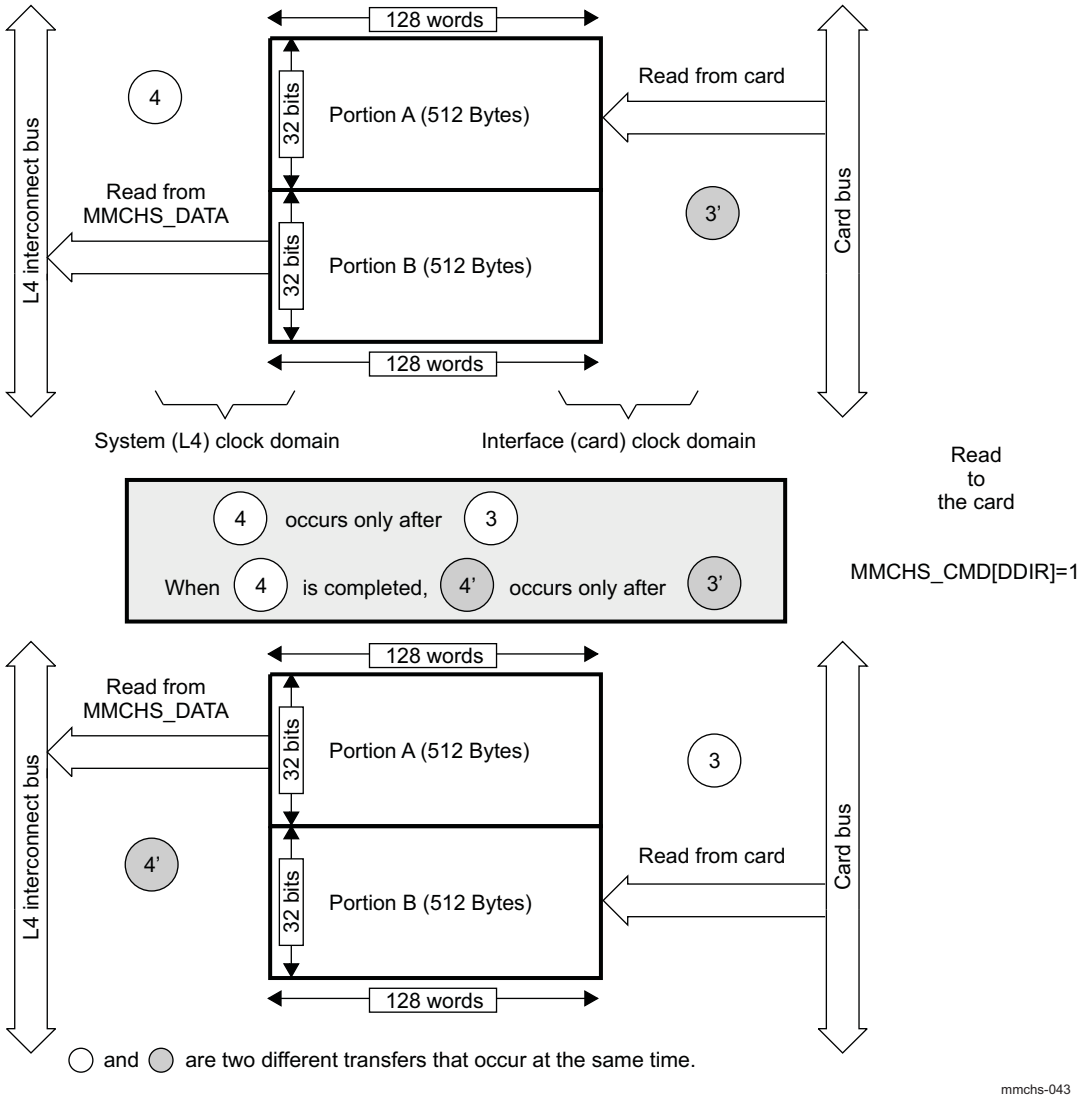


Figure 19-22. Buffer Management for a Read



- When the size of the data block to transfer is larger than 512 bytes, meaning the value written in BLEN is 0x201 or larger, only one data transfer can occur from one data bus to the other data bus at a time. The MMC/SD/SDIOi host controller uses the entire data buffer as a single 1024-byte portion. In this mode, a bad access (MMCi.MMCHS_STAT[29] BADA) is signaled when two data transfers occur from one data bus to the other data bus and vice versa at the same time.

19.4.3.1.1 Data Buffer Status

The data buffer status is defined in the following interrupt status register and status register:

- Interrupt status registers (see):
  - MMCi.MMCHS_STAT[29] BADA: Bad access to data space
  - MMCi.MMCHS_STAT[5] BRR: Buffer read ready
  - MMCi.MMCHS_STAT[4] BWR: Buffer write ready
- Status registers (see):
  - MMCi.MMCHS_PSTATE[11] BRE: Buffer read enable
  - MMCi.MMCHS_PSTATE[10] BWE: Buffer write enable

19.4.4 Transfer Process

The process of a transfer is dependent on the type of command. It can be with or without a response, with or without data.

19.4.4.1 Different Types of Commands

Different types of commands are specific to MMC, SD, or SDIO cards. See the *Multimedia Card System Specification*, v4.2, the *SD Memory Card Specifications*, v2.0, the *SDIO Card Specification, Part E1*, v1.10, or the *SD Card Specification, Part A2, SD Host Controller Standard Specification*, v1.00, for more details.

19.4.4.2 Different Types of Responses

Different types of responses are specific to MMC, SD, or SDIO cards. See the *Multimedia Card System Specification*, v4.2, the *SD Memory Card Specifications*, v2.0, the *SDIO Card Specification, Part E1*, v1.10, or the *SD Card Specification, Part A2, SD Host Controller Standard Specification*, v1.00, for more details.

Table 19-5 shows how the MMC, SD, and SDIO responses are stored in the MMCHS_RSPxx registers.

Table 19-5. MMC, SD, SDIO responses in the MMCHS_RSPxx registers

Kind of Response	Response Field	Response Register
R1, R1b (normal response), R3, R4, R5, R5b, R6	RESP[39:8](1)	MMCHS_RSP10[31:0]
R1b (Auto CMD12 response)	RESP[39:8](1)	MMCHS_RSP76[31:0]
R2	RESP[127:0](1)	MMCHS_RSP76[31:0] MMCHS_RSP54[31:0] MMCHS_RSP32[31:0] MMCHS_RSP10[31:0]

(1) RESP refers to the command response format described in the specifications mentioned above.

When the host controller modifies part of the MMCHS_RSPxx registers, it preserves the unmodified bits.

The host controller stores the Auto CMD12 response in the MMCHS_RSP76[31:0] register because the Host Controller may have a multiple block data DAT line transfer executing concurrently with a command. This allows the host controller to avoid overwriting the Auto CMD12 response with the command response stored in MMCHS_RSP10 register and vice versa.

19.4.5 Transfer or Command Status and Errors Reporting

Flags in the MMC/SD/SDIOi host controller show status of communication with the card:

- A timeout (of a command, a data, or a response)
- A CRC

Error conditions generate interrupts. See Table 19-6 and register description for more details.

Table 19-6. CC and TC Values Upon Error Detected

Error hold in the MMCi.MMCHS_STAT register		CC	TC	Comments
29	BADA			No dependency with CC nor TC BADA is related to MMCHS_DATA register accesses. Its assertion is not dependent of the ongoing transfer.
28	CERR	1		CC is set upon CERR.
22	DEB		1	TC is set upon DEB.
21	DCRC		1	TC is set upon DCRC.
20	DTO			DTO and TC are mutually exclusive DCRC and DEB cannot occur with DTO.
19	CIE	1		CC is set upon CIE.
18	CEB	1		CC is set upon CEB.

Table 19-6. CC and TC Values Upon Error Detected (continued)

Error hold in the MMCi.MMCHS_STAT register		CC	TC	Comments
17	CCRC	1		CC can be set upon CCRC - See CTO comment
16	CTO			CTO and CC are mutually exclusive. CIE, CEB and CERR cannot occur with CTO. CTO can occur at the same time as CCRC: it indicates a command abort due to a contention on CMD line. In this case no CC appears.

19.4.6 Transfer Stop

Whenever a transfer is initiated, the transmission may be willed to stop whereas it is still not finished. Several cases can be faced depending on the transfer type:

- Multiple blocks oriented transfers (for which transfer length is known)
- Continuous stream transfers (which have an infinite length)

NOTE: Since the MMC/SD/SDIOi controller manages transfers based on a block granularity, the buffer will accept a block only if there is enough space to completely store it. Consequently, if a block is pending in the buffer, no command will be sent to the card because the card clock will be shut off by the controller.

The MMC/SD/SDIOi controller includes two features which makes a transfer stop more convenient and easier to manage:

- Auto CMD12 (for MMC and SD only).

  This feature is enabled by setting the MMCi.MMCHS_CMD[2] ACEN bit to 0x1 (this setting is relevant for a MMC/SD transfer with a known number of blocks to transfer). When the Auto CMD12 feature is enabled, the MMC/SD/SDIOi controller will automatically issue a CMD12 command when the expected number of blocks has been exchanged.
- Stop at block gap

  This feature is enabled by setting the MMCi.MMCHS_HCTL[16] SBGR bit to 0x1. When enabled, this capability holds the transfer on until the end of a block boundary. If a stop transmission is needed, software can use this pause to send a CMD12 to the card.

NOTE: For MMC and SD cards, the stop at block gap feature is not supported in READ mode.

For SDIO cards, this setting can be supported in READ mode if the card has a read wait capability.

Table 19-7 shows the common way to stop a transfer, indicating command to send and features to enable.

Table 19-7. MMC/SD/SDIOi Controller Transfer Stop Command Summary

	WRITE transfer		READ transfer	
	SD / MMC	SDIO	SD / MMC	SDIO
Single block	Transfer ends automatically Wait TC	Transfer ends automatically Wait TC	Transfer ends automatically Wait TC	Transfer ends automatically Wait TC

Table 19-7. MMC/SD/SDIOi Controller Transfer Stop Command Summary (continued)

		WRITE transfer		READ transfer	
Multi blocks (finite or infinite)	Before the programmed block boundary	Send CMD12 Wait TC	Send CMD52 Wait TC	Send CMD12 Wait TC	Send CMD52 Wait TC
	Stop at the end of the transfer (finite transfer only)	Auto CMD12 active Transfer ends automatically Wait TC	Set MMCi.MMCHS_HCTL[16] SBGR bit to 0x1. Send CMD52 Wait TC	Auto CMD12 active Transfer ends automatically Wait TC	**If READ_WAIT supported** Stop at block gap Wait TC
					If READ_WAIT not supported Send CMD52 Wait TC

NOTE: The MMC/SD/SDIOi controller will send the stop command to the card on a block boundary, regardless the moment the command was written to the controller registers.

19.4.7 MMC CE-ATA Command Completion Disable Management

The MMC/SD/SDIOi host controller supports CE-ATA features, in particular the detection of command completion token. When a command that requires a command completion signal (MMCHS_CON[12] CEATA and MMCHS_CMD[2] ACEN set to 1) is launched, host system is no longer allowed to emit a new command in parallel of data transfer unless it is a command completion disable.

The settings to emit a command completion disable token follow:

- MMCHS_CON[12] CEATA is set to 1.
- MMCHS_CON[2] HR set to 1.
- Clear the MMCHS_ARG register.
- Write into MMCHS_CMD register with value 0x00000000.

When a command completion disable token was emitted (that is, MMCHS_STAT[0] CC received), the host system is again allowed to emit another type of command (for example a transfer abort command CMD12 to abort transfer).

A critical case can be met when command completion signal disable (CCSD) is emitted during the last data block transfer, the sequence on command line could be sent very close to command completion signal (CCS) token sent by the card.

Three cases can be met:

- CCS is receive just before CCSD is emitted:

  An interrupt CIRQ is generated with CCS detection, CCSD is transmitted to card then an interrupt CC is generated when CCSD ends. In this case, card consider the CCSD sequence.
- CCS is not generated or generated during the CCSD transfer:

  The CCS bit cannot be detected (conflict is not possible as they drive the same level on command line, then no CIRQ interrupt is generated; besides CC interrupt is generated when CCSD ends).
- CCS is generated without CCSD token required:

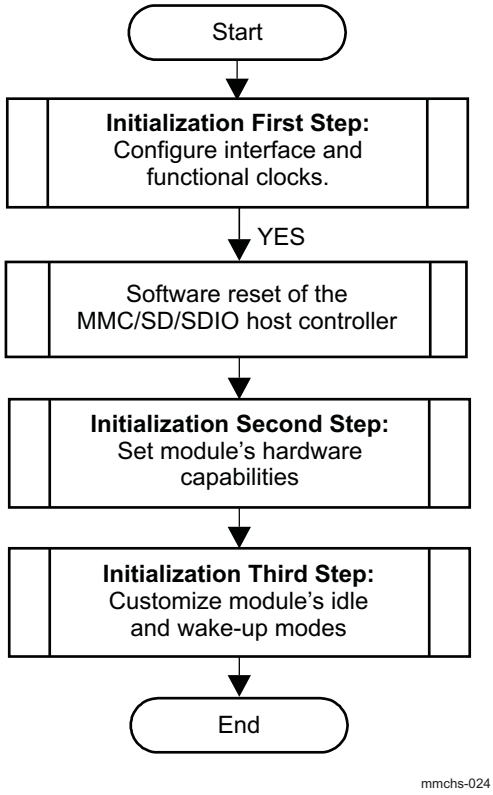
  Only the interrupt CIRQ is generated when CCS is detected.

19.5 MMC/SD/SDIO Basic Programming Model

19.5.1 MMC/SD/SDIO Host Controller Initialization Flow

Figure 19-23 shows the general boot process.

Figure 19-23. MMC/SD/SDIO Controller Meta Initialization Steps



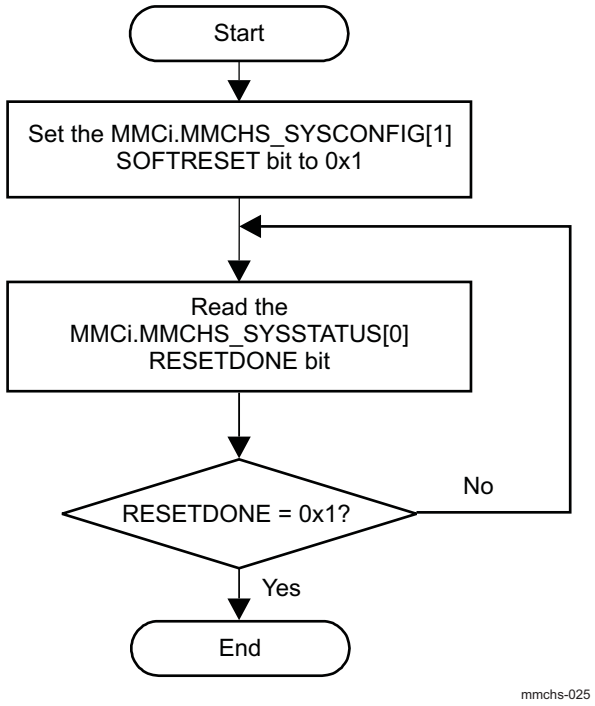
19.5.1.1 Enable Interface and Functional clock for MMC Controller

Prior to any MMCHS register access one must enable MMCHS interface clock and functional clock in PRCM module registers PRCM.CM_ICLKEN1_CORE and PRCM.CM_FCLKEN1_CORE. Please refer to , *Power, Reset, and Clock Management*.

19.5.1.2 MMCHS Soft Reset Flow

Figure 19-24 shows the soft reset process of MMCHS controller.

Figure 19-24. MMC/SD/SDIO Controller Software Reset Flow



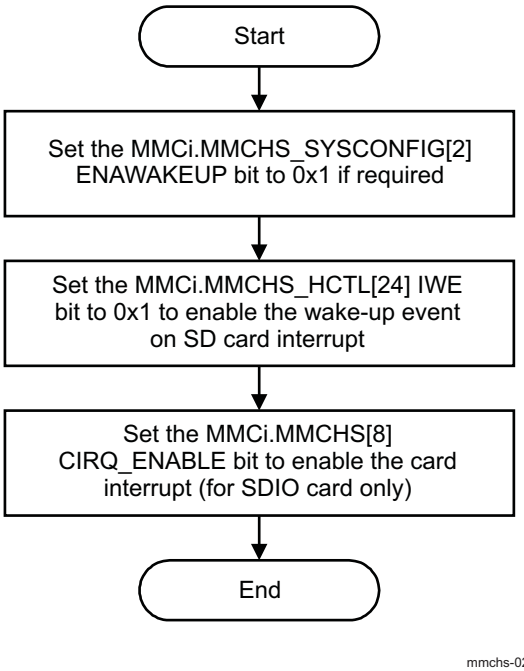
19.5.1.3 Set MMCHS Default Capabilities

Software must read capabilities (in boot ROM for instance) and is allowed to set (write) MMCi.MMCHS_CAPA[26:24] and MMCi.MMCHS_CUR_CAPA[23:0] registers before the MMC/SD/SDIO host driver is started.

19.5.1.4 Wake-Up Configuration

Figure 19-25 details MMCHS controller wake-up configuration.

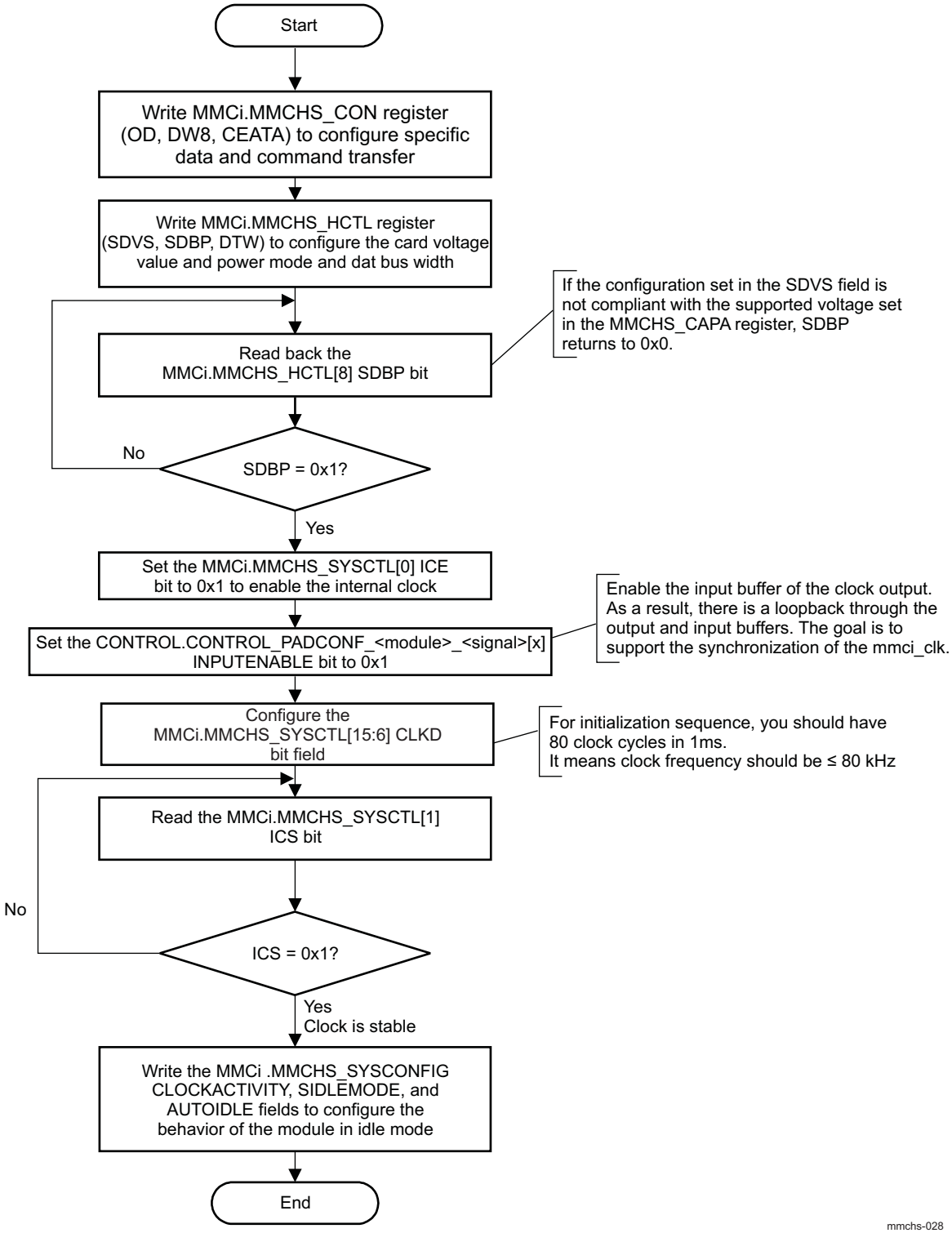
Figure 19-25. MMC/SD/SDIO Controller Wake-Up Configuration



19.5.1.5 MMC Host and Bus Configuration

Figure 19-26 details MMC bus configuration process.

Figure 19-26. MMC/SD/SDIO Controller Bus Configuration



19.5.2 Basic Operations for MMC/SD/SDIO Host Controller

The MMC/SD/SDIO host controller performs data transfers: data to card (referred to as write transfers) and data from card (referred to as read transfers).

The host controller requires transfers to run on a block-by-block basis, rather than on a DMA burst size basis. A single DMA request (or block request interrupt) is signaled for each block. Pipelining is supported as long as the block size is less than one half of the memory buffer size.

19.5.2.1 Card Detection, Identification, and Selection

Figure 19-27 and Figure 19-28 show the card identification and selection process.

Figure 19-27. MMC/SD/SDIO Controller Card Identification and Selection - Part 1

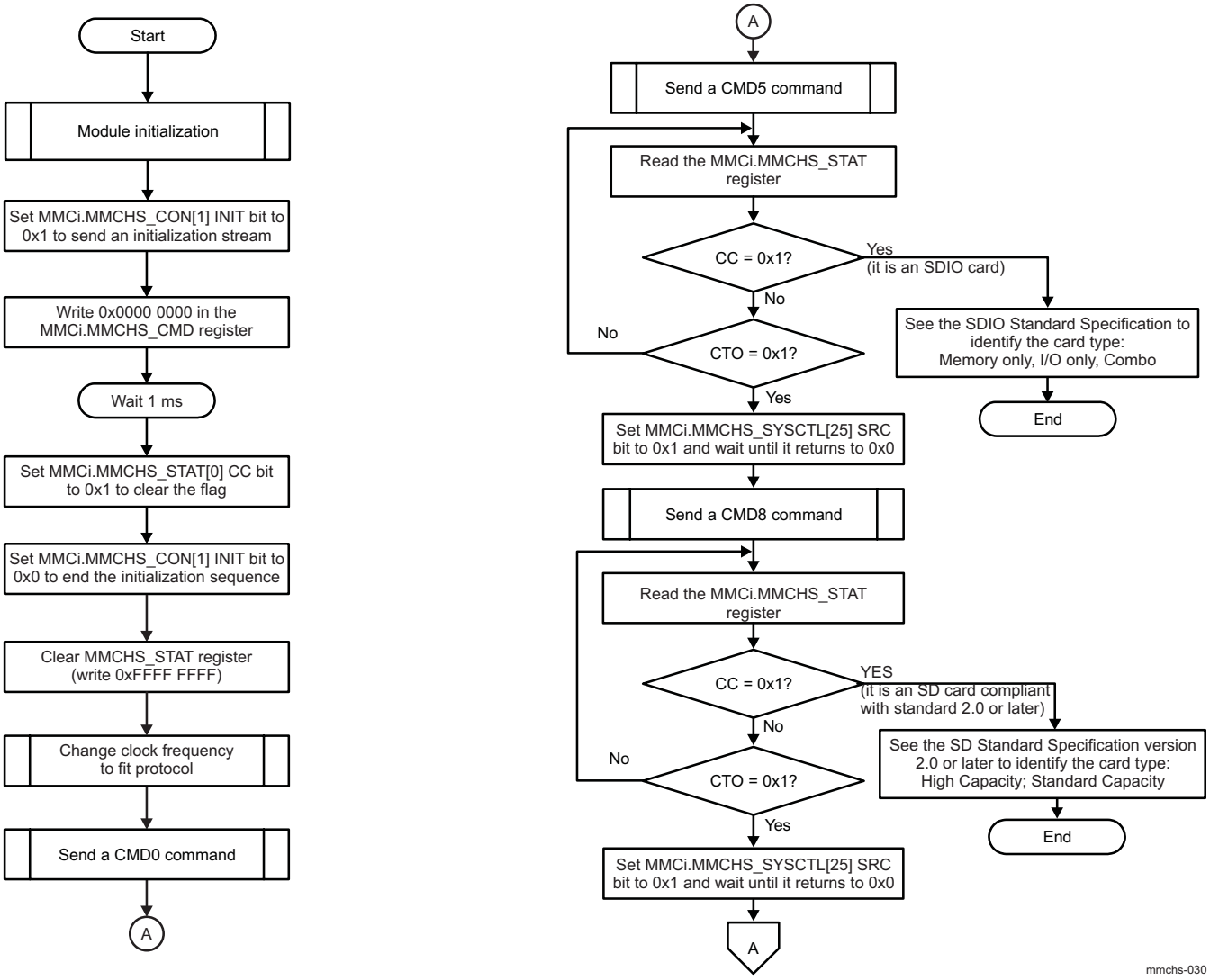
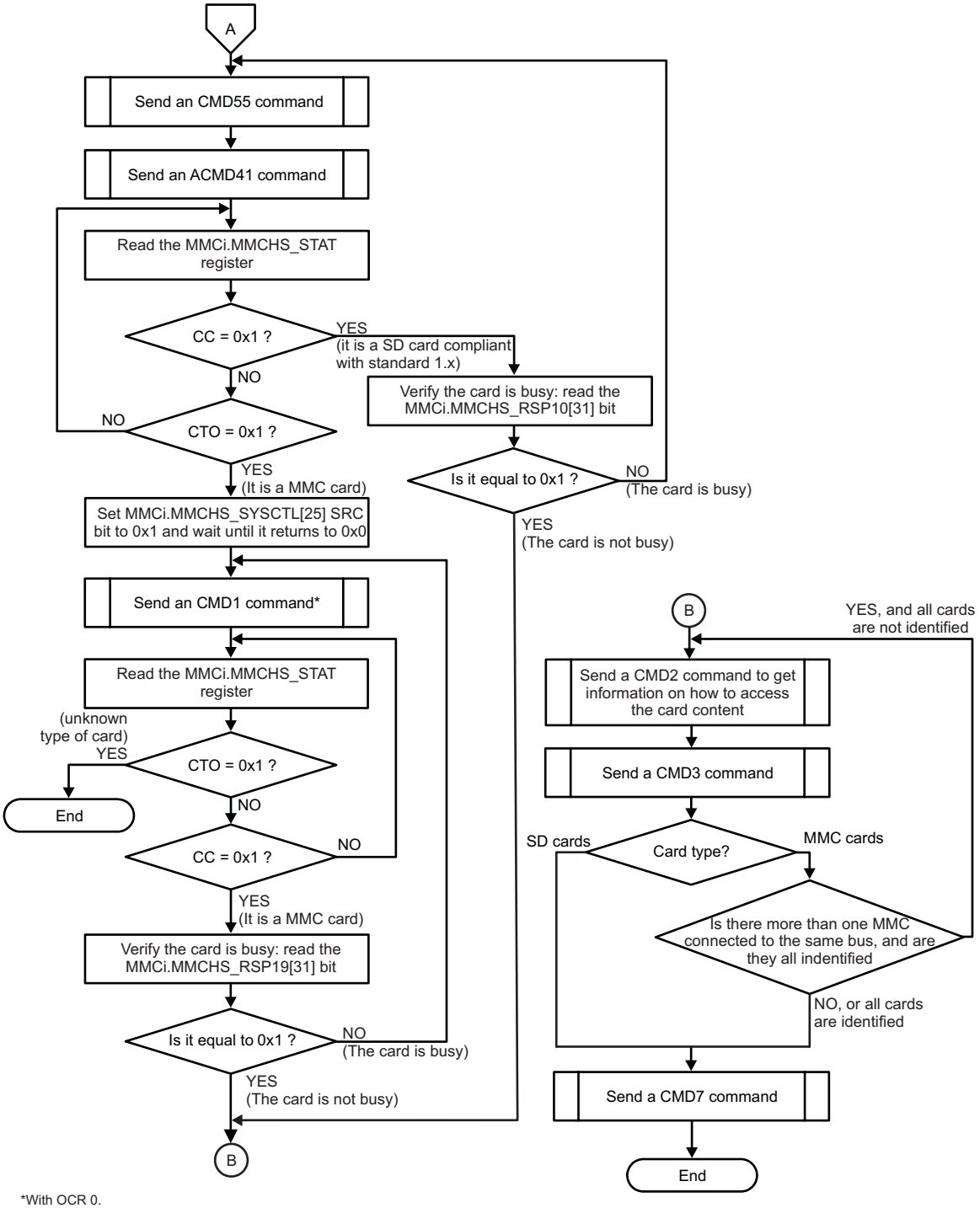


Figure 19-28. MMC/SD/SDIO Controller Card Identification and Selection - Part 2



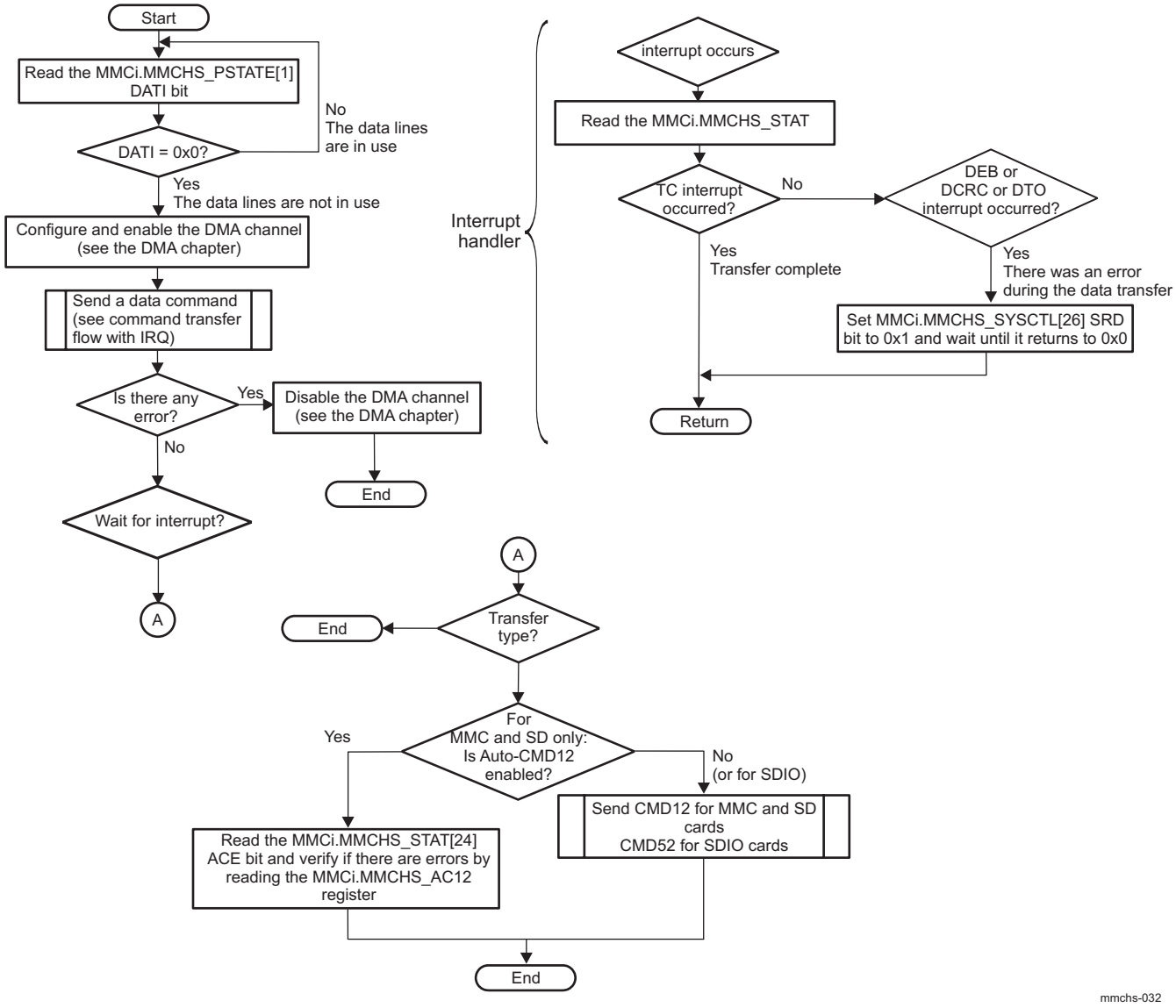
*With OCR 0.
In case of a CMD1 with OCR=0, a second CMD1 must be sent to the card with the "negociated" voltage.

108-031

19.5.2.2 Read/Write Transfer Flow in DMA Mode with Interrupt

Figure 19-29 describes the read and write protocol in DMA mode with interrupt signaling. Refer to Chapter 7, *DMA*, for more information on the DMA settings.

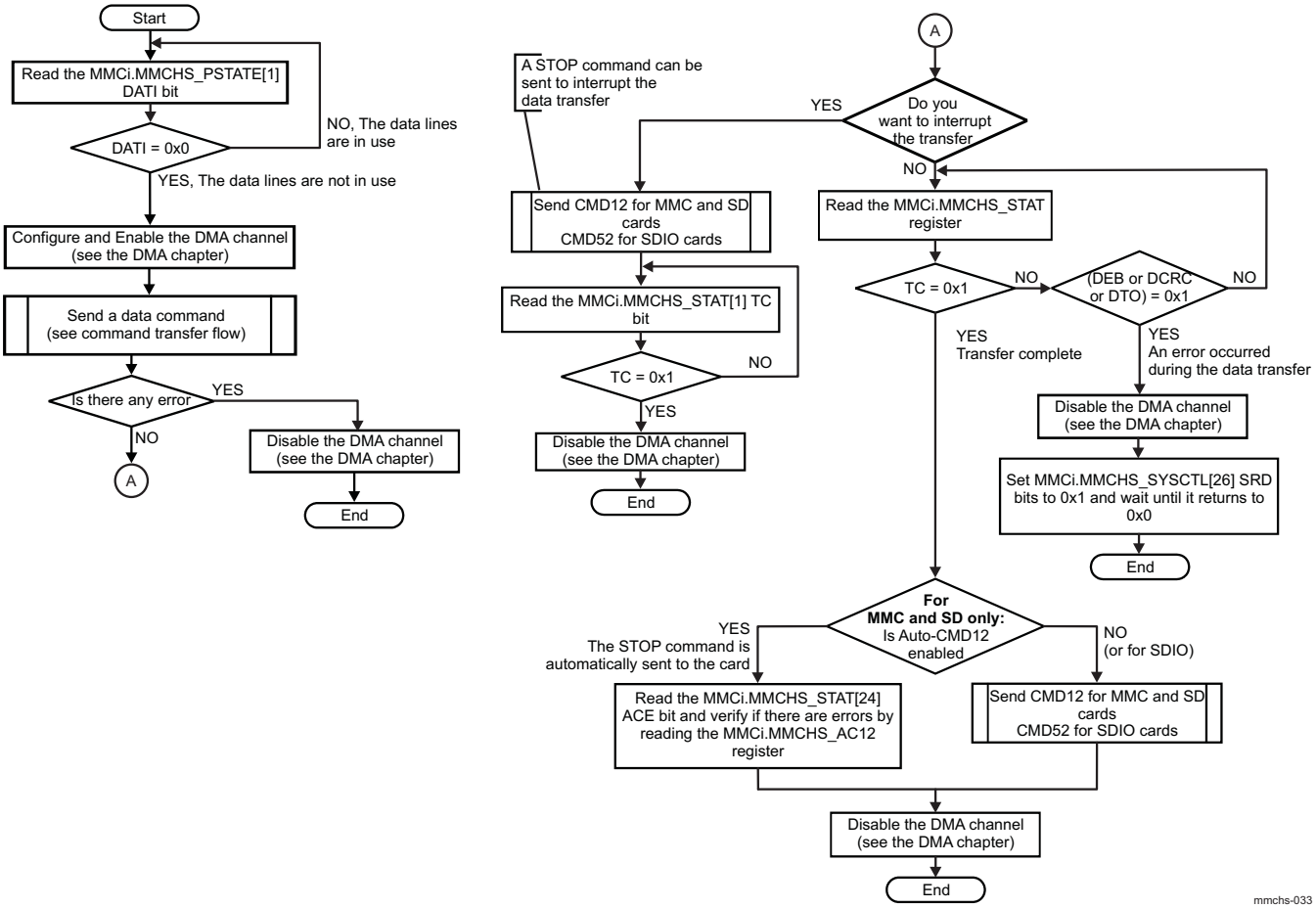
Figure 19-29. MMC/SD/SDIO Controller Read/Write Transfer Flow in DMA Mode with Interrupt



19.5.2.3 Read/Write Transfer Flow in DMA Mode with Polling

Figure 19-30 describes the read and write protocol in DMA mode. Refer to Chapter 7, *DMA*, for more information on the DMA settings.

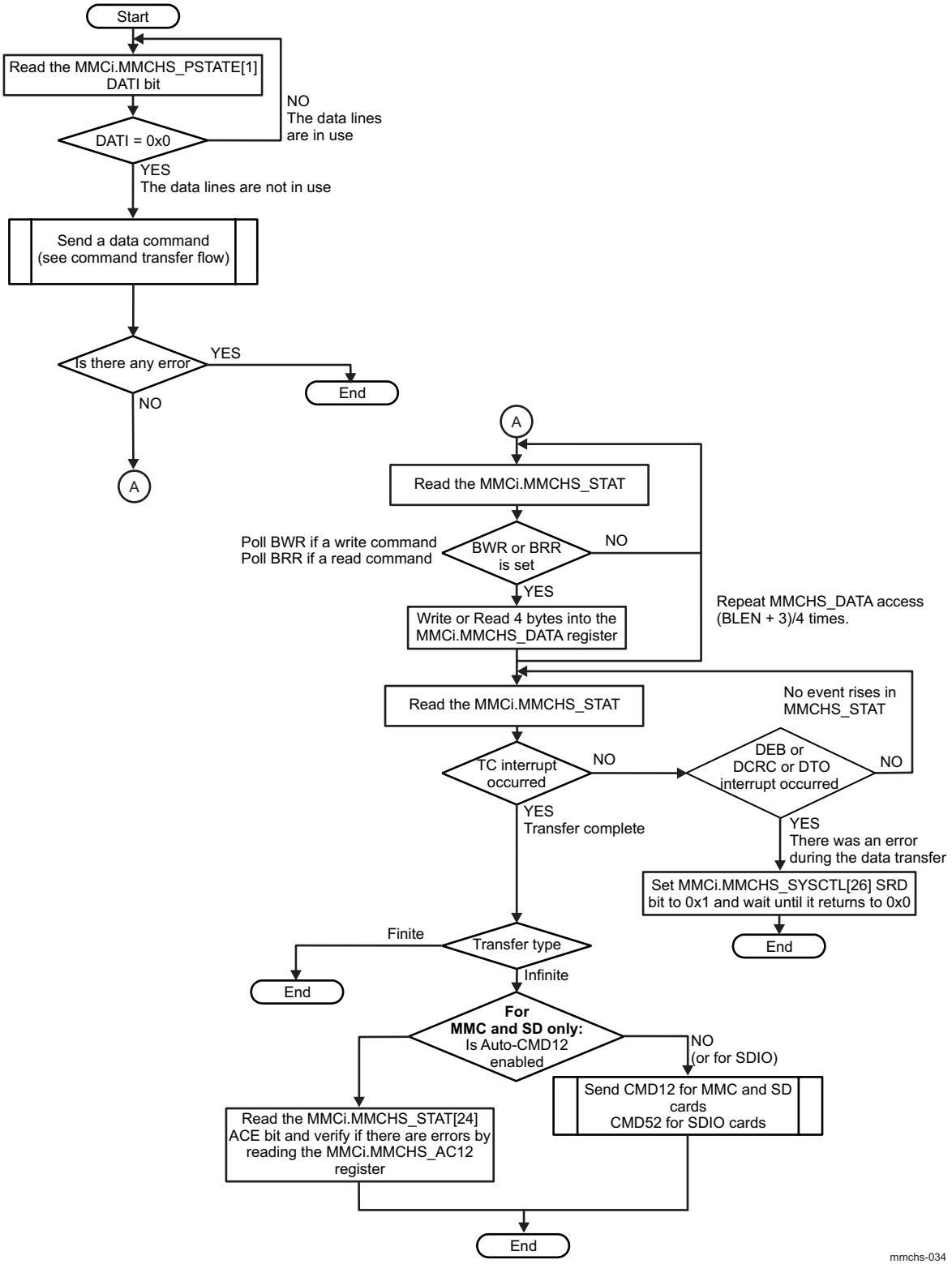
Figure 19-30. MMC/SD/SDIO Controller Read/Write Transfer Flow in DMA Mode with Polling



19.5.2.4 Read/Write Transfer Flow without DMA with Polling

Figure 19-31 describes a read/write transfer without using the DMA and with polling.

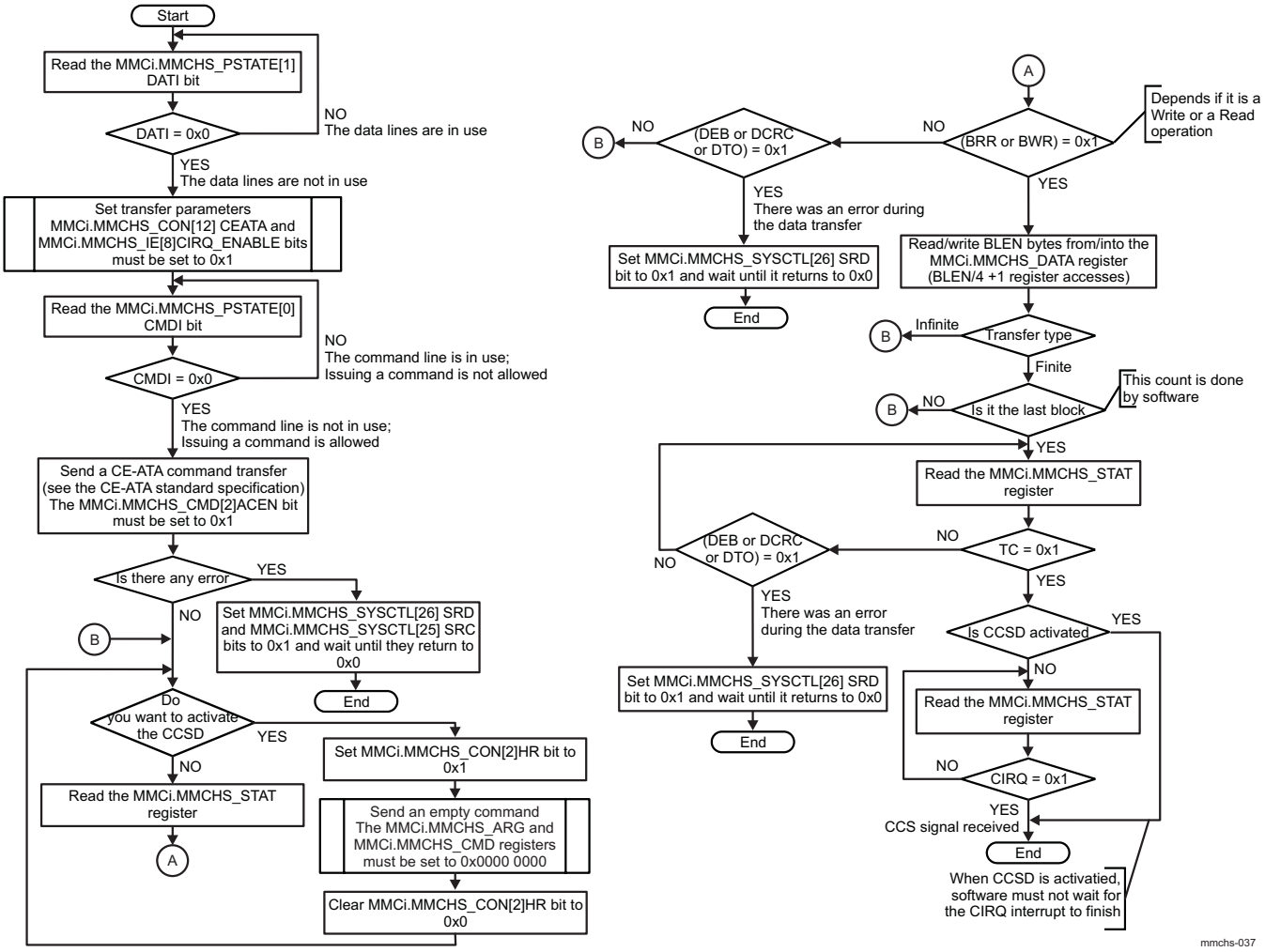
Figure 19-31. MMC/SD/SDIO Controller Read/Write Transfer Flow without DMA with Polling



19.5.2.5 Read/Write Transfer Flow in CE-ATA Mode

Figure 19-32 describes the read and write CE-ATA protocol when in Polling mode.

Figure 19-32. MMC/SD/SDIO Controller Read/Write in CE-ATA Mode



CAUTION

CE-ATA protocol is only supported by MMC cards.

In CE-ATA mode, issuing command during the transfer (except a CCSD command) is not allowed

In CE-ATA mode, infinite transfers are not allowed, only finite transfers are permitted.

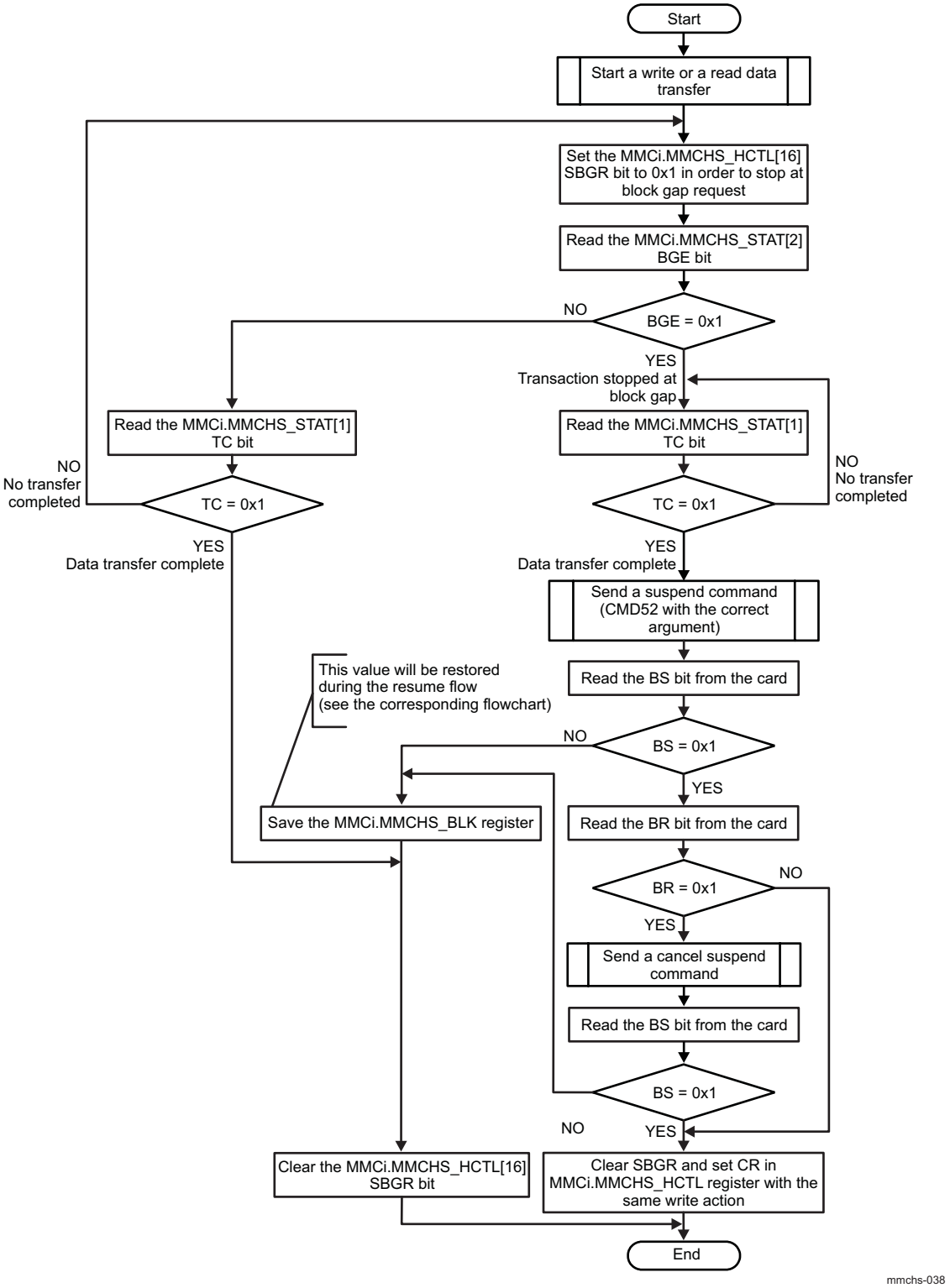
19.5.2.6 Suspend-Resume Flow

The suspend and resume feature is only supported by SDIO cards.

19.5.2.6.1 Suspend Flow

Figure 19-33 describes the suspend flow for SDIO cards.

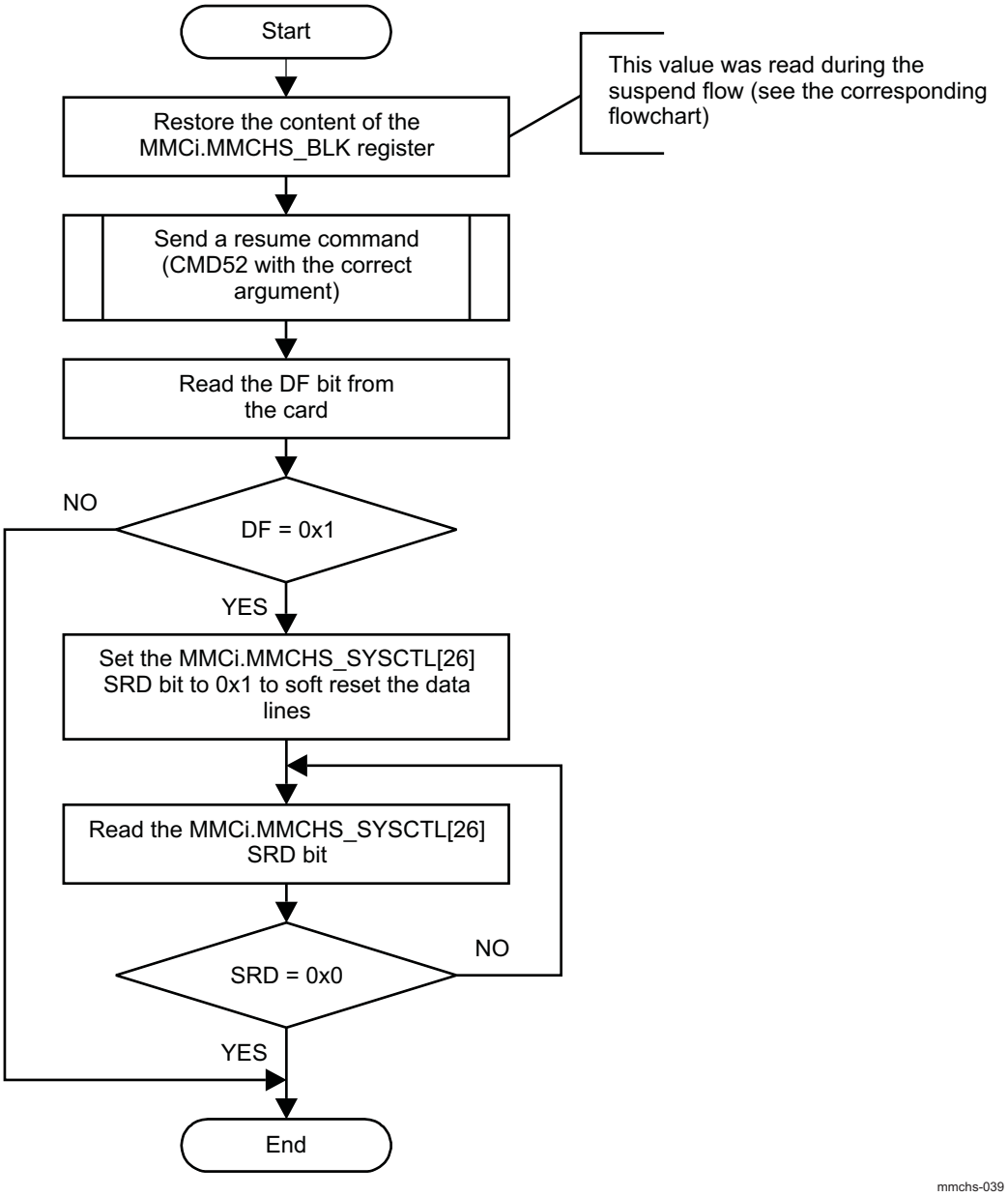
Figure 19-33. MMC/SD/SDIO Controller Suspend Flow



19.5.2.6.2 Resume Flow

Figure 19-34 describes the resume flow for SDIO cards.

Figure 19-34. MMC/SD/SDIO Controller Resume Flow



19.5.2.7 Basic Operations - Steps Detailed

19.5.2.7.1 Command Transfer Flow

Figure 19-35 describes how to send a command to the card using polling instead of interrupts for event signaling.

Figure 19-35. MMC/SD/SDIO Controller Command Transfer Flow with Polling

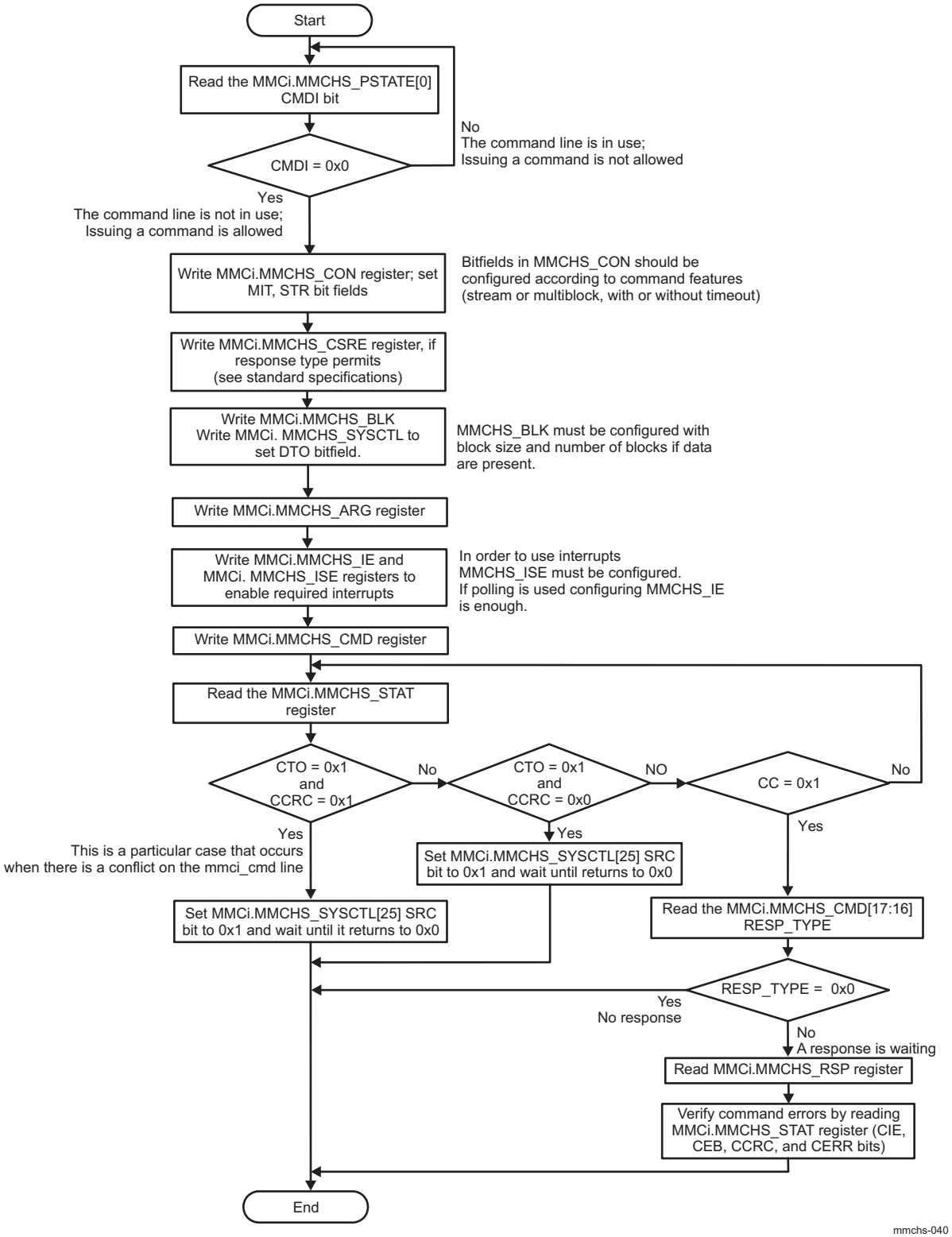
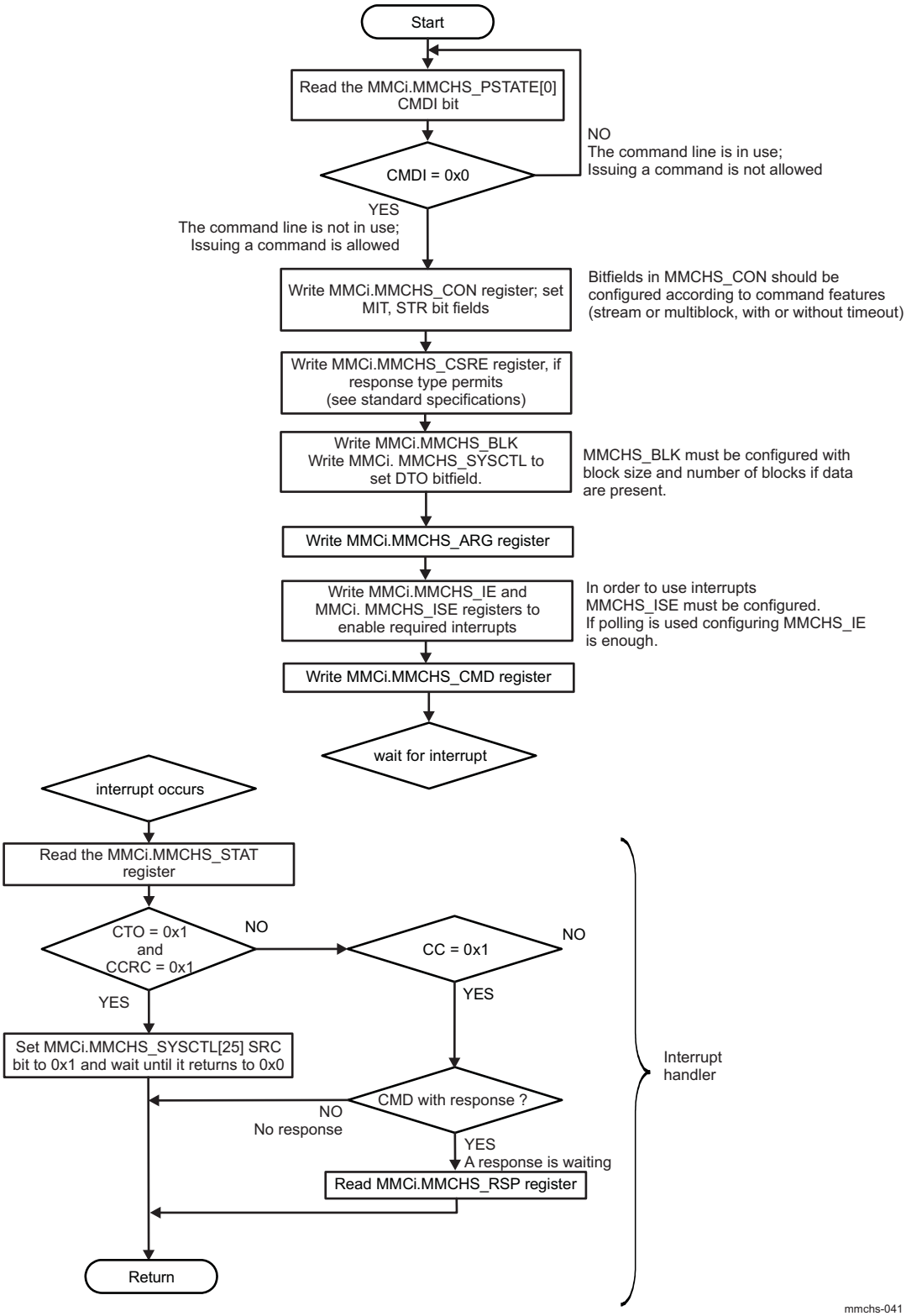


Figure 19-36 describes how to send a command to the card using interrupts for event signaling.

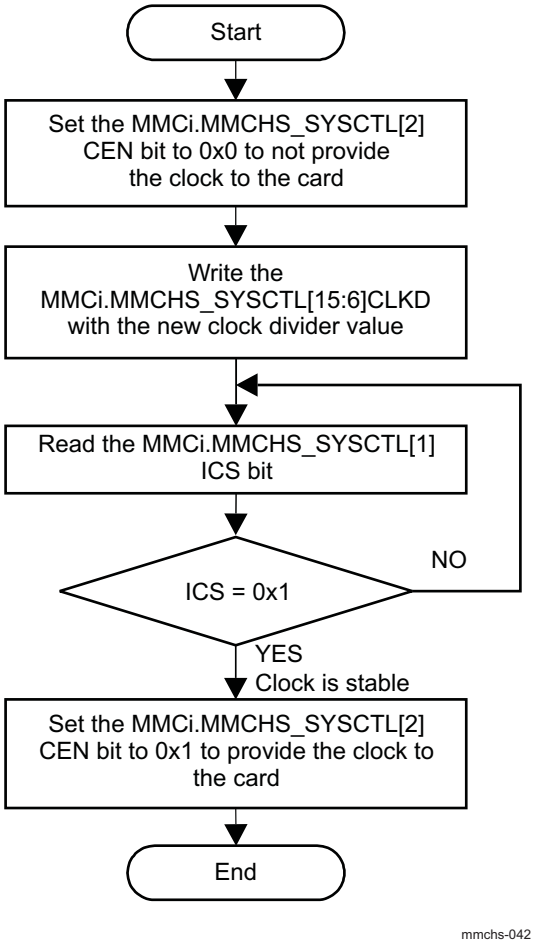
Figure 19-36. MMC/SD/SDIO Controller Command Transfer Flow with Interrupts



19.5.2.7.2 MMCHS Clock Frequency Change

Figure 19-37 describes the different steps that allow to change the MMC/SD/SDIO output clock frequency.

Figure 19-37. MMC/SD/SDIO Controller Clock Frequency Change Flow



19.5.3 MMC/SD/SDIO1 Bus Voltage Selection

The MMC/SD/SDIO1 controller can operate with two types of card voltages: 1.8 V and 3.0 V. For this reason, dual voltage pads are implemented on this interface. For technological concerns those pads must have an internal bias voltage reference to operate. The PBIAS_LITE module supplies this bias voltage, depending on the CONTROL.CONTROL_PBIAS_LITE register settings.

See , *Extended-Drain I/O Pin and PBIAS Cell*, for more information about the PBIAS_LITE cell.

, *Extended-Drain I/Os and PBIAS Cell Basic Programming Guide*, describes the steps involved in transitioning from 1.8 V to 3.0 V and from 3.0 V to 1.8 V, applicable to the MMC/SD/SDIO1 controller.

CAUTION

The BIAS voltage must be set using the procedure described in , *Extended-Drain I/Os and PBIAS Cell Basic Programming Guide*. Failure to follow this procedure can damage the MMCHS interface.

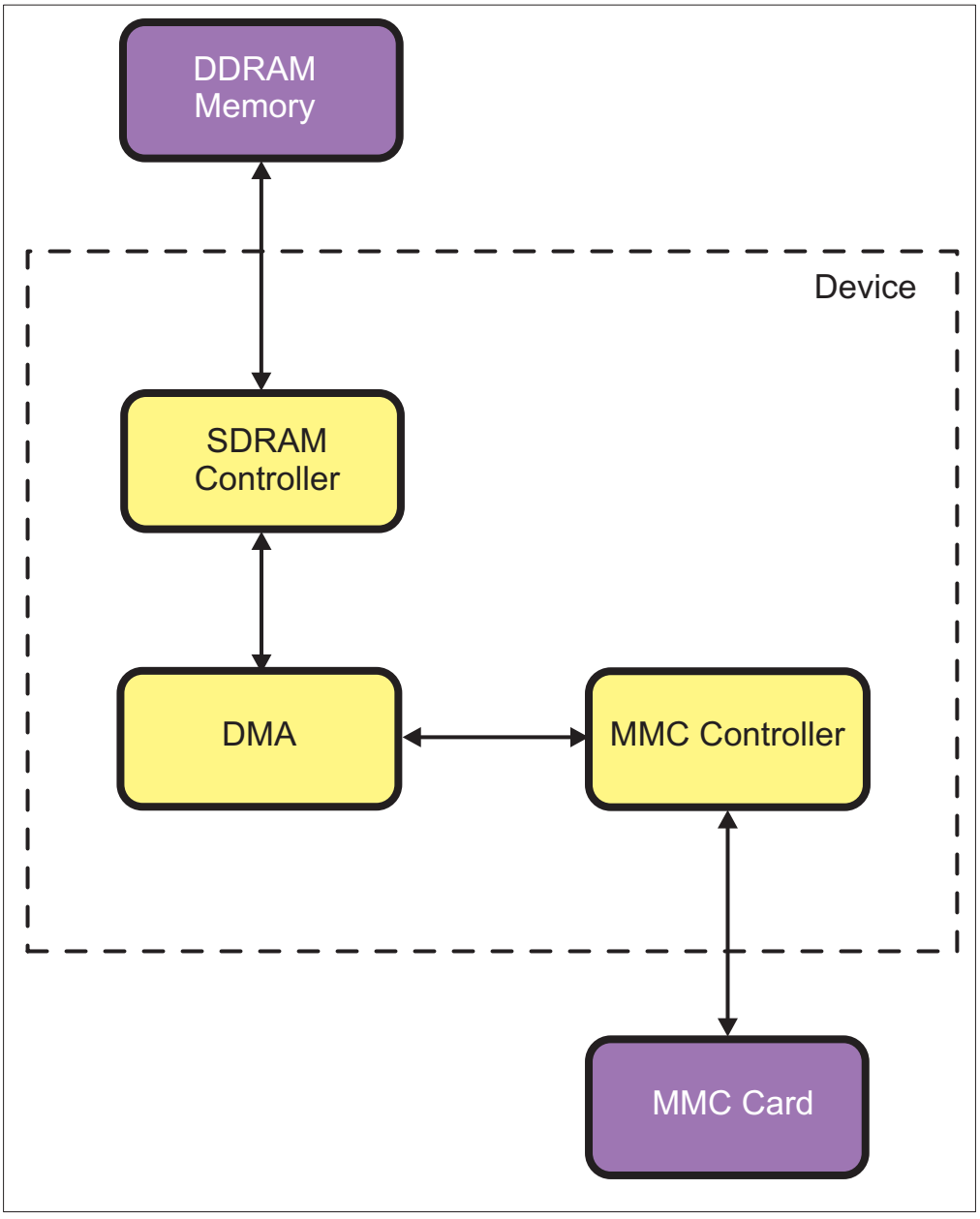
19.6 MMC/SD/SDIO Use Cases and Tips

19.6.1 MMCHS Controller Usage

19.6.1.1 Overview

The MMCHS controller is in charge of managing raw data storage in a MMC memory card. The MMCHS controller transfers data between DDRAM and MMC card. Figure 19-38 gives an overview of MMCHS controller position in the use case.

Figure 19-38. Overview



For the Camcorder use case, the MMCHS controller is configured to operate with the following features :

- High speed mode with a card clock frequency of 48 MHz.
- 8 data lines.

- 1.8 V and 3.0 V voltage capabilities.

MMC card power supply is not provided by the MMCHS controller itself but rather by the companion device. please refer to I2C TRM output to learn how to set these voltage levels.

Only MMC1 controller is used in this configuration. Hence this document describes the output of the test carried out with MMC1 controller.

19.6.1.2 Environment

In order to operate in the correct manner, the MMCHS controller needs the System Control Module (SCM) to be configured with the right muxing mode and with the right pull up state. The MMC card needs to be powered with the right power level and this is done with the companion device. Figure 19-39 gives the environment picture of the MMCHS controller.

Figure 19-39. Environment

19.6.1.2.1 Command and Data Transfer Formats

When communicating with a MMC card, The MMCHS controller is always the master. The communication between the MMCHS controller and the MMC card always starts by sending a command. Both command and data transfers are started with a command.

The data transfer type used by the MMCHS controller is a finite multiple block transfer. Figure 19-40 illustrates a command transfer without data. Figure 19-41 and Figure 19-42 illustrate a multiple block transfer between the MMCHS controller and the MMC card.

Figure 19-40. Command Transfer

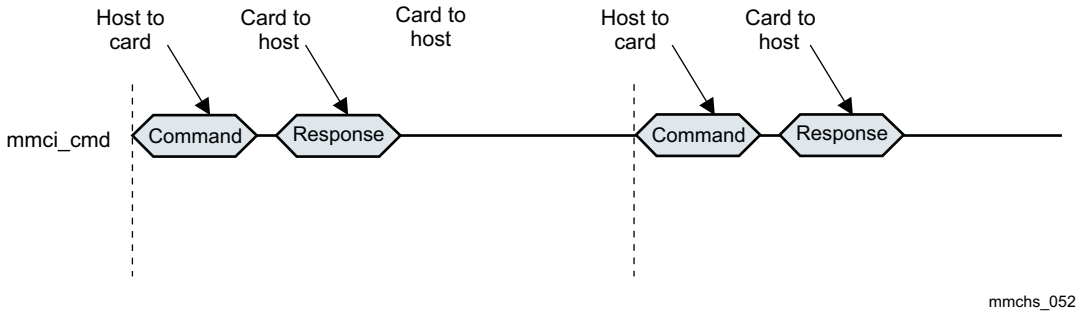


Figure 19-41. Data Read Transfer

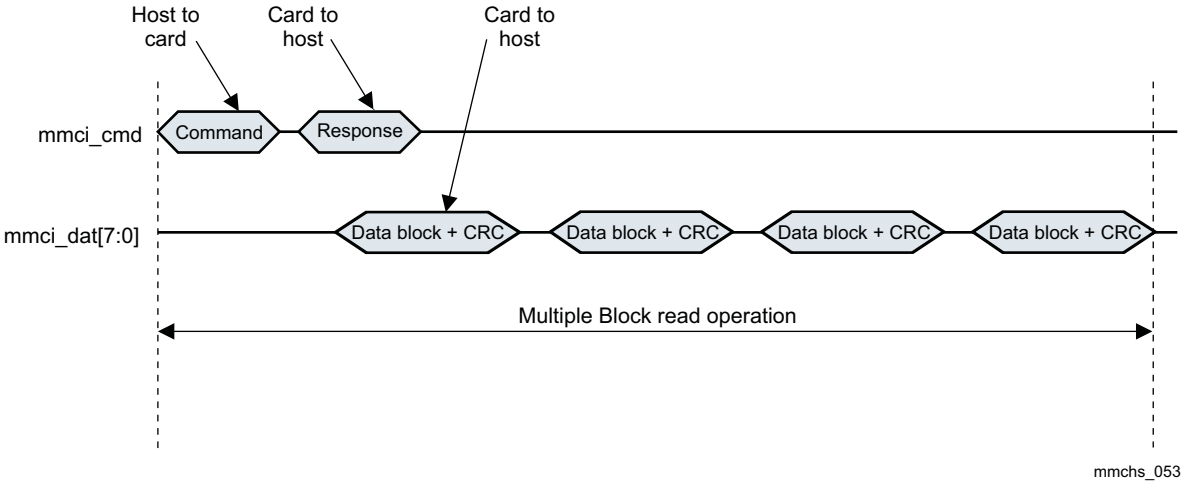
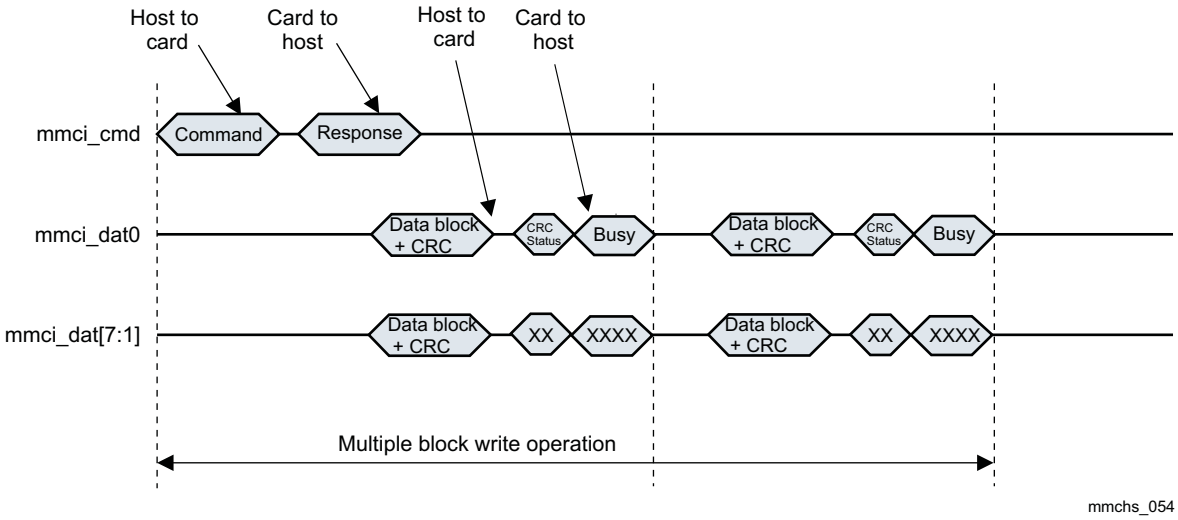


Figure 19-42. Data Write Transfer



19.6.1.3 Programming Flow

19.6.1.3.1 Initial Configuration

The initialization of the MMCHS controller is done through these steps :

1. MMCHS controller interface and functional clocks enabling.
2. MMCHS controller software reset.
3. MMCHS controller voltage capabilities initialization.
4. MMCHS controller default initialization.
5. MMCHS controller INIT procedure start.
6. MMCHS controller pre-card identification configuration.

   For more information about different steps the MMCHS controller goes through during initial configuration refer to Section 19.5, *MMC/SD/SDIO Basic Programming Model*.

19.6.1.3.1.1 MMCHS Controller Interface and Functional Clocks Enabling

To enable the interface and functional clocks of the MMCHS1 controller, the following steps must be done:

1. Enable the interface clock for the MMCHS1 controller (set the PRCM.CM_ICLKEN1_CORE[24] EN_MMCHS1).
2. Enable the functional clock for the MMCHS1 module (set the PRCM.CM_FCLKEN1_CORE[24] EN_MMCHS1).

Table 19-8 shows all PRCM registers to be configured to enable interface and functional clocks for MMCHS1 controller.

Table 19-8. Register Print for the MMCHS1 controller's clocks Initialization

Register Name	Register Address	Value	Value Description
PRCM.CM_ICLKEN1_CORE	0x4800 4A10	0x01000000	MMCHS1 interface clock enabled
PRCM.CM_FCLKEN1_CORE	0x4800 4A00	0x01000000	MMCHS1 functional clock enabled

19.6.1.3.1.2 MMCHS Controller Software Reset

In order to software reset the MMCHS1 controller, the following steps must be done:

1. Write 0x2 in MMCHS1.MMCHS_SYSCONFIG register.
2. Wait until MMCHS1.MMCHS_SYSSTATUS[0] RESETDONE turns 1.

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_SYSCONFIG	0x4809 C010	0x00000002	Activate software reset
MMCHS1.MMCHS_SYSSTATUS	0x4809 C014	0x00000001	Reset is over.

19.6.1.3.1.3 MMCHS Controller Voltage Capabilities Initialization

MMCHS1 controller's voltage capabilities should be set in MMCHS1.MMCHS_CAPA. Refer to Table 19-9.

Table 19-9. MMCHS Controller Voltage Capabilities Initialization

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CAPA	0x4809 C140	current_value | 0x06000000	Activate VS18 and VS30 in MMCHS_CAPA register.

19.6.1.3.1.4 MMCHS Controller Default Initialization

Before sending any command, the MMCHS controller is configured with default values :

1. Default voltage support is set to 1.8 V in MMCHS1.MMCHS_HCTL[11:9] SDVS.
2. MMC bus is set to open drain in MMCHS1.MMCHS_CON[0] OD.
3. MMC data bus width is set to 1 in MMCHS1.MMCHS_HCTL[1] DTW.
4. MMC Card's power is off.
5. MMC Card's clock is on in MMCHS1.MMCHS_SYSCTL[0] ICE and MMCHS1.MMCHS_SYSCTL[2] CEN.
6. MMCHS controller bus power up in MMCHS1.MMCHS_HCTL[8] SDBP.
7. MMC card's clock frequency is set to 150 KHz in MMCHS1.MMCHS_SYSCTL[15:6] CLKD.

Table 19-10 shows the values that should be written in the right register pool.

Table 19-10. MMC Controller Default Initialization Values

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_HCTL	0x4809C128	0x00000b00	data bus width = 1, voltage = 1.8v, MMC bus power is on (not card's power)
MMCHS1.MMCHS_SYSCTL	0x4809C12C	0x0000a007	card's clock enable and card's clock frequency divider.
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	Set MMC bus mode to open drain.

A small notice about MMCHS1.MMCHS_HCTL. Even if the value written in it is 0x00000b00, the value read from it is equal to 0x00000a00. This is due to the fact that the MMCHS controller in not in a card state mode which sets automatically MMCHS_HCTL[8] SDBP bit to 0.

19.6.1.3.1.5 MMCHS Controller INIT Procedure Start

Prior to issuing any command, the MMCHS controller has to execute a special INIT procedure. The MMCHS controller has to generate a clock during 1ms. During the INIT procedure, the MMCHS controller generates 80 clock periods. In order to keep the 1ms gap, the MMCHS controller should be configured to generate a clock whose frequency is smaller or equal to 80 KHz.

The INIt procedure is executed by setting MMCHS1.MMCHS_CON[1] INIT bit field to 1 and by sending a dummy command, writing 0x00000000 in MMCHS1.MMCHS_CMD register.

To assure the synchronous clock on the clock output of the MMC1 controller, set the CONTROL.CONTROL_PADCONF_MMC1_CLK[8] bit. In this way, the clock output is also configured as input and the result is looped back through the output and input buffers of the pad.

Table 19-11 shows the values that should be written in the right register pool.

Table 19-11. MMCHS Controller INIT Procedure Start

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	current_value | 0x00000002	Sets MMCHS1.MMCHS_CON[1] INIT to 1
MMCHS1.MMCHS_CMD	0x4809C10C	0x00000000	Sends dummy command.
CONTROL.CONTROL_PADCONF_MMC1_CLK	0x4800 2144	0x100	Set the INPUTENABLE bit to assure synchronous clock.

19.6.1.3.1.6 MMCHS Controller Pre-card Identification Configuration

Before card identification starts, the MMCHS controller's configuration should change. MMC card's clock should now be 400 KHz according to MMC system spec requirements. Table 19-12 shows the values that should be written in the right register pool.

Table 19-12. MMCHS Controller Pre-Card Identification Configuration

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_HCTL	0x4809C128	0x00000b00	Data bus width = 1, voltage = 1.8 V, MMC bus power is on (not card's power)
MMCHS1.MMCHS_SYSCTL	0x4809C12C	0x00003C07	Card's clock enable and card's clock frequency divider.
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	Set MMC bus mode to open drain.

19.6.1.3.2 MMC Card Identification

MMC card identification is performed by issuing many MMC commands. Each command imposes certain configuration values in a pool of registers. The status of command transfer is read in MMCHS1.MMCHS_STAT register and command response if any is read from MMCHS1.MMCHS_RSP10, MMCHS1.MMCHS_RSP32, MMCHS1.MMCHS_RSP54 and MMCHS1.MMCHS_RSP76.

This TRM output describes a use case where interrupts are used to signal MMCHS controller status changes.

For more details about the card identification sequence, please refer to Section 19.5 *MMC/SD/SDIO Basic Programming Model*.

19.6.1.3.2.1 Sending CMD0

This command resets the MMC card (see Table 19-13).

Table 19-13. Sending CMD0

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x00040001	Enables CC and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x00040001	Enables CC and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x00000000	Sends CMD0 whose opcode is 0.

19.6.1.3.2.2 Sending CMD5

This command asks a SDIO card to send its operating conditions (see Table 19-14). This command will fail if there is no SDIO card. In case of success the response will be in MMCHS1.MMCHS_RSP10 register.

Table 19-14. Sending CMD5

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x00050001	Enables CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x00050001	Enables CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x05020000	Sends CMD5 whose opcode is 5 and response type is 48 bits.

19.6.1.3.2.2.1 Sending CMD8

This command asks a SD card version 2.X to send its operating conditions (see Table 19-15). This command will fail if there is no SD card verison 2.X. In case of success the response will be in MMCHS1.MMCHS_RSP10 register.

Table 19-15. Sending CMD8

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x81a0000	Sends CMD8 whose opcode is 8, response type is 48 bits with CICE and CCCE enabled.

19.6.1.3.2.3 Sending CMD55

This is a special command used to prevent the card that the following command is going to be an application one (see Table 19-16). This is used to prepare the issuing of ACMD41 (opcode = 41) that usually asks a SD card version 1.X to send its operating conditions. If no SD card version 1.X is connected to the MMCHS controller this command will fail. In case of success, the response will be received in MMCHS1.MMCHS_RSP10 register.

Table 19-16. Sending CMD55

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x371a0000	Sends CMD55 whose opcode is 55, response type is 48 bits with CICE and CCCE enabled.

19.6.1.3.2.4 Sending CMD1

Once the card response is available in register MMCHS1.MMCHS_RSP10, the software is responsible to compare Card OCR and Host OCR, and then send a second CMD1 command with the cross-checked OCR.

This way, the card is notified of the Operating Voltage to work with.

Table 19-17. Sending CMD1

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x00050001	Enables CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x00050001	Enables CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x01020000	Sends CMD1 whose opcode is 1 and response type is 48 bits.

Once the card response is available in register MMCHS1.MMCHS_RSP10, the software should compare card OCR and host OCR, and then send a second CMD1 command with the cross-checked OCR. This tells the card what operating voltage to use.

19.6.1.3.2.5 Sending CMD2

This command asks the MMC card to send its CID register's content (see Table 19-18). The response is 128 bits wide and is received in MMCHS1.MMCHS_RSP10, MMCHS1.MMCHS_RSP32, MMCHS1.MMCHS_RSP54 and MMCHS1.MMCHS_RSP76 registers.

Table 19-18. Sending CMD2

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x00070001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x00070001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x02090000	Sends CMD2 whose opcode is 2, response type is 136 bits with CCCE enabled.

19.6.1.3.2.6 Sending CMD3

This command sets MMC card address (see Table 19-19). Useful when MMCHS controller switches to addressed mode.

Table 19-19. Sending CMD3

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000001	MMC bus is still in open drain state for broadcast.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.

Table 19-19. Sending CMD3 (continued)

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CMD	0x4809C10C	0x031a0000	Sends CMD3 whose opcode is 3, response type is 48 bits with CICE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x00010000	MMCHS_ARG register carries MMC card's address. We choose to assign address 1 any other 16 bit wide value is valid.

19.6.1.3.3 MMC Bus Setting Change After Card Identification

After CMD3 command transfer is completed successfully, an auto-negotiation on voltage value an start. This is the frontier when the MMCHS controller should switch from identification mode to transfer mode. This impacts the controller in a way that it should change its bus state from open drain to push-pull. Table 19-20 gives several registers and their values.

Table 19-20. MMC Bus Setting Change Table

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000000	Bus is now in push-pull mode.
MMCHS1.MMCHS_HCTL	0x4809C128	0x00000B00	Bus power is on, 1.8 V is selected.
MMCHS1.MMCHS_SYSCTL	0x4809C12C	0x00003C07	MMCHS controller's internal clock is stable and enabled, MMC card's clock is on. Divider value is 240 which means that MMCHS controller is still supplying a 400 KHz clock.

19.6.1.3.4 Reading the CSD Register of a MMC Card

After settling on a voltage value, additional information must be read from the MMC card. This data is stored in MMC card CSD register. The card sends CSD register content after receiving CMD9 command. The CSD register holds important information on the card, MMC system specification version support, maximum clock speed support, memory capacity, minimum block length, read and write transfer latency timings.

19.6.1.3.4.1 Sending CMD9

This command asks the card to send its csd register's content (see Table 19-21). The 136 bit (128 bits are valid payload) response is received in MMCHS1.MMCHS_RSP10, MMCHS1.MMCHS_RSP32, MMCHS1.MMCHS_RSP54 and MMCHS1.MMCHS_RSP76 registers. CMD9 is an addressed command which means that card's address must be written in MMCHS1.MMCHS_ARG register before the command is issued.

Table 19-21. Sending CMD9

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000000	MMC bus is in push-pull mode.
MMCHS1.MMCHS_IE	0x4809C134	0x00070001	Enables CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x00070001	Enables CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x09090000	Sends CMD9 whose opcode is 9, response type is 136 bits with CCCE enabled.

Table 19-21. Sending CMD9 (continued)

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_ARG	0x4809C108	0x00010000	MMCHS_ARG register carries MMC card's address. We choose to assign address 1 any other 16 bit wide value is valid.

After receiving and parsing CMD9 response, MMC card clock speed must change to take advantage to card's maximum speed. This has an impact on MMC bus as we should perform a clock frequency change. At this stage the maximum clock frequency will be 20 MHz (please refer to MMC system specification from www.mmca.org).

Clock frequency change procedure is performed in several steps. Please refer to Section 19.5, MMC/SD/SDIO Basic Programming Model, Section 19.5.2.7.2, *MMCHS Clock Frequency Change*.

Table 19-22 shows the value written in MMCHS1.MMCHS_SYSCTL register.

Table 19-22. MMCHS_SYSCTL Value

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_SYSCTL	0x4809C12C	0x00000147	MMCHS controller's internal clock is stable and enabled, MMC card's clock is on. Divider value is 5 which means that MMCHS controller is supplying a 19.2 MHz clock < 20 MHz.

Another important parameter read from CSD register is MMC system specification version. If this parameter points to a value greater than or equal to 4, the MMC card is capable of a speed up to 52 MHz and a bus width up to 8 (1, 4 or 8 are the possible options). In order to enable these two extra features, MMCHS controller must issue a CMD6 command with specific argument.

A CMD6 command is issued in the data transfer mode after MMC card is selected. MMC card selection consists of sending CMD7 command with MMC card's address in command argument.

Table 19-23 shows the set of register impacted by CMD7 issue action.

Table 19-23. Sending CMD7

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000000	MMC bus is in push-pull mode.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x071a0000	Sends CMD7 whose opcode is 7, response type is 48 bits with CICE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x00010000	MMCHS_ARG register carries MMC card's address. We choose to assign address 1 any other 16 bit wide value is valid.

After a CMD7 transfer is complete, the MMC card is ready to receive a CMD6 command. CMD6 command is used to write a byte in MMC card extended CSD register (ext_csd). It is an IO access function. There are two write actions, the first one enables a specific data bus width in the card. For our use case we used maximum data bus width 8. The second one enables high speed feature in the card.

19.6.1.3.4.1.1 Setting Data Bus Width to 8

Table 19-24 shows the set of registers impacted by issuing CMD6.

Table 19-24. Setting Data Bus Width with CMD6

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000000	MMC bus is in push-pull mode.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x061b0000	Sends CMD6 whose opcode is 6, response type is 48 bits with busy, with CICE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x03b70200	(3 << 24) | (byte_address << 16) | (byte_value << 8). byte_address is the byte address in ext_csd register.

After issuing CMD6 completes successfully and MMC card leaves busy state, MMCHS controller should change its data bus width. This is done by changing MMCHS1.MMCHS_CON configuration value.

Table 19-25. MMCHS_CON Value

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000020	MMCHS controller's data bus width is set to 8.

19.6.1.3.4.1.2 Enable High Speed Feature

Table 19-26 shows the set of registers impacted by issuing CMD6 issuing.

Table 19-26. Enabling High Speed with CMD6

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000020	MMC bus is in push-pull mode. DW8 is enabled.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x061b0000	Sends CMD6 whose opcode is 6, response type is 48 bits with busy, with CICE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x03b90100	(3 << 24) | (byte_address << 16) | (byte_value << 8). byte_address is the byte address in ext_csd register.

After issuing CMD6 completes successfully and MMC card leaves busy state, MMCHS controller should now change its output clock to bring it to 48 MHz. 52 MHz, max frequency value supported by MMC card version 4 and above, is not supported because 96 MHz, MMCHS controller functional clock, is not a multiple of 52 MHz. We fall off to 48 MHz.

Table 19-27. MMCHS_SYSCTL Value

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_SYSCTL	0x4809C12C	0x00000087	MMCHS controller's internal clock is stable and enabled, MMC card's clock is on. Divider value is 2 which means that MMCHS controller is supplying a 48 MHz clock.

19.6.1.3.5 MMC Write Transfer

Either data read or data write transfer uses DMA controller to perform memory (DDRAM) to/from MMCHS controller transfers. The DMA part is not described in this chapter, it is described in Chapter 7, *DMA*.

Before any data transfer begins, the card must selected by issuing CMD7 command (Table 19-23).

A write transfer is a finite multiple block write transfer. In order to perform a write transfer, the following steps must performed.

19.6.1.3.5.1 Send CMD16

Issuing CMD16 allows to set the block length. For our use case we decided to use a static block length of 512 bytes. The block length value is passed to the MMC card via MMCHS1.MMCHS_ARG register. The registers impacted by this operation are as follows:

Table 19-28. Setting Block Length

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000020	MMC bus is in push-pull mode. DW8 is enabled.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x101a0000	Sends CMD16 whose opcode is 16, response type is 48 bits, with CICE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x00000200	Block length is 512 = 0x200

19.6.1.3.5.2 Send CMD23

Issuing CMD23 allows to set the number of how many 512-byte blocks the MMC card should expect from the MMCHS controller. The number of blocks is passed to MMC card via MMCHS1.MMCHS_ARG register. The registers impacted by this operation are as follows:

Table 19-29. Setting Number of Blocks

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000020	MMC bus is in push-pull mode. DW8 is enabled.
MMCHS1.MMCHS_IE	0x4809C134	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB events to occur.

Table 19-29. Setting Number of Blocks (continued)

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_ISE	0x4809C138	0x100f0001	Enables CERR, CIE, CCRC, CC, CTO and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x171a0000	Sends CMD23 whose opcode is 23, response type is 48 bits, with CICE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x00000008	Number of 512-byte blocks in 4 KB buffer is 8.

19.6.1.3.5.3 Send CMD25

Issuing CMD25 starts the finite, multiple block write transfer. Before the transfer starts, DMA controller should be configured for this operation. For more details about DMA configuration please refer to Chapter 7, *DMA*.

Table 19-30. CMD25 Issuing

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000020	MMC bus is in push-pull mode. DW8 is enabled.
MMCHS1.MMCHS_IE	0x4809C134	0x107f0013	Enables CERR, CIE, CCRC, CC, TC, BWR, CTO, DTO, DCRC, DEB and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x107f0013	Enables CERR, CIE, CCRC, CC, TC, BWR, CTO, DTO, DCRC, DEB and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x193a0023	Sends CMD25 whose opcode is 25, response type is 48 bits, with CICE, DP, MSBS, BCE, DE and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x00000000	Not used.
MMCHS1.MMCHS_BLK	0x4809C104	0x00080200	(number_blocks << 16) | (block_length)

19.6.1.3.6 MMC Read Transfer

Either data read or data write transfer uses DMA controller to perform memory (DDRAM) to/from MMCHS controller transfers. The DMA part is not described in this document, it is described in Chapter 7, *DMA*.

Before any data transfer begins, the card must selected by issuing CMD7 command (Table 19-23).

A read transfer is a finite multiple block write transfer. In order to perform a read transfer, the following steps must performed.

19.6.1.3.6.1 Send CMD16

Issuing CMD16 allows to set the block length. For our use case we decided to use a static block length of 512 bytes. The block length value is passed to the MMC card via MMCHS1.MMCHS_ARG register. See Table 19-28.

19.6.1.3.6.2 Send CMD23

Issuing CMD23 allows to set the number of how many 512-byte blocks the MMC card should expect from the MMCHS controller. The number of blocks is passed to MMC card via MMCHS1.MMCHS_ARG register. See Table 19-29.

19.6.1.3.6.3 Send CMD18

Issuing CMD18 starts the finite, multiple block read transfer. Before the transfer starts, DMA controller should be configured for this operation. For more details about DMA configuration please refer to Chapter 7, *DMA*.

Table 19-31. CMD18 Issuing

Register Name	Register Address	Value	Value Description
MMCHS1.MMCHS_CON	0x4809C02C	0x00000020	MMC bus is in push-pull mode. DW8 is enabled.
MMCHS1.MMCHS_IE	0x4809C134	0x107f0023	Enables CERR, CIE, CCRC, CC, TC, BRR, CTO, DTO, DCRC, DEB and CEB events to occur.
MMCHS1.MMCHS_ISE	0x4809C138	0x107f0023	Enables CERR, CIE, CCRC, CC, TC, BRR, CTO, DTO, DCRC, DEB and CEB interrupts to rise.
MMCHS1.MMCHS_CMD	0x4809C10C	0x123a0033	Sends CMD18 whose opcode is 18, response type is 48 bits, with CICE, DP, MSBS, BCE, DE, DDIR and CCCE enabled.
MMCHS1.MMCHS_ARG	0x4809C108	0x00000000	Not used.
MMCHS1.MMCHS_BLK	0x4809C104	0x00080200	(number_blocks << 16) | (block_length)

19.6.1.3.7 Dealing with High Capacity Cards

Unlike standard-capacity cards, memory addressing mode for high-capacity cards (SDHC and HC MMC) is in 512-byte block format instead of byte format. This means that byte and partial accesses are not allowed with high-capacity cards.

The 32-bit wide argument configured in the MMCHS_ARG register and sent in data transfer commands CMD17, CMD18, CMD24, and CMD25, carries the block index where the read/write should start and not the 32-bit byte address.

For high-capacity cards, the block size is fixed at 512 bytes. Any data transfer, when the data bus is involved, must be a multiple of 512 bytes.

The number of blocks for a high-capacity card, which determines the capacity of the card block_count x 512 bytes, is accessible through field C_SIZE in the card s CSD register version 2.0 for SD cards, or through the SEC_COUNT field in the card s EXT_CSD register for MMC cards compliant with MMC specification version 4.x.

To write/read a packet whose size is less than 512 bytes to/from a high-capacity card, write/read the whole 512-byte block that contains the range of bytes to modify/read.

19.7 MMC/SD/SDIO Register Manual

19.7.1 MMC/SD/SDIO Instance Summary

Table 19-32 lists the instance summary.

Table 19-32. Instance Summary

Module Name	Base Address	Size
MMCHS1	0x4809 C000	512 bytes
MMCHS2	0x480B 4000	512 bytes
MMCHS3	0x480A D000	512 bytes

19.7.2 MMC/SD/SDIO Registers Mapping Summary

Table 19-33 lists the MMC/SD/SDIO1 to MMC/SD/SDIO3 registers. through describe the register bits.

CAUTION

The MMC/SD/SDIOi registers are limited to 32-bit data accesses. 16-bit and 8-bit are not allowed and can corrupt register content.

Table 19-33. MMC/SD/SDIO1 Register Summary

Register	Type	Register Width (Bits)	Offset Address	MMCHS1 Physical Address	MMCHS2 Physical Address	MMCHS3 Physical Address
MMCHS_SYSCONFIG	RW	32	0x10	0x4809 C010	0x480B 4010	0x480A D010
MMCHS_SYSSTATUS	R	32	0x14	0x4809 C014	0x480B 4014	0x480A D014
MMCHS_CSRE	RW	32	0x24	0x4809 C024	0x480B 4024	0x480A D024
MMCHS_SYSTEST	RW	32	0x28	0x4809 C028	0x480B 4028	0x480A D028
MMCHS_CON	RW	32	0x2C	0x4809 C02C	0x480B 402C	0x480A D02C
MMCHS_PWCNT	RW	32	0x30	0x4809 C030	0x480B 4030	0x480A D030
MMCHS_BLK	RW	32	0x104	0x4809 C104	0x480B 4104	0x480A D104
MMCHS_ARG	RW	32	0x108	0x4809 C108	0x480B 4108	0x480A D108
MMCHS_CMD	RW	32	0x10C	0x4809 C10C	0x480B 410C	0x480A D10C
MMCHS_RSP10	R	32	0x110	0x4809 C110	0x480B 4110	0x480A D110
MMCHS_RSP32	R	32	0x114	0x4809 C114	0x480B 4114	0x480A D114
MMCHS_RSP54	R	32	0x118	0x4809 C118	0x480B 4118	0x480A D118
MMCHS_RSP76	R	32	0x11C	0x4809 C11C	0x480B 411C	0x480A D11C
MMCHS_DATA	RW	32	0x120	0x4809 C120	0x480B 4120	0x480A D120
MMCHS_PSTATE	R	32	0x124	0x4809 C124	0x480B 4124	0x480A D124
MMCHS_HCTL	RW	32	0x128	0x4809 C128	0x480B 4128	0x480A D128
MMCHS_SYSCTL	RW	32	0x12C	0x4809 C12C	0x480B 412C	0x480A D12C
MMCHS_STAT	RW	32	0x130	0x4809 C130	0x480B 4130	0x480A D130
MMCHS_IE	RW	32	0x134	0x4809 C134	0x480B 4134	0x480A D134
MMCHS_ISE	RW	32	0x138	0x4809 C138	0x480B 4138	0x480A D138
MMCHS_AC12	R	32	0x13C	0x4809 C13C	0x480B 413C	0x480A D13C
MMCHS_CAPA	RW	32	0x140	0x4809 C140	0x480B 4140	0x480A D140
MMCHS_CUR_CAPA	RW	32	ox148	0x4809 C148	0x480B 4148	0x480A D148
MMCHS_REV	R	32	0x1FC	0x4809 C1FC	0x480B 41FC	0x480A D1FC

Table 19-34. MMCHS_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4809 C010	**Instance**	MMCHS1
	0x480A D010		MMCHS3
	0x480B 4010		MMCHS2
Description	System Configuration Register This register allows controlling various parameters of the Interconnect interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						CLOCKACTIVITY		Reserved			SIDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:10	Reserved	These bits are initialized to zero, and writes to them are ignored. Reads return 0	R	0x00000
9:8	CLOCKACTIVITY	Clocks activity during wake up mode period. Bit8: Interface clock Bit9: Functional clock 0x0: Interface and Functional clock may be switched off. 0x1: Interface clock is maintained. Functional clock may be switched-off. 0x2: Functional clock is maintained. Interface clock may be switched-off. 0x3: Interface and Functional clocks are maintained.	RW	0x0
7:5	Reserved	These bits are initialized to zero, and writes to them are ignored. Reads return 0	R	0
4:3	SIDLEMODE	Power management 0x0: If an idle request is detected, the MMC/SD/SDIO host controller acknowledges it unconditionally and goes in Inactive mode. Interrupt and DMA requests are unconditionally deasserted. 0x1: If an idle request is detected, the request is ignored and the module keeps on behaving normally. 0x2: If an idle request is detected, the module will switch to wake up mode based on its internal activity, and the wake up capability can be used if the wake up capability is enabled (bit MMCi.MMCHS_SYSCONFIG[2] ENAWAKEUP bit is set to 1). 0x3: Reserved - do not use	RW	0x2
2	ENAWAKEUP	Wake-up feature control 0x0: Wake-up capability is disabled 0x1: Wake-up capability is enabled	RW	1
1	SOFTRESET	Software reset. The bit is automatically reset by the hardware. During reset, it always returns 0. Read 0x0: Normal mode Write 0x0: No effect. Read 0x1: The module is reset. Write 0x1: Trigger a module reset.	RW	0
0	AUTOIDLE	Internal Clock gating strategy 0x0: Clocks are free-running 0x1: Automatic clock gating strategy is applied, based on the interconnect and MMC interface activity	RW	1

Table 19-35. Register Call Summary for Register MMCHS_SYSCONFIG

MMC/SD/SDIO Integration
- Clocks: [0] [1] [2] [3] [4] [5] [6]
- Resets: [7]

MMC/SD/SDIO Use Cases and Tips
- Programming Flow: [8] [9]

MMC/SD/SDIO Register Manual
- MMC/SD/SDIO Registers Mapping Summary: [10] [11] [12] [13] [14] [15] [16] [17]

Table 19-36. MMCHS_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4809 C014	**Instance**	MMCHS1
	0x480A D014		MMCHS3
	0x480B 4014		MMCHS2
Description	System Status Register This register provides status information about the module excluding the interrupt status information		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Reserved bit field. Do not write any value.	R	0x00000000
0	RESETDONE	Internal Reset Monitoring Note: the debounce clock , the interface clock and the functional clock shall be provided to the MMC/SD/SDIO host controller to allow the internal reset monitoring. Read 0x0: Internal module reset is on-going Read 0x1: Reset completed.	R	0

Table 19-37. Register Call Summary for Register MMCHS_SYSSTATUS

MMC/SD/SDIO Integration
- Resets: [0] [1] [2]

MMC/SD/SDIO Use Cases and Tips
- Programming Flow: [3] [4]

MMC/SD/SDIO Register Manual
- MMC/SD/SDIO Registers Mapping Summary: [5]

Table 19-38. MMCHS_CSRE

Address Offset	0x024		
Physical Address	0x4809 C024	**Instance**	MMCHS1
	0x480A D024		MMCHS3
	0x480B 4024		MMCHS2
Description	Card status response error This register enables the host controller to detect card status errors of response type R1, R1b for all cards and of R5, R5b and R6 response for cards types SD or SDIO. When a bit MMCi.MMCHS_CSRE[I] is set to 1, if the corresponding bit at the same position in the response MMCi.MMCHS_RSP10[I] is set to 1, the host controller indicates a card error (MMCi.MMCHS_STAT[28] CERR bit) interrupt status to avoid the host driver reading the response register (MMCi.MMCHS_RSP10). Note: No automatic card error detection for autoCMD12 is implemented; the host system has to check autoCMD12 response register (MMCi.MMCHS_RSP76) for possible card errors.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSRE																															

Bits	Field Name	Description	Type	Reset
31:0	CSRE	Card status response error	RW	0x00000000

Table 19-39. Register Call Summary for Register MMCHS_CSRE

MMC/SD/SDIO Register Manual

- MMC/SD/SDIO Registers Mapping Summary: [0] [1] [2]

Table 19-40. MMCHS_SYSTEST

Address Offset	0x028		
Physical Address	0x4809 C028	**Instance**	MMCHS1
	0x480A D028		MMCHS3
	0x480B 4028		MMCHS2
Description	System Test register This register is used to control the signals that connect to I/O pins when the module is configured in system test (SYSTEST) mode for boundary connectivity verification. Note: In SYSTEST mode, a write into MMCi.MMCHS_CMD register will not start a transfer. The buffer behaves as a stack accessible only by the local host (push and pop operations). In this mode, the Transfer Block Size (MMCi.MMCHS_BLK[10:0] BLEN bits) and the Blocks count for current transfer (MMCi.MMCHS_BLK[31:16] NBLK bits) are needed to generate a Buffer write ready interrupt (MMCi.MMCHS_STAT[4] BWR bit) or a Buffer read ready interrupt (MMCi.MMCHS_STAT[5] BRR bit) and DMA requests if enabled.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															OBI	SDCD	SDWP	WAKD	SSB	D7D	D6D	D5D	D4D	D3D	D2D	D1D	D0D	DDIR	CDAT	CDIR	MCKD

Bits	Field Name	Description	Type	Reset
31:17	Reserved	Reserved bit field. Do not write any value. Reads return 0.	R	0x00000
16	OBI	Out-Of-Band Interrupt (OBI) data value.	RW	0
		0x0: The Out-of-Band Interrupt pin is driven low.		
		0x1: The Out-of-Band Interrupt pin is driven high.		
15	SDCD	Card detect input signal (SDCD) data value	R	0
		0x0: The card detect pin is driven low.		
		0x1: The card detect pin is driven high.		

Bits	Field Name	Description	Type	Reset
14	SDWP	Write protect input signal (SDWP) data value. 0x0: The write protect pin SDWP is driven low. 0x1: The write protect pin SDWP is driven high.	R	0
13	WAKD	Wake request output signal data value. Read 0x0: No action. Returns 0. Write 0x0: The pin SWAKEUP is driven low. Read 0x1: No action. Returns 1. Write 0x1: The pin SWAKEUP is driven high.	RW	0
12	SSB	Set status bit This bit must be cleared prior attempting to clear a status bit of the interrupt status register (MMCi.MMCHS_STAT). Read 0x0: No action. Returns 0. Write 0x0: Clear this SSB bit field. Writing 0 does not clear already set status bits. Read 0x1: No action. Returns 1. Write 0x1: Force to 1 all status bits of the interrupt status register (MMCi.MMCHS_STAT) only if the corresponding bit field in the Interrupt signal enable register (MMCi.MMCHS_ISE) is set.	RW	0
11	D7D	DAT7 input/output signal data value. Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT7 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0. Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT7 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect. Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT7 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1. Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT7 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.	RW	0
10	D6D	DAT6 input/output signal data value. Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT6 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0. Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT6 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect. Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT6 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1. Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT6 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.	RW	0
9	D5D	DAT5 input/output signal data value.	RW	0

Bits	Field Name	Description	Type	Reset
		Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT5 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.		
		Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT5 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
		Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT5 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.		
		Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT5 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
8	D4D	DAT4 input/output signal data value.	RW	0
		Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT4 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.		
		Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT4 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
		Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT4 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.		
		Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT4 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
7	D3D	DAT3 input/output signal data value.	RW	0
		Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT3 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.		
		Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT3 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
		Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT3 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.		
		Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT3 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
6	D2D	DAT2 input/output signal data value.	RW	0
		Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT2 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.		

Bits	Field Name	Description	Type	Reset
		Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT2 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect. Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT2 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1. Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT2 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.		
5	D1D	DAT1 input/output signal data value. Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT1 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0. Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT1 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect. Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT1 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1. Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT1 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.	RW	0
4	D0D	DAT0 input/output signal data value. Read 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT0 line (low). If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0. Write 0x0: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT0 line is driven low. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect. Read 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT0 line (high) If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1. Write 0x1: If MMCi.MMCHS_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT0 line is driven high. If MMCi.MMCHS_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.	RW	0
3	DDIR	Control of the DAT[7:0] pins direction. Read 0x0: No action. Returns 0. Write 0x0: The DAT lines are outputs (host to card). Read 0x1: No action. Returns 1. Write 0x1: The DAT lines are inputs (card to host).	RW	0
2	CDAT	CMD input/output signal data value. Read 0x0: If MMCi.MMCHS_SYSTEST[1] CDIR bit = 1 (input mode direction), returns the value on the CMD line (low). If MMCi.MMCHS_SYSTEST[1] CDIR bit = 0 (output mode direction), returns 0 .	RW	0

Bits	Field Name	Description		Type	Reset
		Write 0x0:	If MMCi.MMCHS_SYSTEST[1] CDIR bit = 0 (output mode direction), the CMD line is driven low. If MMCi.MMCHS_SYSTEST[1] CDIR bit = 1 (input mode direction), no effect.		
		Read 0x1:	If MMCi.MMCHS_SYSTEST[1] CDIR bit = 1 (input mode direction), returns the value on the CMD line (high) If MMCi.MMCHS_SYSTEST[1] CDIR bit = 0 (output mode direction), returns 1 .		
		Write 0x1:	If MMCi.MMCHS_SYSTEST[1] CDIR bit = 0 (output mode direction), the CMD line is driven high. If MMCi.MMCHS_SYSTEST[1] CDIR bit = 1 (input mode direction), no effect.		
1	CDIR	Control of the CMD pin direction.		RW	0
		Read 0x0:	No action. Returns 0.		
		Write 0x0:	The CMD line is an output (host to card).		
		Read 0x1:	No action. Returns 1.		
		Write 0x1:	The CMD line is an input (card to host) .		
0	MCKD	MMC clock output signal data value.		RW	0
		Read 0x0:	No action. Returns 0.		
		Write 0x0:	The output clock is driven low.		
		Read 0x1:	No action. Returns 1.		
		Write 0x1:	The output clock is driven high.		

Table 19-41. Register Call Summary for Register MMCHS_SYSTEST

MMC/SD/SDIO Register Manual

- MMC/SD/SDIO Registers Mapping Summary: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72]

Table 19-42. MMCHS_CON

Address Offset	0x02C		
Physical Address	0x4809 C02C	**Instance**	MMCHS1
	0x480A D02C		MMCHS3
	0x480B 402C		MMCHS2
Description	Configuration register This register is used: - to select the functional mode for any card. - to send an initialization sequence to any card. - to enable the detection on the mmci_dat[1] signal of a card interrupt for SDIO cards only. And also to configure: - specific data and command transfers for MMC cards only. - the parameters related to the card detect and write protect input signals.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															CLKEXTFREE	PADEN	OBIE	OBIP	CEATA	CTPL	DVAL		WPP	CDP	MIT	DW8	MODE	STR	HR	INIT	OD

Bits	Field Name	Description	Type	Reset
31:17	Reserved	Reserved bit field. Do not write any value	R	0x00000
16	CLKEXTFREE	External clock free running. This register is used to maintain card clock out of transfer transaction to enable slave module (for example to generate a synchronous interrupt on mmci_dat[1]). The Clock will be maintain only if MMCi.MMCHS_SYSCTL[2] CEN bit is set. 0x0: External card clock is cut off outside active transaction period. 0x1: External card clock is maintain even out of active transaction period only if MMCi.MMCHS_SYSCTL[2] CEN bit is set.	RW	0
15	PADEN	Control Power for MMC Lines. This register is only useful when MMC PADs contain power saving mechanism to minimize its leakage power. It works as a GPIO that directly control the ACTIVE pin of PADs. Excepted for mmci_dat[1], the signal is also combine outside the module with the dedicated power control MMCi.MMCHS_CON[11] CTPL bit. 0x0: ADPIDLE module pin is not forced, it is automatically generated by the MMC fsms. 0x1: ADPIDLE module pin is forced to active state.	RW	0
14	OBIE	Out-of-Band Interrupt Enable (MMC cards only). This bit enables the detection of Out-of-Band Interrupt on MMCOBI input pin. The usage of the Out-of-Band signal (OBI) is optional and depends on the system integration. 0x0: Out-of-Band interrupt detection disabled. 0x1: Out-of-Band interrupt detection enabled.	RW	0
13	OBIP	Out-of-Band Interrupt Polarity (MMC cards only). This bit selects the active level of the out-of-band interrupt coming from MMC cards. The usage of the Out-of-Band signal (OBI) is optional and depends on the system integration. 0x0: active high level. 0x1: active low level.	RW	0
12	CEATA	CE-ATA control mode (MMC cards compliant with CE-ATA): By default, this bit is set to 0. It is use to indicate that next commands are considered as specific CE-ATA commands that potentially use 'command completion' features. 0x0: Standard MMC/SD/SDIO mode. 0x1: CE-ATA mode. Next commands are considered as CE-ATA commands.	RW	0
11	CTPL	Control Power for mmci_dat[1] line (MMC and SD cards): By default, this bit is set to 0 and the host controller automatically disables all the input buffers outside of a transaction to minimize the leakage current. SDIO cards: When this bit is set to 1, the host controller automatically disables all the input buffers except the buffer of mmci_dat[1] outside of a transaction in order to detect asynchronous card interrupt on mmci_dat[1] line and minimize the leakage current of the buffers. 0x0: Disable all the input buffers outside of a transaction. 0x1: Disable all the input buffers except the buffer of mmci_dat[1] outside of a transaction.	RW	0
10:9	DVAL	Debounce filter value (All cards) This register is used to define a debounce period to filter the card detect input signal (SDCD). The usage of the card detect input signal (SDCD) is optional and depends on the system integration and the type of the connector housing that accommodates the card. 0x0: 33 us debounce period. 0x1: 231 us debounce period.	RW	0x3

Bits	Field Name	Description	Type	Reset
		0x2: 1 ms debounce period. 0x3: 8.4 ms debounce period.		
8	WPP	Write protect polarity (SD and SDIO cards only) This bit selects the active level of the write protect input signal (SDWP). The usage of the write protect input signal (SDWP) is optional and depends on the system integration and the type of the connector housing that accommodates the card. 0x0: Active high level. 0x1: Active low level.	RW	0
7	CDP	Card detect polarity (All cards) This bit selects the active level of the card detect input signal (SDCD). The usage of the card detect input signal (SDCD) is optional and depends on the system integration and the type of the connector housing that accommodates the card. 0x0: Active high level. 0x1: Active low level.	RW	0
6	MIT	MMC interrupt command (Only for MMC cards.) This bit must be set to 1, when the next write access to the command register (MMCi.MMCHS_CMD) is for writing a MMC interrupt command (CMD40) requiring the command timeout detection to be disabled for the command response. 0x0: Command timeout enabled 0x1: Command timeout disabled	RW	0
5	DW8	8-bit mode MMC select For SD/SDIO cards, this bit must be set to 0. For MMC card, this bit must be set following a valid SWITCH command (CMD6) with the correct value and extend CSD index written in the argument. Prior to this command, the MMC card configuration register (CSD and EXT_CSD) must be verified for compliancy with MMC standard specification. 0x0: 1-bit or 4-bit Data width (mmci_dat[0] or mmci_dat[3:0] used, MMC, SD cards) 0x1: 8-bit Data width (mmci_dat[7:0] used, MMC cards)	RW	0
4	MODE	Mode select (All cards) These bits select the functional mode. 0x0: Functional mode. Transfers to the MMC/SD/SDIO cards follow the card protocol. MMC clock is enabled. MMC/SD transfers are operated under the control of the MMCi.MMCHS_CMD register. 0x1: SYSTEST mode. The signal pins are configured as general-purpose input/output and the 1024-byte buffer is configured as a stack memory accessible only by the local host or system DMA. The pins retain their default type (input, output or in-out). SYSTEST mode is operated under the control of the SYSTEST register.	RW	0
3	STR	Stream command (Only for MMC cards). This bit must be set to 1 only for the stream data transfers (read or write) of the adtc commands. Stream read is a class 1 command (CMD11: READ_DAT_UNTIL_STOP). Stream write is a class 3 command (CMD20: WRITE_DAT_UNTIL_STOP). 0x0: Block oriented data transfer. 0x1: Stream oriented data transfer.	RW	0

Bits	Field Name	Description	Type	Reset
2	HR	Broadcast host response (Only for MMC cards). This register is used to force the host to generate a 48-bit response for bc command type. It can be used to terminate the interrupt mode by generating a CMD40 response by the core. In order to have the host response to be generated in open drain mode, the register MMCHS_CON[OD] must be set to 1. When MMCi.MMCHS_CON[12] CEATA bit is set to 1 and MMCi.MMCHS_ARG set to 0x00000000, when writing 0x00000000 into MMCi.MMCHS_CMD register, the host controller performs a 'command completion signal disable' token (i.e. mmci_cmd line held to '0' during 47 cycles followed by a 1). 0x0: The host does not generate a 48-bit response instead of a command. 0x1: The host generates a 48-bit response instead of a command or a command completion signal disable token.	RW	0
1	INIT	Send initialization stream (All cards). When this bit is set to 1, and the card is idle, an initialization sequence is sent to the card. An initialization sequence consists of setting the mmci_cmd line to 1 during 80 clock cycles. The initialization sequence is mandatory - but it is not required to do it through this bit - this bit makes it easier. Clock divider (MMCi.MMCHS_SYSCTL[15:6] CLKD bits) should be set to ensure that 80 clock periods are greater than 1ms. Note: in this mode, there is no command sent to the card and no response is expected. A command complete interrupt will be generated once the initialization sequence is completed. MMCi.MMCHS_STAT[0] CC bit can be polled. 0x0: The host does not send an initialization sequence. 0x1: The host sends an initialization sequence.	RW	0
0	OD	Card open drain mode (Only for MMC cards). This bit must be set to 1 for MMC card commands 1, 2, 3 and 40, and if the MMC card bus is operating in open-drain mode during the response phase to the command sent. Typically, during card identification mode when the card is either in idle, ready or ident state. It is also necessary to set this bit to 1, for a broadcast host response (see Broadcast host response register MMCi.MMCHS_CON[2] HR bit) 0x0: No Open Drain 0x1: Open Drain or Broadcast host response	RW	0

Table 19-43. Register Call Summary for Register MMCHS_CON

MMC/SD/SDIO Functional Description
• MMC CE-ATA Command Completion Disable Management: [0] [1] [2]
MMC/SD/SDIO Use Cases and Tips
• Programming Flow: [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26]
MMC/SD/SDIO Register Manual
• MMC/SD/SDIO Registers Mapping Summary: [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39]

Table 19-44. MMCHS_PWCNT

Address Offset	0x030		
Physical Address	0x4809 C030	**Instance**	MMCHS1
	0x480A D030		MMCHS3
	0x480B 4030		MMCHS2
Description	Power counter register This register is used to program a mmc counter to delay command transfers after activating the PAD power, this value depends on PAD characteristics and voltage.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PWRCNT															

Bits	Field Name	Description	Type	Reset
31:16	Reserved	These bits are initialized to zero, and writes to them are ignored. Reads return 0.	R	0x00
15:0	PWRCNT	Power counter register. This register is used to introduce a delay between the PAD ACTIVE pin assertion and the command issued. 0x0: No additional delay added. 0x1: TCF delay (card clock period). 0x2: TCF x 2 delay (card clock period). 0xFFFE: TCF x 65534 delay (card clock period). 0xFFFF: TCF x 65535 delay (card clock period).	RW	0x0000

Table 19-45. Register Call Summary for Register MMCHS_PWCNT

MMC/SD/SDIO Register Manual

- MMC/SD/SDIO Registers Mapping Summary: [0]

Table 19-46. MMCHS_BLK

Address Offset	0x104		
Physical Address	0x4809 C104	**Instance**	MMCHS1
	0x480A D104		MMCHS3
	0x480B 4104		MMCHS2
Description	Transfer Length Configuration register This register shall be used for any card.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBLK																Reserved					BLEN										

Bits	Field Name	Description	Type	Reset
31:16	NBLK	Blocks count for current transfer. This register is enabled when Block count Enable (MMCi.MMCHS_CMD[1] BCE bit) is set to 1 and is valid only for multiple block transfers. Setting the block count to 0 results no data blocks being transferred. Note: The host controller decrements the block count after each block transfer and stops when the count reaches zero. This register can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value and write operation will be ignored. In suspend context, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, The local host shall restore the previously saved block count. 0x0: Stop count 0x1: 1 block 0x2: 2 blocks 0xFFFF: 65535 blocks	RW	0x0000
15:11	Reserved	Reserved bit field. Do not write any value	R	0x00
10:0	BLEN	Transfer Block Size. This register specifies the block size for block data transfers. Read operations during transfers may return an invalid value, and write operations are ignored. When a CMD12 command is issued to stop the transfer, a read of the BLEN field after transfer completion (MMCi.MMCHS_STAT[1] TC bit set to 1) will not return the true byte number of data length while the stop occurs but the value written in this register before transfer is launched. 0x0: No data transfer 0x1: 1 byte block length 0x2: 2 bytes block length 0x3: 3 bytes block length 0x1FF: 511 bytes block length 0x200: 512 bytes block length 0x3FF: 1023 bytes block length 0x400: 1024 bytes block length	RW	0x000

Table 19-47. Register Call Summary for Register MMCHS_BLK

MMC/SD/SDIO Integration • DMA Requests: [0] [1]
MMC/SD/SDIO Functional Description • Data Buffer: [2]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [3] [4]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [5] [6] [7] [8] [9] [10] [11] [12]

Table 19-48. MMCHS_ARG

Address Offset	0x108		
Physical Address	0x4809 C108	**Instance**	MMCHS1
	0x480A D108		MMCHS3
	0x480B 4108		MMCHS2
Description	Command argument Register This register contains command argument specified as bit 39-8 of Command-Format These registers must be initialized prior to sending the command itself to the card (write action into the register MMCi.MMCHS_CMD register). Only exception is for a command index specifying stuff bits in arguments, making a write unnecessary.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG																															

Bits	Field Name	Description	Type	Reset
31:0	ARG	Command argument bits [31:0](1)	RW	0x00000000

(1) For CMD52, ARG has to be programmed with IO_RW_DIRECT[39:8]. Refer to SDIO specification.

Table 19-49. Register Call Summary for Register MMCHS_ARG

MMC/SD/SDIO Functional Description
- MMC CE-ATA Command Completion Disable Management: [0]

MMC/SD/SDIO Use Cases and Tips
- Programming Flow: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18]

MMC/SD/SDIO Register Manual
- MMC/SD/SDIO Registers Mapping Summary: [19] [20]

Table 19-50. MMCHS_CMD

Address Offset	0x10C		
Physical Address	0x4809 C10C	**Instance**	MMCHS1
	0x480A D10C		MMCHS3
	0x480B 410C		MMCHS2
Description	Command and transfer mode register MMCi.MMCHS_CMD[31:16] = the command register MMCi.MMCHS_CMD[15:0] = the transfer mode. This register configures the data and command transfers. A write into the most significant byte send the command. A write into MMCi.MMCHS_CMD[15:0] registers during data transfer has no effect. This register shall be used for any card.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				INDX				CMD_TYPE		DP	CICE	CCCE	Reserved	RSP_TYPE		Reserved								MSBS	DDIR	Reserved		ACEN	BCE	DE	

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Reserved bit field. Do not write any value	R	0x0
29:24	INDX	Command index Binary encoded value from 0 to 63 specifying the command number send to card	RW	0x00
		0x0: CMD0 or ACMD0		
		0x1: CMD1 or ACMD1		

Bits	Field Name	Description	Type	Reset
		0x3F: CMD63 or ACMD63		
23:22	CMD_TYPE	Command type. This register specifies three types of special command: Suspend, Resume and Abort. These bits shall be set to 0b00 for all other commands. 0x0: Others Commands 0x1: Upon CMD52 "Bus Suspend" operation 0x2: Upon CMD52 "Function Select" operation 0x3: Upon CMD12 or CMD52 "I/O Abort" command	RW	0x0
21	DP	Data present select. This register indicates that data is present and mmci_dat line shall be used. It must be set to 0 in the following conditions: - Command using only mmci_cmd line -Command with no data transfer but using busy signal on mmci_dat[0] -Resume command 0x0: Command with no data transfer 0x1: Command with data transfer	RW	0
20	CICE	Command Index check enable. This bit must be set to 1 to enable index check on command response to compare the index field in the response against the index of the command. If the index is not the same in the response as in the command, it is reported as a command index error (MMCi.MMCHS_STAT[19] CIE bit set to1) Note: The CICE bit cannot be configured for an Auto CMD12, then index check is automatically checked when this command is issued. 0x0: Index check disable 0x1: Index check enable	RW	0
19	CCCE	Command CRC check enable. This bit must be set to 1 to enable CRC7 check on command response to protect the response against transmission errors on the bus. If an error is detected, it is reported as a command CRC error (MMCi.MMCHS_STAT[17] CCRC bit set to 1). Note: The CCCE bit cannot be configured for an Auto CMD12, and then CRC check is automatically checked when this command is issued. 0x0: CRC7 check disable 0x1: CRC7 check enable	RW	0
18	Reserved	Reserved bit field. Do not write any value.	R	0
17:16	RSP_TYPE	Response type. This bits defines the response type of the command. 0x0: No response 0x1: Response Length 136 bits 0x2: Response Length 48 bits 0x3: Response Length 48 bits with busy after response	RW	0x0
15:6	Reserved	Reserved bit field. Do not write any value.	R	0x000
5	MSBS	Multi/Single block select. This bit must be set to 1 for data transfer in case of multi block command. For any others command this bit shall be set to 0. 0x0: Single block. If this bit is 0, it is not necessary to set the register MMCi.MMCHS_BLK[31:16] NBLK bits. 0x1: Multi block. When Block Count is disabled (MMCi.MMCHS_CMD[1] BCE bit is set to 0) in Multiple block transfers (MMCi.MMCHS_CMD[5] MSBS bit is set to 1), the module can perform infinite transfer.	RW	0
4	DDIR	Data transfer Direction. Select This bit defines either data transfer will be a read or a write. 0x0: Data Write (host to card) 0x1: Data Read (card to host)	RW	0
3	Reserved	Reserved bit field. Do not write any value.	R	0

Bits	Field Name	Description	Type	Reset
2	ACEN	Auto CMD12 Enable. (SDIO does not support this feature.) When this bit is set to 1, the host controller issues a CMD12 automatically after the transfer completion of the last block. The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop data transfer. For CE-ATA commands (MMCi.MMCHS_CON[12] CEATA bit set to 1), auto CMD12 is useless; therefore when this bit is set the mechanism to detect command completion signal, named CCS, interrupt is activated. 0x0: Auto CMD12 disable 0x1: Auto CMD12 enable or CCS detection enabled.	RW	0
1	BCE	Block Count Enable (Multiple block transfers only). This bit is used to enable the block count register (MMCHS_BLK[31:16] NBLK bits). When Block Count is disabled (MMCHS_CMD[1] BCE bit is set to 0) in Multiple block transfers (MMCHS_CMD[5] MSBS bits is set to 1), the module can perform infinite transfer. 0x0: Block count disabled for infinite transfer. 0x1: Block count enabled for multiple block transfer with known number of blocks	RW	0
0	DE	DMA Enable. This bit is used to enable DMA mode for host data access. 0x0: DMA mode disable 0x1: DMA mode enable	RW	0

Table 19-51. Register Call Summary for Register MMCHS_CMD

MMC/SD/SDIO Environment • MMC/SD/SDIO Protocol and Data Format: [0] [1] [2] [3]
MMC/SD/SDIO Integration • DMA Requests: [4]
MMC/SD/SDIO Functional Description • Data Buffer: [5] • Transfer Stop: [6] • MMC CE-ATA Command Completion Disable Management: [7] [8]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46]

Table 19-52. MMCHS_RSP10

Address Offset	0x110		
Physical Address	0x4809 C110	**Instance**	MMCHS1
	0x480A D110		MMCHS3
	0x480B 4110		MMCHS2
Description	Command response[31:0] Register This 32-bit register holds bits positions [31:0] of command response type R1/R1b/R2/R3/R4/R5/R5b/R6		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSP1																RSP0															

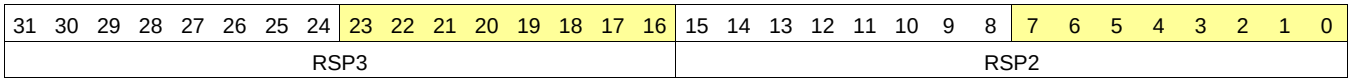
Bits	Field Name	Description	Type	Reset
31:16	RSP1	R1/R1b (normal response) /R3/R4/R5/R5b/R6 : Command Response [39:24] R2: Command Response [31:16]	R	0x0000
15:0	RSP0	R1/R1b (normal response) /R3/R4/R5/R5b/R6 : Command Response [23:8] R2: Command Response [15:0]	R	0x0000

Table 19-53. Register Call Summary for Register MMCHS_RSP10

MMC/SD/SDIO Functional Description • Different Types of Responses: [0] [1] [2]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [3] [4] [5] [6] [7] [8] [9] [10]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [11] [12] [13]

Table 19-54. MMCHS_RSP32

Address Offset	0x114		
Physical Address	0x4809 C114	**Instance**	MMCHS1
	0x480A D114		MMCHS3
	0x480B 4114		MMCHS2
Description	Command response[63:32] Register This 32-bit register holds bits positions [63:32] of command response type R2		
Type	R		



Bits	Field Name	Description	Type	Reset
31:16	RSP3	R2: Command Response [63:48]	R	0x0000
15:0	RSP2	R2: Command Response [47:32]	R	0x0000

Table 19-55. Register Call Summary for Register MMCHS_RSP32

MMC/SD/SDIO Functional Description • Different Types of Responses: [0]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [1] [2] [3]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [4]

Table 19-56. MMCHS_RSP54

Address Offset	0x118		
Physical Address	0x4809 C118	**Instance**	MMCHS1
	0x480A D118		MMCHS3
	0x480B 4118		MMCHS2
Description	Command response[95:64] Register This 32-bit register holds bits positions [95:64] of command response type R2		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSP5																RSP4															

Bits	Field Name	Description	Type	Reset
31:16	RSP5	R2: Command Response [95:80]	R	0x0000
15:0	RSP4	R2: Command Response [79:64]	R	0x0000

Table 19-57. Register Call Summary for Register MMCHS_RSP54

MMC/SD/SDIO Functional Description • Different Types of Responses: [0]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [1] [2] [3]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [4]

Table 19-58. MMCHS_RSP76

Address Offset	0x11C		
Physical Address	0x4809 C11C	**Instance**	MMCHS1
	0x480A D11C		MMCHS3
	0x480B 411C		MMCHS2
Description	Command response[127:96] Register This 32-bit register holds bits positions [127:96] of command response type R2		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSP7																RSP6															

Bits	Field Name	Description	Type	Reset
31:16	RSP7	R1b (Auto CMD12 response): Command Response [39:24] R2: Command Response [127:112]	R	0x0000
15:0	RSP6	R1b (Auto CMD12 response): Command Response [23:8] R2: Command Response [111:96]	R	0x0000

Table 19-59. Register Call Summary for Register MMCHS_RSP76

MMC/SD/SDIO Functional Description • Different Types of Responses: [0] [1] [2]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [3] [4] [5]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [6] [7] [8]

Table 19-60. MMCHS_DATA

Address Offset	0x120		
Physical Address	0x4809 C120	**Instance**	MMCHS1
	0x480A D120		MMCHS3
	0x480B 4120		MMCHS2
Description	Data Register This register is the 32-bit entry point of the buffer for read or write data transfers. The buffer size is 32bits x256(1024 bytes). Bytes within a word are stored and read in little endian format. This buffer can be used as two 512 byte buffers to transfer data efficiently without reducing the throughput. Sequential and contiguous access is necessary to increment the pointer correctly. Random or skipped access is not allowed. In little endian, if the local host accesses this register byte-wise or 16bit-wise, the least significant byte (bits [7:0]) must always be written/read first. The update of the buffer address is done on the most significant byte write for full 32-bit DATA register or on the most significant byte of the last word of block transfer. Example 1: Byte or 16-bit access Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=1100 (2-bytes) OK Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=0100 (1-byte) OK Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=1000 (1-byte) Bad		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA																															

Bits	Field Name	Description	Type	Reset
31:0	DATA	Data Register [31:0] In functional mode (MMCI.MMCHS_CON[4] MODE bit set to the default value 0): A read access to this register is allowed only when the buffer read enable status is set to 1 (MMCi.MMCHS_PSTATE[11] BRE bit), otherwise a bad access (MMCi.MMCHS_STAT[29] BADA bit) is signaled. A write access to this register is allowed only when the buffer write enable status is set to 1 (MMCi.MMCHS_PSTATE[10] BWE bit), otherwise a bad access (MMCi.MMCHS_STAT[29] BADA bit) is signaled and the data is not written.	RW	0x00000000

Table 19-61. Register Call Summary for Register MMCHS_DATA

MMC/SD/SDIO Functional Description

- Data Buffer: [0] [1] [2] [3] [4] [5] [6] [7]
- Transfer or Command Status and Errors Reporting: [8]

MMC/SD/SDIO Register Manual

- MMC/SD/SDIO Registers Mapping Summary: [9] [10] [11] [12] [13]

Table 19-62. MMCHS_PSTATE

Address Offset	0x124		
Physical Address	0x4809 C124	**Instance**	MMCHS1
	0x480A D124		MMCHS3
	0x480B 4124		MMCHS2
Description	Present state register. The Host can get status of the Host Controller from this 32-bit read only register.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CLEV	DLEV				Reserved	Reserved	Reserved		Reserved				BRE	BWE	RTA	WTA	Reserved					DLA	DATI	CMDI

Bits	Field Name	Description	Type	Reset
31:25	Reserved	Reserved bit field. Do not write any value.	R	0x00
24	CLEV	mmci_cmd line signal level. This status is used to check the mmci_cmd line level to recover from errors, and for debugging. The value of this register after reset depends on the mmci_cmd line level at that time. Read 0x0: The mmci_cmd line level is 0. Read 0x1: The mmci_cmd line level is 1.	R	-
23:20	DLEV	mmci_dat[3:0] line signal level mmci_dat[3] => bit 23 mmci_dat[2] => bit 22 mmci_dat[1] => bit 21 mmci_dat[0] => bit 20 This status is used to check mmci_dat line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from mmci_dat[0]. The value of these registers after reset depends on the mmci_dat lines level at that time.	R	0x-
19	Reserved	Reserved bit field. Do not write any value	R	0
18	Reserved	Reserved bit field. Do not write any value This bit is not affected by soft reset.	R	1
17:16	Reserved	Reserved bit field. Do not write any value The value of these bits after soft reset is 0x0. These bits will be automatically set to 0x3 after debounce time. Debounce time is fixed to 256 x32kHz clock cycles.	R	00
15:12	Reserved	Reserved bit field. Do not write any value	R	0x0
11	BRE	Buffer read enable. This bit is used for non-DMA read transfers. It indicates that a complete block specified by MMCi.MMCHS_BLK[10:0] BLEN bits has been written in the buffer and is ready to be read. It is set to 0 when the entire block is read from the buffer. It is set to 1 when a block data is ready in the buffer and generates the Buffer read ready status of interrupt (MMCi.MMCHS_STAT[5] BRR bit). Read 0x0: Read BLEN bytes disable Read 0x1: Read BLEN bytes enable. Readable data exists in the buffer.	R	0
10	BWE	Buffer Write enable. This status is used for non-DMA write transfers. It indicates if space is available for write data. Read 0x0: There is no room left in the buffer to write BLEN bytes of data. Read 0x1: There is enough space in the buffer to write BLEN bytes of data.	R	0
9	RTA	Read transfer active. This status is used for detecting completion of a read transfer. It is set to 1 after the end bit of read command or by activating a continue request (MMCi.MMCHS_HCTL[17] CR bit) following a stop at block gap request. This bit is set to 0 when all data have been read by the local host after last block or after a stop at block gap request. Read 0x0: No valid data on the mmci_dat lines. Read 0x1: Read data transfer on going.	R	0
8	WTA	Write transfer active. This status indicates a write transfer active. It is set to 1 after the end bit of write command or by activating a continue request (MMCi.MMCHS_HCTL[17] CR bit) following a stop at block gap request. This bit is set to 0 when CRC status has been received after last block or after a stop at block gap request. Read 0x0: No valid data on the mmci_dat lines. Read 0x1: Write data transfer on going.	R	0
7:3	Reserved	Reserved bit field. Do not write any value	R	0x00

Bits	Field Name	Description	Type	Reset
2	DLA	mmci_dat line active. This status bit indicates whether one of the mmci_dat line is in use. In the case of read transactions (card to host): This bit is set to 1 after the end bit of read command or by activating continue request MMCi.MMCHS_HCTL[17] CR bit. This bit is set to 0 when the host controller received the end bit of the last data block or at the beginning of the read wait mode. In the case of write transactions (host to card): This bit is set to 1 after the end bit of write command or by activating continue request MMCi.MMCHS_HCTL[17] CR bit. This bit is set to 0 on the end of busy event for the last block; host controller must wait 8 clock cycles with line not busy to really consider not "busy state" or after the busy block as a result of a stop at gap request. Read 0x0: mmci_dat Line inactive Read 0x1: mmci_dat Line active	R	0
1	DATI	Command inhibit (mmci_dat). This status bit is generated if either mmci_dat line is active (MMCi.MMCHS_PSTATE[2] DLA bit) or Read transfer is active (MMCi.MMCHS_PSTATE[9] RTA bit) or when a command with busy is issued. This bit prevents the local host to issue a command. A change of this bit from 1 to 0 generates a transfer complete interrupt (MMCi.MMCHS_STAT[1] TC bit). Read 0x0: Issuing of command using the mmci_dat lines is allowed Read 0x1: Issuing of command using mmci_dat lines is not allowed	R	0
0	CMDI	Command inhibit(mmci_cmd). This status bit indicates that the mmci_cmd line is in use. This bit is set to 0 when the most significant byte is written into the command register. This bit is not set when Auto CMD12 is transmitted. This bit is set to 0 in either the following cases: - After the end bit of the command response, excepted if there is a command conflict error (MMCi.MMCHS_STAT[17] CCRC bit or MMCi.MMCHS_STAT[18] CEB bit set to 1) or a Auto CMD12 is not executed (MMCi.MMCHS_AC12[0] ACNE bit). - After the end bit of the command without response (MMCi.MMCHS_CMD[17:16] RSP_TYPE bits set to "00"). In case of a command data error is detected (MMCi.MMCHS_STAT[19] CTO bit set to 1), this register is not automatically cleared. Read 0x0: Issuing of command using mmci_cmd line is allowed Read 0x1: Issuing of command using mmci_cmd line is not allowed	R	0

Table 19-63. Register Call Summary for Register MMCHS_PSTATE

MMC/SD/SDIO Integration • Resets: [0]
MMC/SD/SDIO Functional Description • Data Buffer: [1] [2] [3] [4]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19]

Table 19-64. MMCHS_HCTL

Address Offset	0x128		
Physical Address	0x4809 C128	**Instance**	MMCHS1
	0x480A D128		MMCHS3
	0x480B 4128		MMCHS2
Description	Control register. This register defines the host controls to set power, wake up and transfer parameters. MMCi.MMCHS_HCTL[31:24] = Wake-up control MMCi.MMCHS_HCTL[23:16] = Block gap control MMCi.MMCHS_HCTL[15:8] = Power control MMCi.MMCHS_HCTL[7:0] = Host control		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				OBWE	REM	INS	IWE	Reserved				IBG	RWC	CR	SBGR	Reserved				SDVS			SDBP	Reserved						DTW	Reserved

Bits	Field Name	Description	Type	Reset
31:28	Reserved	Reserved bit field. Do not write any value.	R	0x00
27	OBWE	Wake-up event enable for 'Out-of-Band' Interrupt. This bit enables wake-up events for 'Out-of-Band' assertion. Wakeup is generated if the wake-up feature is enabled (MMCi.MMCHS_SYSCONFIG[2] ENAWAKEUP bit). The write to this register is ignored when MMCi.MMCHS_CON[14] OBIE bit is not set. 0x0: Disable wakeup on 'Out-of-Band' Interrupt. 0x1: Enable wakeup on 'Out-of-Band' Interrupt.	RW	0
26	REM	Wake-up event enable on SD card removal. This bit enables wake-up events for card removal assertion. Wakeup is generated if the wake-up feature is enabled (MMCi.MMCHS_SYSCONFIG[2] ENAWAKEUP bit). 0x0: Disable wakeup on card removal. 0x1: Enable wakeup on card removal.	RW	0
25	INS	Wake-up event enable on SD card insertion This bit enables wake-up events for card insertion assertion. Wakeup is generated if the wake-up feature is enabled (MMCi.MMCHS_SYSCONFIG[2] ENAWAKEUP bit). 0x0: Disable wakeup on card insertion. 0x1: Enable wakeup on card insertion.	RW	0
24	IWE	Wake-up event enable on SD card interrupt. This bit enables wake-up events for card interrupt assertion. Wakeup is generated if the wake-up feature is enabled (MMCi.MMCHS_SYSCONFIG[2] ENAWAKEUP bit) and enable status bit is set (MMCi.MMCHS_IE[8] CIRQ_ENABLE bit). 0x0: Disable wakeup on card interrupt 0x1: Enable wakeup on card interrupt	RW	0
23:20	Reserved	Reserved bit field. Do not write any value	R	0x0
19	IBG	Interrupt block at gap. This bit is valid only in 4-bit mode of SDIO card to enable interrupt detection in the interrupt cycle at block gap for a multiple block transfer. For MMC cards and for SD card this bit should be set to 0. 0x0: Disable interrupt detection at the block gap in 4-bit mode 0x1: Enable interrupt detection at the block gap in 4-bit mode	RW	0
18	RWC	Read wait control. The read wait function is optional only for SDIO cards. If the card supports read wait, this bit must be enabled, then requesting a stop at block gap (MMCi.MMCHS_HCTL[16] SBGR bit) generates a read wait period after the current end of block. Be careful, if read wait is not supported it may cause a conflict on mmci_dat line. 0x0: Disable Read Wait Control. Suspend/Resume cannot be supported. 0x1: Enable Read Wait Control	RW	0
17	CR	Continue request. This bit is used to restart a transaction that was stopped by requesting a stop at block gap (MMCi.MMCHS_HCTL[16] SBGR bit). Set this bit to 1 restarts the transfer. The bit is automatically set to 0 by the host controller when transfer has restarted i.e. mmci_dat line is active (MMCi.MMCHS_PSTATE[2] DLA bit) or transferring data (MMCi.MMCHS_PSTATE[8] WTA bit). The Stop at block gap request must be disabled (MMCi.MMCHS_HCTL[16] SBGR bit =0) before setting this bit. 0x0: No affect 0x1: transfer restart	RW	0

Bits	Field Name	Description	Type	Reset
16	SBGR	Stop at block gap request. This bit is used to stop executing a transaction at the next block gap. The transfer can restart with a continue request (MMCi.MMCHS_HCTL[17] CR bit) or during a suspend/resume sequence. In case of read transfer, the card must support read wait control. In case of write transfer, the host driver shall set this bit after all block data written. Until the transfer completion (MMCi.MMCHS_STAT[1] TC bit set to 1), the host driver shall leave this bit set to 1.If this bit is set, the local host shall not write to the data register (MMCi.MMCHS_DATA). 0x0: Transfer mode 0x1: Stop at block gap	RW	0
15:12	Reserved	Reserved bit field. Do not write any value.	R	0x0
11:9	SDVS	SD bus voltage select (All cards). The host driver should set these bits to select the voltage level for the card according to the voltage supported by the system (MMCi.MMCHS_CAPA[26] VS18 bit, MMCi.MMCHS_CAPA[25] VS30 bit, MMCi.MMCHS_CAPA[24] VS33 bit) before starting a transfer. 0x5: 1.8V (Typical) 0x6: 3.0V (Typical) 0x7: 3.3V (Typical) MMCHS2: This field must be set to 0x5. MMCHS3: This field must be set to 0x5.	RW	0x0
8	SDBP	SD bus power. Before setting this bit, the host driver shall select the SD bus voltage (MMCi.MMCHS_HCTL[11:9] SDVS bits). If the host controller detects the No card state, this bit is automatically set to 0. If the module is power off, a write in the command register (MMCi.MMCHS_CMD) will not start the transfer. A write to this bit has no effect if the selected SD bus voltage is not supported according to capability register (MMCi.MMCHS_CAPA[VS*]). 0x0: Power off 0x1: Power on	RW	0
7:2	Reserved	Reserved bit field. Do not write any value.	R	0x00
1	DTW	Data transfer width. For MMC card, this bit must be set following a valid SWITCH command (CMD6) with the correct value and extend CSD index written in the argument. Prior to this command, the MMC card configuration register (CSD and EXT_CSD) must be verified for compliance with *MMC standard specification 4.x* This register has no effect when the MMC 8-bit mode is selected (MMCi.MMCHS_CON[5] DW8 bit set to 1) For SD/SDIO cards, this bit must be set following a valid SET_BUS_WIDTH command (ACMD6) with the value written in bit 1 of the argument. Prior to this command, the SD card configuration register (SCR) must be verified for the supported bus width by the SD card. 0x0: 1-bit Data width (mmci_dat[0] used) 0x1: 4-bit Data width (mmci_dat[3:0] used)	RW	0
0	Reserved	Reserved bit field. Do not write any value.	R	0

Table 19-65. Register Call Summary for Register MMCHS_HCTL

MMC/SD/SDIO Integration • Clocks: [0] [1]
MMC/SD/SDIO Functional Description • Transfer Stop: [2] [3]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [4] [5] [6] [7] [8] [9] [10] [11]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28]

Table 19-66. MMCHS_SYSCTL

Address Offset	0x12C		
Physical Address	0x4809 C12C	**Instance**	MMCHS1
	0x480A D12C		MMCHS3
	0x480B 412C		MMCHS2
Description	SD system control register. This register defines the system controls to set software resets, clock frequency management and data timeout. MMCHS_SYSCTL[31:24] = Software resets MMCHS_SYSCTL[23:16] = Timeout control MMCHS_SYSCTL[15:0] = Clock control		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SRD	SRC	SRA	Reserved				DTO				CLKD								Reserved					CEN	ICS	ICE

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved bit field. Do not write any value.	R	0x00
26	SRD	Software reset for mmci_dat line. This bit is set to 1 for reset and released to 0 when completed .mmci_dat finite state machine in both clock domain are also reset. Here below are the registers cleared by the MMCHS_SYSCTL[26] SRD bit: MMCi.MMCHS_DATA MMCi.MMCHS_PSTATE: BRE, BWE, RTA, WTA, DLA and DATI MMCi.MMCHS_HCTL: SBGR and CR MMCi.MMCHS_STAT: BRR, BWR, BGE and TC Interconnect and MMC buffer data management is reinitialized. 0x0: Reset completed 0x1: Software reset for mmci_dat line	RW	0
25	SRC	Software reset for mmci_cmd line. This bit is set to 1 for reset and released to 0 when completed. mmci_cmd finite state machine in both clock domain are also reset. Here below the registers cleared by the MMCi.MMCHS_SYSCTL[25] SRC bit: MMCi.MMCHS_PSTATE: CMDI MMCi.MMCHS_STAT: CC Interconnect and MMC command status management is reinitialized. 0x0: Reset completed 0x1: Software reset for mmci_cmd line	RW	0
24	SRA	Software reset for all. This bit is set to 1 for reset , and released to 0 when completed. This reset affects the entire host controller except for the card detection circuit and capabilities registers. 0x0: Reset completed 0x1: Software reset for all the design	RW	0
23:20	Reserved	Reserved bit field. Do not write any value.	R	0x0
19:16	DTO	Data timeout counter value and busy timeout. This value determines the interval by which mmci_dat lines timeouts are detected. The host driver needs to set this bit field based on - the maximum read access time (NAC) (Refer to the SD Specification Part1 Physical Layer), - the data read access time values (TAAC and NSAC) in the card specific data register (CSD) of the card, - the timeout clock base frequency (MMCi.MMCHS_CAPA[5:0] TCF bits). If the card does not respond within the specified number of cycles, a data timeout error occurs (MMCi.MMCHS_STAT[20] DTO bit). The MMCi.MMCHS_SYSCTL[19,16] DTO bit field is also used to check busy duration, to generate busy timeout for commands with busy response or for busy programming during a write command. Timeout on CRC status is generated if no CRC token is present after a block write. 0x0: TCF x 2^13 0x1: TCF x 2^14	RW	0x0

Bits	Field Name	Description	Type	Reset
		0xE: TCF x 2^27		
		0xF: Reserved		
15:6	CLKD	Clock frequency select These bits define the ratio between a reference clock frequency (system dependant) and the output clock frequency on the mmci_clk pin of either the memory card (MMC, SD or SDIO).	RW	0x000
		0x0: Clock Ref bypass		
		0x1: Clock Ref bypass		
		0x2: Clock Ref / 2		
		0x3: Clock Ref / 3		
		0x3FF: Clock Ref / 1023		
5:3	Reserved	Reserved bit field. Do not write any value	R	0x0
2	CEN	Clock enable. This bit controls if the clock is provided to the card or not.	RW	0
		0x0: The clock is not provided to the card . Clock frequency can be changed .		
		0x1: The clock is provided to the card and can be automatically gated when MMCi.MMCHS_SYSCONFIG[0] AUTOIDLE bit is set to 1 (default value) . The host driver shall wait to set this bit to 1 until the Internal clock is stable (MMCi.MMCHS_SYSCTL[1] ICS bit).		
1	ICS	Internal clock stable (status)This bit indicates either the internal clock is stable or not.	R	0
		Read 0x0: The internal clock is not stable.		
		Read 0x1: The internal clock is stable after enabling the clock (MMCi.MMCHS_SYSCTL[0] ICE bit) or after changing the clock ratio (MMCi.MMCHS_SYSCTL[15:6] CLKD bits).		
0	ICE	Internal clock enable. This register controls the internal clock activity. In very low power state, the internal clock is stopped. Note: The activity of the debounce clock (used for wake-up events) and the interface clock (used for reads and writes to the module register map) are not affected by this register.	RW	0
		0x0: The internal clock is stopped (very low power state).		
		0x1: The internal clock oscillates and can be automatically gated when MMCi.MMCHS_SYSCONFIG[0] AUTOIDLE bit is set to 1 (default value) .		

Table 19-67. Register Call Summary for Register MMCHS_SYSCTL

MMC/SD/SDIO Environment • MMC/SD/SDIO Protocol and Data Format: [0]
MMC/SD/SDIO Integration • Resets: [1] [2] [3] [4] [5]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [6] [7] [8] [9] [10] [11] [12] [13] [14]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29]

Table 19-68. MMCHS_STAT

Address Offset	0x130		
Physical Address	0x4809 C130	**Instance**	MMCHS1
	0x480A D130		MMCHS3
	0x480B 4130		MMCHS2
Description	Interrupt status register The interrupt status regroups all the status of the module internal events that can generate an interrupt. MMCHS_STAT[31:16] = Error Interrupt Status MMCHS_STAT[15:0] = Normal Interrupt Status		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		BADA	CERR	Reserved			ACE	Reserved	DEB	DCRC	DTO	CIE	CEB	CCRC	CTO	ERRI	Reserved					OBI	CIRQ	Reserved		BRR	BWR	Reserved	BGE	TC	CC

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Reserved bit field. Do not write any value	R	0x0
29	BADA	Bad access to data space. This bit is set automatically to indicate a bad access to buffer when not allowed: - During a read access to the data register (MMCi.MMCHS_DATA) while buffer reads are not allowed (MMCi.MMCHS_PSTATE[11] BRE bit =0) - During a write access to the data register (MMCi.MMCHS_DATA) while buffer writes are not allowed (MMCi.MMCHS_PSTATE[10] BWE bit=0) Read 0x0: No Interrupt. Write 0x0: Status bit unchanged Read 0x1: Bad Access Write 0x1: Status is cleared	RW	0
28	CERR	Card error. This bit is set automatically when there is at least one error in a response of type R1, R1b, R6, R5 or R5b. Only bits referenced as type E (error) in status field in the response can set a card status error. An error bit in the response is flagged only if corresponding bit in card status response error MMCi.MMCHS_CSRE in set. There is no card error detection for autoCMD12 command. The host driver shall read MMCi.MMCHS_RSP76 register to detect error bits in the command response. Read 0x0: No Error Write 0x0: Status bit unchanged Read 0x1: Card error Write 0x1: Status is cleared	RW	0
27:25	Reserved	Reserved bit field. Do not write any value	R	0x0
24	ACE	Auto CMD12 error. This bit is set automatically when one of the bits in Auto CMD12 Error status register has changed from 0 to 1. Read 0x0: No Error Write 0x0: Status bit unchanged Read 0x1: AutoCMD12 error Write 0x1: Status is cleared	RW	0
23	Reserved	Reserved bit field. Do not write any value	R	0
22	DEB	Data End Bit error. This bit is set automatically when detecting a 0 at the end bit position of read data on mmci_dat line or at the end position of the CRC status in write mode. Read 0x0: No Error	RW	0

Bits	Field Name	Description	Type	Reset
		Write 0x0: Status bit unchanged		
		Read 0x1: Data end bit error		
		Write 0x1: Status is cleared		
21	DCRC	Data CRC Error. This bit is set automatically when there is a CRC16 error in the data phase response following a block read command or if there is a 3-bit CRC status different of a position "010" token during a block write command.	RW	0
		Read 0x0: No Error.		
		Write 0x0: Status bit unchanged		
		Read 0x1: Data CRC error		
		Write 0x1: Status is cleared		
20	DTO	Data timeout error. This bit is set automatically according to the following conditions: - Busy timeout for R1b, R5b response type. - Busy timeout after write CRC status. - Write CRC status timeout. - Read data timeout.	RW	0
		Read 0x0: No error		
		Write 0x0: Status bit unchanged		
		Read 0x1: Time out		
		Write 0x1: Status is cleared		
19	CIE	Command index error. This bit is set automatically when response index differs from corresponding command index previously emitted. It depends on the enable bit (MMCi.MMCHS_CMD[20] CICE).	RW	0
		Read 0x0: No error		
		Write 0x0: Status bit unchanged		
		Read 0x1: Command index error		
		Write 0x1: Status is cleared		
18	CEB	Command end bit error. This bit is set automatically when detecting a 0 at the end bit position of a command response.	RW	0
		Read 0x0: No error		
		Write 0x0: Status bit unchanged		
		Read 0x1: Command end bit error		
		Write 0x1: Status is cleared		
17	CCRC	Command CRC Error. This bit is set automatically when there is a CRC7 error in the command response depending on the enable bit (MMCi.MMCHS_CMD[19] CCCE).	RW	0
		Read 0x0: No Error		
		Write 0x0: Status bit unchanged		
		Read 0x1: Command CRC error		
		Write 0x1: Status is cleared		
16	CTO	Command Timeout Error. This bit is set automatically when no response is received within 64 clock cycles from the end bit of the command. For commands that reply within 5 clock cycles - the timeout is still detected at 64 clock cycles.	RW	0
		Read 0x0: No error		
		Write 0x0: Status bit unchanged		
		Read 0x1: Time Out		
		Write 0x1: Status is cleared		

Bits	Field Name	Description	Type	Reset
15	ERRI	Error Interrupt. If any of the bits in the Error Interrupt Status register (MMCi.MMCHS_STAT[31:16]) are set, then this bit is set to 1. Therefore the host driver can efficiently test for an error by checking this bit first. Writes to this bit are ignored. Read 0x0: No Interrupt Read 0x1: Error interrupt event(s) occurred	R	0
14:10	Reserved	Reserved bit field. Do not write any value.	R	0x00
9	OBI	Out-Of-Band interrupt (This interrupt is only useful for MMC card). This bit is set automatically when MMCi.MMCHS_CON[14] OBIE bit is set and an Out-of-Band interrupt occurs on OBI pin. The interrupt detection depends on polarity controlled by MMCi.MMCHS_CON[13] OBIP bit. The Out-of-Band interrupt signal is a system specific feature for future use, this signal is not required for existing specification implementation. Read 0x0: No Out-Of-Band interrupt. Write 0x0: Status bit unchanged. Read 0x1: Interrupt Out-Of-Band occurs. Write 0x1: Status is cleared.	R	0
8	CIRQ	Card interrupt. This bit is only used for SD and SDIO cards. In 1-bit mode, interrupt source is asynchronous (can be a source of asynchronous wakeup). In 4-bit mode, interrupt source is sampled during the interrupt cycle. In CE-ATA mode, interrupt source is detected when the card drives mmci_cmd line to zero during one cycle after data transmission end. All modes above are fully exclusive. The controller interrupt must be clear by setting MMCi.MMCHS_IE[8] CIRQ_ENABLE to 0, then the host driver must start the interrupt service with card (clearing card interrupt status) to remove card interrupt source. Otherwise the Controller interrupt will be reasserted as soon as MMCi.MMCHS_IE[8] CIRQ_ENABLE is set to 1. Writes to this bit are ignored. Read 0x0: No card interrupt Read 0x1: Generate card interrupt	R	0
7:6	Reserved	Reserved bit field. Do not write any value.	RW	00
5	BRR	Buffer read ready. This bit is set automatically during a read operation to the card (see class 2 - block oriented read commands) when one block specified by the MMCi.MMCHS_BLK[10:0] BLEN bit field is completely written in the buffer. It indicates that the memory card has filled out the buffer and that the local host needs to empty the buffer by reading it. Note: If the DMA receive-mode is enabled, this bit is never set; instead a DMA receive request to the main DMA controller of the system is generated. Read 0x0: Not Ready to read buffer Write 0x0: Status bit unchanged Read 0x1: Ready to read buffer Write 0x1: Status is cleared	RW	0
4	BWR	Buffer write ready. This bit is set automatically during a write operation to the card (see class 4 - block oriented write command) when the host can write a complete block as specified by MMCi.MMCHS_BLK[10:0] BLEN. It indicates that the memory card has emptied one block from the buffer and that the local host is able to write one block of data into the buffer. Note: If the DMA transmit mode is enabled, this bit is never set; instead, a DMA transmit request to the main DMA controller of the system is generated. Read 0x0: Not Ready to write buffer Write 0x0: Status bit unchanged Read 0x1: Ready to write buffer Write 0x1: Status is cleared	RW	0

Bits	Field Name	Description	Type	Reset
3	Reserved	Reserved bit field. Do not write any value	R	0
2	BGE	Block gap event. When a stop at block gap is requested (MMCi.MMCHS_HCTL[16] SBGR bit), this bit is automatically set when transaction is stopped at the block gap during a read or write operation. This event does not occur when the stop at block gap is requested on the last block. In read mode, a 1-to-0 transition of the mmci_dat line active status (MMCi.MMCHS_PSTATE[2] DLA bit) between data blocks generates a Block gap event interrupt. Read 0x0: No block gap event Write 0x0: Status bit unchanged Read 0x1: Transaction stopped at block gap Write 0x1: Status is cleared	RW	0
1	TC	Transfer completed. This bit is always set when a read/write transfer is completed or between two blocks when the transfer is stopped due to a stop at block gap request (MMCi.MMCHS_HCTL[16] SBGR bit). This bit is also set when exiting a command in a busy state (if the command has a busy notification capability). In Read mode: This bit is automatically set on completion of a read transfer (MMCi.MMCHS_PSTATE[9] RTA bit). In write mode: This bit is set automatically on completion of the mmci_dat line use (MMCi.MMCHS_PSTATE[2] DLA bit). Read 0x0: No transfer complete Write 0x0: Status bit unchanged Read 0x1: Data transfer complete Write 0x1: Status is cleared	RW	0
0	CC	Command complete. This bit is set when a 1-to-0 transition occurs in the register command inhibit (MMCi.MMCHS_PSTATE[0] CMDI bit) If the command is a type for which no response is expected, then the command complete interrupt is generated at the end of the command. A command timeout error (MMCi.MMCHS_STAT[16] CTO bit) has higher priority than command complete (MMCi.MMCHS_STAT[0] CC bit). If a response is expected but none is received, then a command timeout error is detected and signaled instead of the command complete interrupt. Read 0x0: No Command complete Write 0x0: Status bit unchanged Read 0x1: Command complete Write 0x1: Status is cleared	RW	0

Table 19-69. Register Call Summary for Register MMCHS_STAT

MMC/SD/SDIO Integration
- Clocks: [0] [1] [2] [3]
- Interrupt Requests: [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18]

MMC/SD/SDIO Functional Description
- Data Buffer: [19] [20] [21] [22] [23] [24]
- Transfer or Command Status and Errors Reporting: [25]
- MMC CE-ATA Command Completion Disable Management: [26]

MMC/SD/SDIO Use Cases and Tips
- Programming Flow: [27]

MMC/SD/SDIO Register Manual
- MMC/SD/SDIO Registers Mapping Summary: [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56]

Table 19-70. MMCHS_IE

Address Offset	0x134		
Physical Address	0x4809 C134	**Instance**	MMCHS1
	0x480A D134		MMCHS3
	0x480B 4134		MMCHS2
Description	Interrupt SD enable register This register allows to enable/disable the module to set status bits, on an event-by-event basis. MMCHS_IE[31:16] = Error Interrupt Status Enable MMCHS_IE[15:0] = Normal Interrupt Status Enable		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		BADA_ENABLE	CERR_ENABLE	Reserved			ACE_ENABLE	Reserved	DEB_ENABLE	DCRC_ENABLE	DTO_ENABLE	CIE_ENABLE	CEB_ENABLE	CCRC_ENABLE	CTO_ENABLE	NULL	Reserved					OBI_ENABLE	CIRQ_ENABLE	Reserved		BRR_ENABLE	BWR_ENABLE	Reserved	BGE_ENABLE	TC_ENABLE	CC_ENABLE

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Reserved bit field. Do not write any value.	R	0
29	BADA_ENABLE	Bad access to data space Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
28	CERR_ENABLE	Card error interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
27:25	Reserved	Reserved bit field. Do not write any value	R	0x0
24	ACE_ENABLE	Auto CMD12 error Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
23	Reserved	Reserved bit field. Do not write any value	R	0
22	DEB_ENABLE	Data end bit error Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
21	DCRC_ENABLE	Data CRC error Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
20	DTO_ENABLE	Data timeout error Interrupt Enable	RW	0
		0x0: The data timeout detection is deactivated. The host controller provides the clock to the card until the card sends the data or the transfer is aborted.		
		0x1: The data timeout detection is enabled.		
19	CIE_ENABLE	Command index error Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
18	CEB_ENABLE	Command end bit error Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		
17	CCRC_ENABLE	Command CRC error Interrupt Enable	RW	0
		0x0: Masked		
		0x1: Enabled		

Bits	Field Name	Description	Type	Reset
16	CTO_ENABLE	Command timeout error Interrupt Enable 0x0: Masked 0x1: Enabled	RW	0
15	NULL	Fixed to 0 The host driver shall control error interrupts using the Error Interrupt Signal Enable register. Writes to this bit are ignored.	R	0
14:10	Reserved	Reserved bit field. Do not write any value.	R	0x00
9	OBI_ENABLE	Out-of-Band interrupt Enable A write to this register when MMCi.MMCHS_CON[14] OBIE is set to '0' is ignored. 0x0: Masked 0x1: Enabled	RW	0
8	CIRQ_ENABLE	Card interrupt Enable A clear of this bit also clears the corresponding status bit. During 1-bit mode, if the interrupt routine does not remove the source of a card interrupt in the SDIO card, the status bit is reasserted when this bit is set to 1. This bit must be set to 1 when entering in smart idle mode to enable system to identity wakeup event and to allow controller to clear internal wakeup source. 0x0: Masked 0x1: Enabled	RW	0
7:6	Reserved	Reserved bit field. Do not write any value	RW	00
5	BRR_ENABLE	Buffer Read Ready Interrupt Enable 0x0: Masked 0x1: Enabled	RW	0
4	BWR_ENABLE	Buffer Write Ready Interrupt Enable 0x0: Masked 0x1: Enabled	RW	0
3	Reserved	Reserved bit field. Do not write any value.	R	0
2	BGE_ENABLE	Block Gap Event Interrupt Enable 0x0: Masked 0x1: Enabled	RW	0
1	TC_ENABLE	Transfer completed Interrupt Enable 0x0: Masked 0x1: Enabled	RW	0
0	CC_ENABLE	Command completed Interrupt Enable 0x0: Masked 0x1: Enabled	RW	0

Table 19-71. Register Call Summary for Register MMCHS_IE

MMC/SD/SDIO Integration
- Clocks: [0] [1]
- Interrupt Requests: [2] [3] [4] [5]

MMC/SD/SDIO Use Cases and Tips
- Programming Flow: [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]

MMC/SD/SDIO Register Manual
- MMC/SD/SDIO Registers Mapping Summary: [21] [22] [23] [24] [25] [26]

Table 19-72. MMCHS_ISE

Address Offset	0x138		
Physical Address	0x4809 C138	**Instance**	MMCHS1
	0x480A D138		MMCHS3
	0x480B 4138		MMCHS2
Description	Interrupt signal enable register This register allows to enable/disable the module internal sources of status, on an event-by-event basis. MMCHS_ISE[31:16] = Error Interrupt Signal Enable MMCHS_ISE[15:0] = Normal Interrupt Signal Enable		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		BADA_SIGEN	CERR_SIGEN	Reserved			ACE_SIGEN	Reserved	DEB_SIGEN	DCRC_SIGEN	DTO_SIGEN	CIE_SIGEN	CEB_SIGEN	CCRC_SIGEN	CTO_SIGEN	NULL	Reserved						OBI_SIGEN	CIRQ_SIGEN	Reserved	Reserved	BRR_SIGEN	BWR_SIGEN	Reserved	BGE_SIGEN	TC_SIGEN	CC_SIGEN

Bits	Field Name	Description	Type	Reset
31:30	Reserved	Reserved bit field. Do not write any value.	R	0
29	BADA_SIGEN	Bad access to data space signal status Enable 0x0: Masked 0x1: Enabled	RW	0
28	CERR_SIGEN	Card error interrupt signal status Enable 0x0: Masked 0x1: Enabled	RW	0
27:25	Reserved	Reserved bit field. Do not write any value	R	0x0
24	ACE_SIGEN	Auto CMD12 error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
23	Reserved	Reserved bit field. Do not write any value	R	0
22	DEB_SIGEN	Data end bit error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
21	DCRC_SIGEN	Data CRC error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
20	DTO_SIGEN	Data timeout error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
19	CIE_SIGEN	Command index error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
18	CEB_SIGEN	Command end bit error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
17	CCRC_SIGEN	Command CRC error signal status Enable 0x0: Masked 0x1: Enabled	RW	0
16	CTO_SIGEN	Command timeout error signal status Enable 0x0: Masked	RW	0

Bits	Field Name	Description	Type	Reset
		0x1: Enabled		
15	NULL	Fixed to 0 The host driver shall control error interrupts using the Error Interrupt Signal Enable register. Writes to this bit are ignored	R	0
14:10	Reserved	Reserved bit field. Do not write any value	R	0x00
9	OBI_SIGEN	Out-Of-Band Interrupt signal status Enable. A write to this register when MMCHS_CON[14] OBIE bit is set to '0' is ignored. 0x0: Masked 0x1: Enabled	RW	0
8	CIRQ_SIGEN	Card interrupt signal status Enable 0x0: Masked 0x1: Enabled	RW	0
7	Reserved	Reserved bit field. Do not write any value	RW	0
6	Reserved	Reserved bit field. Do not write any value	RW	0
5	BRR_SIGEN	Buffer Read Ready signal status Enable 0x0: Masked 0x1: Enabled	RW	0
4	BWR_SIGEN	Buffer Write Ready signal status Enable 0x0: Masked 0x1: Enabled	RW	0
3	Reserved	Reserved bit field. Do not write any value	R	0
2	BGE_SIGEN	Black Gap Event signal status Enable 0x0: Masked 0x1: Enabled	RW	0
1	TC_SIGEN	Transfer completed signal status Enable 0x0: Masked 0x1: Enabled	RW	0
0	CC_SIGEN	Command completed signal status Enable 0x0: Masked 0x1: Enabled	RW	0

Table 19-73. Register Call Summary for Register MMCHS_ISE

MMC/SD/SDIO Integration
- Clocks: [0] [1]
- Interrupt Requests: [2] [3] [4] [5] [6]

MMC/SD/SDIO Use Cases and Tips
- Programming Flow: [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21]

MMC/SD/SDIO Register Manual
- MMC/SD/SDIO Registers Mapping Summary: [22] [23] [24] [25]

Table 19-74. MMCHS_AC12

Address Offset	0x13C		
Physical Address	0x4809 C13C	**Instance**	MMCHS1
	0x480A D13C		MMCHS3
	0x480B 413C		MMCHS2
Description	Auto CMD12 Error Status Register. The host driver may determine which of the errors cases related to Auto CMD12 has occurred by checking this MMCi.MMCHS_AC12 register when an Auto CMD12 Error interrupt occurs. This register is valid only when Auto CMD12 is enabled (MMCi.MMCHS_CMD[2] ACEN bit) and Auto CMD12Error (MMCi.MMCHS_STAT[24] ACE bit) is set to 1. Note: These bits are automatically reset when starting a new adtc command with data.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								CNI	Reserved		ACIE	ACEB	ACCE	ACTO	ACNE

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Reserved bit field. Do not write any value.	R	0x0
7	CNI	Command not issue by Auto CMD12 error. If this bit is set to 1, it means that pending command is not executed due to Auto CMD12 error: ACEB, ACCE, ACTO or ACNE. Read 0x0: Not error Read 0x1: Command not issued	R	0
6:5	Reserved	Reserved bit field. Do not write any value.	R	0x0
4	ACIE	Auto CMD12 index error. This bit is a set to 1 when response index differs from corresponding command auto CMD12 index previously emitted. This bit depends on the command index check enable (MMCi.MMCHS_CMD[20] CICE bit). Read 0x0: No error Read 0x1: Auto CMD12 Index Error	R	0
3	ACEB	Auto CMD12 end bit error. This bit is set to 1 when detecting a 0 at the end bit position of auto CMD12 command response. Read 0x0: No error Read 0x1: AutoCMD12 End bit Error	R	0
2	ACCE	Auto CMD12 CRC error. This bit is automatically set to 1 when a CRC7 error is detected in the auto CMD12 command response depending on the enable in the MMCi.MMCHS_CMD[19] CCCE bit. Read 0x0: No error Read 0x1: Auto CMD12 CRC Error	R	0
1	ACTO	Auto CMD12 timeout error. This bit is set to 1 if no response is received within 64 clock cycles from the end bit of the auto CMD12 command. Read 0x0: No error Read 0x1: Auto CMD12 Time Out	R	0
0	ACNE	Auto CMD12 not executed. This bit is set to 1 if multiple block data transfer command has started and if an error occurs in command before Auto CMD12 starts. Read 0x0: Auto CMD12 Executed Read 0x1: Auto CMD12 Not Executed	R	0

Table 19-75. Register Call Summary for Register MMCHS_AC12

MMC/SD/SDIO Register Manual

- MMC/SD/SDIO Registers Mapping Summary: [0] [1] [2]

Table 19-76. MMCHS_CAPA

Address Offset	0x140		
Physical Address	0x4809 C140	**Instance**	MMCHS1
	0x480A D140		MMCHS3
	0x480B 4140		MMCHS2
Description	Capabilities register. This register lists the capabilities of the MMC/SD/SDIO host controller.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					VS18	VS30	VS33	SRS	DS	HSS	Reserved			MBL		Reserved		BCF						TCU	Reserved	TCF					

Bits	Field Name	Description	Type	Reset
31:27	Reserved	Reserved bit field. Do not write any value.	R	0x00
26	VS18	Voltage support 1.8V Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via MMCi_RESET signal). Read 0x0: 1.8 V Not Supported Write 0x0: 1.8 V Not supported Read 0x1: 1.8 V Supported Write 0x1: 1.8 V Supported MMCHS1, 2 and 3: This bit must be set to 1.	RW	0
25	VS30	Voltage support 3.0V Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via MMCi_RESET signal) Read 0x0: 3.0 V Not Supported Write 0x0: 3.0 V Not supported Read 0x1: 3.0 V Supported Write 0x1: 3.0 V Supported MMCHS1: This bit must be set to 1. MMCHS2 and 3: This bit must be left to 0.	RW	0
24	VS33	Voltage support 3.3V Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via MMCi_RESET signal) Read 0x0: 3.3 V Not Supported Write 0x0: 3.3 V Not supported Read 0x1: 3.3 V Supported Write 0x1: 3.3 V Supported MMCHS1, 2 and 3: This bit must be left to 0.	RW	0
23	SRS	Suspend/Resume support (SDIO cards only)This bit indicates whether the host controller supports Suspend/Resume functionality. Read 0x0: The Host controller does not Suspend/Resume functionality. Read 0x1: The Host controller supports Suspend/Resume functionality.	R	1
22	DS	DMA support. This bit indicates that the Host Controller is able to use DMA to transfer data between system memory and the Host Controller directly. Read 0x0: DMA Not Supported Read 0x1: DMA Supported	R	1

Bits	Field Name	Description	Type	Reset
21	HSS	High speed support. This bit indicates that the host controller supports high speed operations and can supply an up-to-52 MHz clock to the card. Read 0x0: High Speed Not Supported Read 0x1: High Speed Supported	R	1
20:18	Reserved	Reserved bit field. Do not write any value.	R	0x0
17:16	MBL	Maximum block length. This value indicates the maximum block size that the host driver can read and write to the buffer in the host controller. The host controller supports 512 bytes and 1024 bytes block transfers. Read 0x0: 512 bytes Read 0x1: 1024 bytes Read 0x2: 2048 bytes	R	0x1
15:14	Reserved	Reserved bit field. Do not write any value	R	0x0
13:8	BCF	Base clock frequency for clock provided to the card. Read 0x0: The value indicating the base (maximum) frequency for the output clock provided to the card is system dependent and is not available in this register. Get the information via another method. See , *Power Reset and Clock Management*, for more information on the value of FUNC_96M_CLK clock signal.	R	0x00
7	TCU	Timeout clock unit. This bit shows the unit of base clock frequency used to detect Data Timeout Error (MMCi.MMCHS_STAT[20] DTO bit). Read 0x0: kHz Read 0x1: MHz	R	1
6	Reserved	Reserved. This bit is initialized to zero, and writes to it are ignored.	R	0
5:0	TCF	Timeout clock frequency. The timeout clock frequency is used to detect Data Timeout Error (MMCi.MMCHS_STAT[20] DTO bit). Read 0x0: The timeout clock frequency depends on the frequency of the clock provided to the card. The value of the timeout clock frequency is not available in this register. **Note:** You can have the timeout clock frequency by dividing FUNC_96M_CLK by the value of the MMCi.MMCHS_SYSCTL[15:6] CLKD bit field.	R	0x00

Table 19-77. Register Call Summary for Register MMCHS_CAPA

MMC/SD/SDIO Integration • Resets: [0]
MMC/SD/SDIO Functional Description • Data Buffer: [1]
MMC/SD/SDIO Basic Programming Model • Set MMCHS Default Capabilities: [2]
MMC/SD/SDIO Use Cases and Tips • Programming Flow: [3] [4] [5]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [6] [7] [8] [9] [10] [11] [12]

Table 19-78. MMCHS_CUR_CAPA

Address Offset	0x148		
Physical Address	0x4809 C148	**Instance**	MMCHS1
	0x480A D148		MMCHS3
	0x480B 4148		MMCHS2
Description	Maximum current capabilities Register. This register indicates the maximum current capability for each voltage. The value is meaningful if the voltage support is set in the capabilities register (MMCi.MMCHS_CAPA). Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via MMCi_RESET signal)		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CUR_1V8								CUR_3V0								CUR_3V3							

Bits	Field Name	Description	Type	Reset
31:24	Reserved	Reserved. This bit is initialized to zero, and writes to it are ignored.	R	0x0
23:16	CUR_1V8	Maximum current for 1.8 V	RW	0x0
		Read 0x0 The maximum current capability for this voltage is not available. Feature not implemented.		
15:8	CUR_3V0	Maximum current for 3.0 V	RW	0x0
		Read 0x0 The maximum current capability for this voltage is not available. Feature not implemented.		
7:0	CUR_3V3	Maximum current for 3.3 V	RW	0x0
		Read 0x0 The maximum current capability for this voltage is not available. Feature not implemented.		

Table 19-79. Register Call Summary for Register MMCHS_CUR_CAPA

MMC/SD/SDIO Integration • Resets: [0]
MMC/SD/SDIO Basic Programming Model • Set MMCHS Default Capabilities: [1]
MMC/SD/SDIO Register Manual • MMC/SD/SDIO Registers Mapping Summary: [2]

Table 19-80. MMCHS_REV

Address Offset	0x1FC		
Physical Address	0x4809 C1FC	**Instance**	MMCHS1
	0x480A D1FC		MMCHS3
	0x480B 41FC		MMCHS2
Description	Versions Register. This register contains the hard coded RTL vendor revision number, the version number of SD specification compliancy and a slot status bit. MMCi.MMCHS_REV[31:16] = Host controller version MMCi.MMCHS_REV[15:0] = Slot Interrupt Status		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREV								SREV								Reserved															SIS

Bits	Field Name	Description	Type	Reset
31:24	VREV	Vendor Version Number: IP revision [31:28] Major revision [27:24] Minor revision Examples: 0x10 for 1.0 0x21 for 2.1	R	See (1)
23:16	SREV	Specification Version Number. This status indicates the Standard SD Host Controller Specification Version. The upper and lower 4-bits indicate the version. Read 0x0: SD Host Specification Version 1.0	R	0x00
15:1	Reserved	Reserved bit field. Do not write any value	R	0x0000
0	SIS	Slot Interrupt Status. This status bit indicates the inverted state of interrupt signal for the module. By a power on reset or by setting a software reset for all (MMCi.MMCHS_SYSCTL[24] SRA), the interrupt signal shall be deasserted and this status shall read 0.	R	0

(1) TI internal data

Table 19-81. Register Call Summary for Register MMCHS_REV

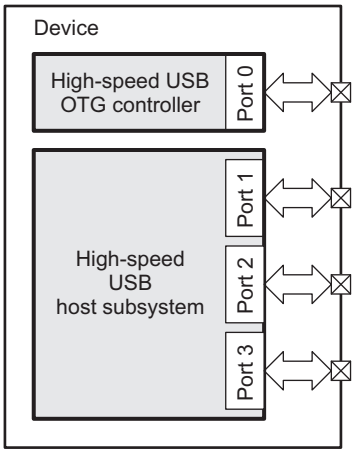
MMC/SD/SDIO Register Manual

- MMC/SD/SDIO Registers Mapping Summary: [0] [1] [2]

Universal Serial Bus (USB)

This chapter describes the high-speed USB On-The-Go (OTG) controller and the high-speed universal serial bus (USB) host subsystem.

Figure 20-1. USB Modules Overview



Topic **Page**

20.1 High-Speed USB OTG Controller

NOTE: The High-Speed USB OTG Controller is an instantiation of the MUSBMHDRC from Mentor Graphics Corporation.

Mentor Graphics is a registered trademark of Mentor Graphics Corporation or its affiliated companies in the United States and other countries.

The High-Speed USB OTG Controller is an instantiation of the MUSBMHDRC from Mentor Graphics Corporation.

20.1.1 Introduction

This chapter describes the high-speed universal serial bus (USB) host subsystem and the high-speed USB On-The-Go (OTG) controller. The controller complies with the USB 2.0 standard high-speed and full-speed functions and low-speed, full-speed, and high-speed limited host mode operations. It also includes support for the Session Request and Host Negotiation Protocols used in point-to-point communications, details of which are given in the USB On-the-Go supplement to the USB 2.0 specification. In addition, the four test modes for high-speed operation described in the USB 2.0 specification. It also allows options that allow it to be forced into full-speed mode, high-speed mode or host mode which may be used in helping debug PHY problems in hardware.

20.1.1.1 Purpose of the Peripheral

The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices and also supports host negotiation.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, and your device-specific data manual.

20.1.2 Features Supported

- Contains 1 usb20otg_f modules, with the following features:
  - Built around the Mentor USB 2.0 OTG core (musbmhdrc)
  - Supports USB 2.0 peripheral at speeds HS (480 Mb/s) and FS (12 Mb/s)
  - Supports USB 2.0 Host or OTG at speeds HS (480 Mb/s), FS (12 Mb/s), and LS (1.5 Mb/s)
  - Supports all modes of transfers (control, bulk, interrupt, and isochronous)
  - Supports high bandwidth ISO mode
  - Supports 16 transmit (TX) and 16 receive (RX) endpoints including endpoint 0
  - Supports USB OTG extensions for session resume (SRP) and Host negotiation (HNP)
  - Includes a 32K endpoint FIFO RAM, and supports programmable FIFO sizes
  - Includes RNDIS mode for accelerating RNDIS type protocols using short packet termination over USB
  - Includes CDC Linux mode for accelerating CDC type protocols using short packet termination over USB
  - Included and RNDIS like mode for terminating RNDIS type protocols without using short packet termination for support of MSC applications
- Includes one USB2.0 OTG PHYs in GS60 process
- Connects to a standard charge pump for VBUS 5 V generation
- Interfaces to the CPU through a CBA 3.1 VBUSP slave port
- Includes a CPPI 4.1 compliant DMA controller sub-module with 15 RX and 15 TX simultaneous data connections
- Includes a CPPI 4.1 DMA scheduler

- DMA supports CPPI Host descriptor formats
- DMA supports stall on buffer starvation
- Supports data buffer sizes up to 4M bytes
- CPPI FIFO interface per TX/RX endpoint
- Provides a CPPI queue manager module with 96 queues for queuing/dequeuing packets

20.1.2.1 Features Not Supported

- 16-bit 30 MHz UTMI+ interface is not supported
- RNDIS, CDC, and Generic RNDIS mode acceleration for USB sizes that are not multiples of 64 bytes
- Endpoint max USB packet sizes that do not conform to the USB 2.0 spec (for FS/LS: 8, 16, 32, 64, and 1023 are defined; for HS: 64, 128, 512, and 1024 are defined)

20.1.3 Functional Description

20.1.3.1 Compliance to Standards

The module is compliant to the USB 2.0 specification as well as the OTG supplement. Refer to the Mentor specification for details.

20.1.3.2 Functional Operation

20.1.3.2.1 Overview

The USB 2.0 OTG subsystem consists of a CPPI 4.1 DMA controller and scheduler, an INTD interrupt distributer, a queue manager, a usb20otg_f USB controller subsystem, and a USB OTG PHY.

The USB 2.0 OTG F controller consists of the Mentor core, a packet processing unit, a switched central resource (SCR), and the bridge sub-modules. The core interfaces to the UTMI+ interface that is connected to a UTMI+ PHY. It also connects to the endpoint FIFO RAM. The module interfaces to a CPPI 4.1 compliant DMA controller through the Egress/Ingress CPPI FIFO interfaces in the PPU. The bridge sub-modules consist of additional new functions needed to connect the Mentor core and make the module PDR 3.5 compliant. The VBUSP to AHB (V2A) sub-module converts to the core's AHB slave port. The transfer (XFER) DMA sub-module controls transferring data from the endpoint FIFOs to the CPPI FIFOs. The interrupt sub-module implements PDR 3.5 compliant interrupts. The clock sub-module implements the CBA 3.1 compliant clocking protocols. The test sub-module implements the PDR 3.5 I/O test requirements. VBUSP retiming sub-modules are used to enhance VBUSP timing. And the SCR connects all the sub-modules together with the DMA state RAM. Figure 20-2 shows the top level block diagram of the sub-modules and their connections. Figure 20-3 shows the USB20OTG_F block diagram.

Figure 20-2. USB Subsystem Block Diagram

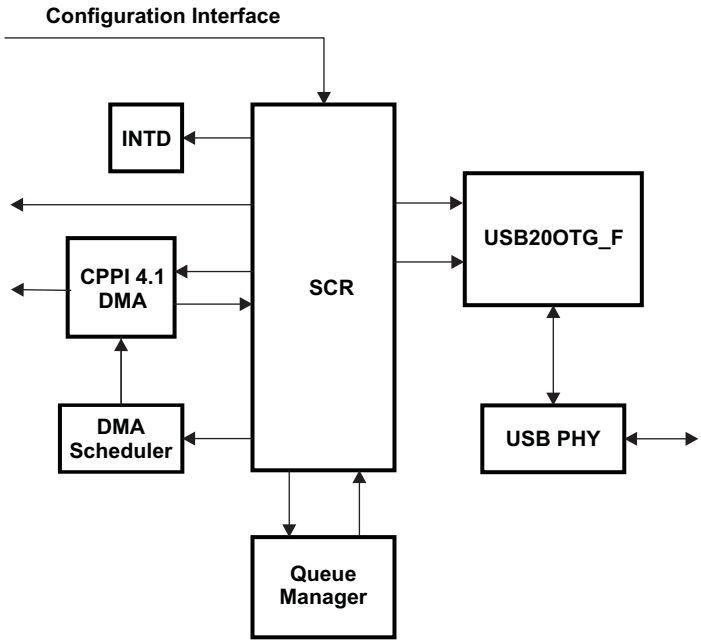
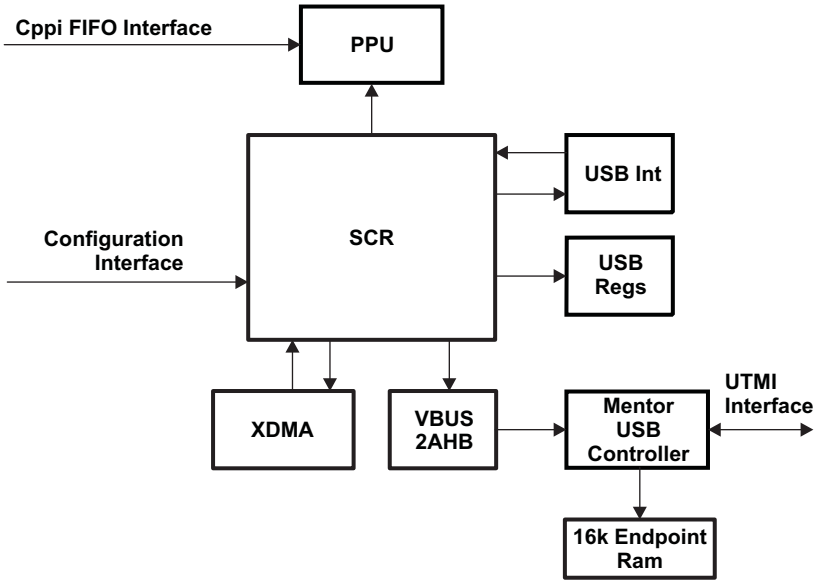


Figure 20-3. USB20OTG_F Block Diagram



20.1.3.2.2 Mentor Core

The core of the USB 2.0 OTG controller is the Mentor USB 2.0 high-speed multi-point dual-role controller, musbmhdrc. This core is IP from Mentor and a detailed description of its operations is covered by the Mentor specification.

The Mentor core communicates to a UTMI+ PHY interface, an AHB slave interface, and a FIFO RAM interface, along with extra system signals. The connections for these interfaces and signals are described in Section 20.1.4.3.

20.1.3.2.3 USB OTG PHY

The USB OTG Subsystem integrates a USB OTG PHY. The PHY interfaces to the Mentor OTG controller via the UTMI+ interface.

The integrated PHY has the following features:

- OTG compliant.
- Has UTMI-plus Level 3 functionality. (Host and OTG devices. HS/FS/LS and preamble packet.)
- Supports OTG, SRP and HNP protocols.
- 8-bit data interface on the UTMI with 60 MHz clock

The integrated PHY does not contain a charge pump, so it will not provide power to VBUS.

20.1.3.2.4 CPPI 4.1 DMA Controller

The CPPI DMA module supports the transmission and reception of USB packets. The CPPI DMA is designed to facilitate the segmentation and reassembly of CPPI compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each networking port. Multiple Tx and Rx channels are provided within the DMA that allow multiple segmentation or reassembly operations to be effectively performed in parallel (but not actually simultaneously). The DMA controller maintains state information for each of the channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs. The DMA controller has its mem1 port brought out to the top level of the subsystem as a VBUSP master interface. This is to allow simultaneous assess by the DMA and the Queue manager to the external switch.

20.1.3.2.4.1 Tx Channel Allocation

A total of 30 Tx channels are provided within the DMA for concurrent packet transfers to the various ports. Table 20-1 shows how the Tx channels are allocated.

Table 20-1. TX Channel Allocation

DMA Channel	Function	Channel Type	Port	Channel In Port	Tx Queue Count	Tx Queue Arbitration Type	Start Tx QM # : Queue #	Packet Type(s)
0	USB EP 1	Endpt	1	0	2	Fixed Priority	32	USB
1	USB EP 2	Endpt	2	0	2	Fixed Priority	34	USB
2	USB EP 3	Endpt	3	0	2	Fixed Priority	36	USB
3	USB EP 4	Endpt	4	0	2	Fixed Priority	38	USB
4	USB EP 5	Endpt	5	0	2	Fixed Priority	40	USB
5	USB EP 6	Endpt	6	0	2	Fixed Priority	42	USB
6	USB EP 7	Endpt	7	0	2	Fixed Priority	44	USB
7	USB EP 8	Endpt	8	0	2	Fixed Priority	46	USB
8	USB EP 9	Endpt	9	0	2	Fixed Priority	48	USB
9	USB EP 10	Endpt	10	0	2	Fixed Priority	50	USB
10	USB EP 11	Endpt	11	0	2	Fixed Priority	52	USB
11	USB EP 12	Endpt	12	0	2	Fixed Priority	54	USB
12	USB EP 13	Endpt	13	0	2	Fixed Priority	56	USB
13	USB EP 14	Endpt	14	0	2	Fixed Priority	58	USB
14	USB EP 15	Endpt	15	0	2	Fixed Priority	60	USB

20.1.3.2.4.2 CPPI DMA Transmit Operation

Packet transmission is accomplished within the DMA by moving data from structures that are located in external memory into the PPU module of the USB Subsystem. This movement of data is performed in blocks whose size varies based on the type of each packet and is performed in compliance with the CPPI 4.1 specification. USB packets are fetched by the Tx DMA in 64 byte blocks.

The DMA performs the following sequence on the Tx Port FIFO interface for each cell that is transferred:

1. Transfer specified number of payload bytes (1 to 64) to the Tx Port FIFO.
2. Write the channel number, packet type, SOP, EOP, and valid cell byte count to the cell information register in the Tx Port FIFO. The cell byte count is calculated by the DMA as the number of valid payload bytes in the cell for this packet type. Valid cell byte counts range from 1 to 64.
3. Write to the cell advance register in the Tx Port FIFO.

20.1.3.2.4.3 Rx Channel Allocation

A total of 30 Rx channels are provided within the DMA for concurrent packet transfers from the various ports. Table 20-2 shows how the Rx channels are allocated.

Table 20-2. RX Channel Allocation

DMA Channel	Function	Channel Type	Src Tag Port	Src Tag Channel	Src Tag Sub-Channel	Rx Queue Count	Packet Type(s)
0	USB EP 1	Endpt	1	0	0	1	USB
1	USB EP 2	Endpt	2	0	0	1	USB
2	USB EP 3	Endpt	3	0	0	1	USB
3	USB EP 4	Endpt	4	0	0	1	USB
4	USB EP 5	Endpt	5	0	0	1	USB
5	USB EP 6	Endpt	6	0	0	1	USB
6	USB EP 7	Endpt	7	0	0	1	USB
7	USB EP 8	Endpt	8	0	0	1	USB
8	USB EP 9	Endpt	9	0	0	1	USB
9	USB EP 10	Endpt	10	0	0	1	USB
10	USB EP 11	Endpt	11	0	0	1	USB
11	USB EP 12	Endpt	12	0	0	1	USB
12	USB EP 13	Endpt	13	0	0	1	USB
13	USB EP 14	Endpt	14	0	0	1	USB
14	USB EP 15	Endpt	15	0	0	1	USB

20.1.3.2.4.4 CPPI DMA Receive Operation

Packet reception is accomplished within the DMA by moving data from Rx Port FIFOs within the Jupiter session router subsystem into structures that are located in external memory. This movement of data is performed in blocks whose size varies based on the type of each packet and is in compliance with the CPPI 4.1 specification. USB packets are stored by the Rx DMA in 64 byte blocks.

The DMA performs the following sequence on the Rx Port FIFO interface of the PPU for each cell that is transferred:

1. Read the channel number, packet type, SOP, EOP, and valid cell byte count from the cell information registers in the Rx Port FIFO.
2. Transfer specified number of payload bytes (1 to 64) from the Rx Port FIFO.
3. Write to the cell advance register in the Rx Port FIFO.

20.1.3.2.5 CPPI DMA Scheduler

The CPPI DMA scheduler is responsible for controlling the rate and order between the different Tx and Rx threads that are provided in the CPPI DMA controller. The following sections describe the details of the scheduler functionality. The scheduler table RAM is a two port memory which is organized as 64 words with storage for 4 scheduler entries in each word.

20.1.3.2.5.1 Scheduler Initialization

Before the scheduler can be used, the Host is required to initialize and enable the block. This initialization is as follows:

- The Host initializes entries within an internal memory array in the scheduler. This array contains up to 256 entries and each entry consists of a DMA channel number and a bit indicating if this is a Tx or Rx opportunity. These entries represent both the order and frequency that various Tx and Rx channels will be processed. A table size of 256 entries allows channel bandwidth to be allocated with a maximum precision of 1/256th of the total DMA bandwidth. The more entries that are present for a given channel, the bigger the slice of the bandwidth that channel will be given. Larger tables can be accommodated to allow for more precision. This array can only be written by the Host, it cannot be read.
- The Host initializes an internal register in the scheduler that sets the actual size of the array; it can be less than 256 entries.
- The Host writes an internal register bit to enable the scheduler. The scheduler is not required to be disabled in order to change the scheduler array contents.

20.1.3.2.5.2 Scheduler Operation

Once the scheduler is enabled, it begins to process the entries in the table, and when appropriate, pass credits to the DMA controller to perform a Tx or Rx operation. The operation of the DMA controller is as follows:

1. After the DMA scheduler is enabled, it begins with the table index set to 0.
2. The scheduler reads the entry pointed to by the index and checks to see if the channel in question is currently in a state where a DMA operation can be accepted.
   - The DMA channel must be enabled, and
   - The CPPI FIFO that the channel talks to has free space on TX (FIFO full signal is not asserted) or a valid block on Rx (FIFO empty signal is not asserted).
3. If the DMA channel is capable of processing a credit to transfer a block, the DMA scheduler will issue that credit via the DMA scheduling interface, which is a point-to-point connection between the DMA scheduler and the DMA controller.
   - The DMA controller may not be ready to accept the credit immediately and is provided a sched_ready signal which is used to stall the scheduler until it can accept the credit. The DMA controller only asserts the sched_ready signal when it is in the IDLE state.
   - Once a credit has been accepted (indicated by sched_req and sched_ready both asserted), the scheduler will increment the index to the next entry and will start at step 2.
4. If the channel in question is not currently capable of processing a credit, the scheduler will increment the index in the scheduler table to the next entry and will start at step 2.
5. Note that when the scheduler attempts to increment its index to the value programmed in the table size register, the index will reset to 0.

20.1.3.2.6 CP_INTD

The cp_intd module controls the distribution of the USB subsystem interrupts to a single or multiple Hosts in a SoC. It combines the functions of many previous pieces of support IP into a single configurable module that allows a central method to control IP interrupts and system interrupts as well as complete customization to a particular SoC's need. The module eliminates the confusion of handling IP interrupts from a PDR IP module without knowledge of the support IP by building interfaces that connect directly to the IP and Interrupt controllers for each Host leaving the SoC to just connect the wires.

The cp_intd encompasses six functions that used to be separate (if even available) pieces of support IP or additional glue logic at the SoC level. These functions are: enabling, status capture, counting, combining or aggregation, pacing and synchronization. There are also polarity and type conversion and EOI functions to support legacy IP. The descriptions of these functions are detailed in the CP_INTD module specification.

Mapping of the USB interrupts to the cp_intd controller is described in Section 20.1.4.2 of this document.

20.1.3.2.7 CPPI Queue Manager

The queue manager accelerates addition and deletion of descriptors from descriptor queues, tracking of queue emptiness, and queue diversion. The following sections describe various operations that the queue manager can do on the queues. See the CPPI 4.1 specification for more information on queues. The Queue Manager module in the USB subsystem provides 96 hardware queues.

Table 20-3. Queue Allocation

Start	Count	Name
0	15	Free Descriptor/Buffer Queues
16	16	Unassigned Queues
32	2	USB0 Endpoint 1 Transmit Queue
34	2	USB0 Endpoint 2 Transmit Queue
36	2	USB0 Endpoint 3 Transmit Queue
38	2	USB0 Endpoint 4 Transmit Queue
40	2	USB0 Endpoint 5 Transmit Queue
42	2	USB0 Endpoint 6 Transmit Queue
44	2	USB0 Endpoint 7 Transmit Queue
46	2	USB0 Endpoint 8 Transmit Queue
48	2	USB0 Endpoint 9 Transmit Queue
50	2	USB0 Endpoint 10 Transmit Queue
52	2	USB0 Endpoint 11 Transmit Queue
54	2	USB0 Endpoint 12 Transmit Queue
56	2	USB0 Endpoint 13 Transmit Queue
58	2	USB0 Endpoint 14 Transmit Queue
60	2	USB0 Endpoint 15 Transmit Queue
62	2	USB1 Endpoint 1 Transmit Queue
64	30	Unassigned
92	4	Completion Queues

20.1.3.2.8 USB20OTG_F Packet Processing Unit

The packet processing unit (PPU) provides an interface and FIFO storage between the CPPI 4.1 DMA controller and the XDMA. The PPU interfaces to the DMA controller via a standard CFIFO interface. Full/empty flags are asserted to the DMA and XDMA depending on cell availability in the PPU FIFO. The PPU contains 30 logical FIFOs with a depth of 2 64-bytes cells each. One FIFO is allocated to each RX and TX channel and cannot be changed.

20.1.3.2.9 USB20OTG_F VBUSP to AHB (V2A)

The core is accessed through an AHB slave interface. There are a few operations that need to access this port: a CPU access from the VBUSP slave port, a register read by the Interrupt sub-module, or a FIFO access by the XFER DMA controller. The V2A will convert the VBUSP slave bus from the SCR to the AHB slave bus on the core. The AHB interface is a split transaction bus, with an address phase followed by a data phase, and successive transfers can overlap their address phases with the previous data phase (see the ARM AMBA Specification for more information on the AHB bus). The VUBSP interface is a single phase bus, so the address is presented until the slave is ready with the data (see the CBA 3.0 Specification for more information on the VBUSP bus). The V2A performs the two AHB phases while holding off the single phase on the VBUSP. When the second phase on the AHB is ready, then the VBUSP is ready and advances to the next phase. This allows a very simple implementation with minimal logic gates. For typical CPU accesses without bursts, there is no way to achieve better performance, and the DMA performance to the endpoint FIFOs should be sufficient for the required bandwidth.

The core is set to big endian mode to match the DMA expectation of data in the FIFOs. The DMA performs endian adjustment to match the system endian specified by the big_endian signal. To match this behavior, accesses to the core from the CPU through the VBUSP slave input will also adjust the data based on the system endian.

The V2A bridge as well as the VBUSP slave port only support bursts in linear incrementing mode. Any other modes cannot be used with the USB 2.0 OTG module.

20.1.3.2.10 USB20OTG_F Transfer (XFER) DMA

The XFER DMA receives DMA requests from the core and initiates the DMAs to the PPU. The first step is receiving the DMA request from the core. The core sends out 30 DMA_req signals, 15 for TX endpoints 1 to 15, and 15 for RX endpoints 1 to 15. For TX endpoints, the DMA_req goes high when a TX buffer for that endpoint is empty and can hold new data. The CPPI DMA performs the requested DMA for that channel, and the PPU indicates to the XDMA that data is available for that channel. Then, the XFER DMA reads from the PPU FIFO and writes the data into the endpoint FIFO inside the core. When the TX buffer in the core is full, the DMA_req goes low. After all the data is written to the endpoint FIFO, the XFER DMA writes the trigger bit in the core endpoint register to enable that TX endpoint for transfer.

For RX endpoints, the DMA_req goes high when an RX buffer holds a completed USB packet. The XFER DMA must check if the PPU FIFO is empty for that RX channel. If the PPU FIFO is empty on that RX channel, then the XFER DMA reads the RX count register in the core to obtain the size of the RX data, and reads the data from the core endpoint FIFOs and writes it to the PPU FIFO. The PPU FIFO signals the CPPI DMA that data is available. Then, the CPPI DMA writes the data out to main memory. The core lowers DMA_req once all the RX data is read from the endpoint FIFO. The XFER DMA also writes the trigger bit in the core endpoint register to enable the RX endpoint buffer for the next transfer.

20.1.3.2.10.1 RNDIS Mode

The USB 2.0 OTG controller also supports accelerating RNDIS type packets over USB. RNDIS mode allows the hardware to recognize and automate large packets being sent over USB, such as how RNDIS utilizes USB. When using transparent mode (the default with no acceleration) each DMA is equivalent to a single USB packet, so each DMA must be for at most the USB max packet size (i.e., 64 bytes for FS Bulk or 512 bytes for HS Bulk) and is a start-of-packet (SOP) and an end-of-packet (EOP) (see the CPPI 4.1 Specification for details on SOP and EOP) upon reception. Similarly, transmit CPPI packets can be no more than the USB max packet size.

RNDIS transfers over USB form a simple protocol to send larger packet sizes than those supported by USB 2.0 specifications. This is accomplished by breaking the larger packet into smaller packets, which are not larger than the max USB packet size. The protocol defines the end of the larger packet by using a USB packet that is not the max USB packet size (called a short packet). In the case where the larger packet size is an exact multiple of the max USB packet length, then the terminating short packet must be a zero length packet after the last data packet. The receiver combines the data packets and completes the RNDIS packet when it combines a short packet at the end.

The XFER DMA supports accelerating RNDIS packets over USB by automating the packing, unpacking, generation, and detection of these larger packets. In RNDIS mode, the CPPI packet refers to the larger packet, and the XFER DMA sends the entire packet out as a sequence of USB packets. For transmission, the CPPI packet data is sent as multiple USB packets of up to the max USB packet size. If the last data matches the max USB packet size, then the XFER DMA generates a zero length packet immediately after the last data packet. For receiving, USB packet data is sent to the same CPPI packet until a short packet is received. Only upon receiving the short packet will the XFER DMA indicate to the CPPI packet that it is now the end-of-packet.

To use RNDIS acceleration, the USB max packet size of any RNDIS mode enabled endpoints must be a multiple of 64 bytes. RNDIS acceleration should not be enabled for endpoints where the max packet size is not a multiple of 64 bytes. Only transparent mode should be used for such endpoints.

20.1.3.2.10.2 Linux CDC Mode

Linux CDC mode is supported. CDC packets act in the same manner as RNDIS packets, except for the case where the last data matches the max USB packet size. In transmit operation, if an endpoint is configured or CDC Linux mode, the XFER DMA then generates a packet containing 1 byte of data, whose value is 0x00, to indicate the end of the transfer, similar to the manner in which the XDMA generates a zero length packet to terminate an RNDIS packet that exactly matches the max USB packet size. During receive operation, the XFER DMA recognizes the one byte zero packet as a termination of the USB packet, and sends a block of data with the EOP indicator set and a byte count of one to the CPPI DMA controller.

20.1.3.2.10.3 Generic RNDIS Mode

The USB 2.0 OTG controller supports a generic RNDIS mode over USB. This mode is identical to the normal RNDIS mode in nearly all respects, except for the exception case where data is being received or transmitted and the last packet of a segment exactly matches the max USB packet size. Normally, this mode receives USB packets in the same manner as RNDIS mode, closing the CPPI packet when a USB packet is received that is less than the endpoint size. Otherwise, the packet is closed when the value in the generic RNDIS EP size register is reached. Using this register, a packet of up to 64k bytes can be received. This is to allow the Host software to program the USB module to receive a packet that is an exact multiple of the endpoint size without having to send an additional short packet to terminate. In transmit operation, this mode behaves like RNDIS mode, except if the CPPI packet is an exact multiple of the endpoint size, a zero byte packet is not sent by the XDMA controller. The generic RNDIS EP size register must be programmed with a value that is an integer multiple of the endpoint size. For example, if the endpoint size is programmed with a value of 64, the generic RNDIS EP size register for that endpoint must be programmed with a value that is an integer multiple of 64, (e.g., 64, 128, 192, 256, etc.).

As in RNDIS mode, the USB max packet size of any generic RNDIS mode enabled endpoints must be a multiple of 64 bytes. Generic RNDIS acceleration should not be enabled for endpoints where the max packet size is not a multiple of 64 bytes. Only transparent mode should be used for such endpoints.

20.1.3.2.10.4 Zero Length Packets

A special case is the handling of zero length packets with the CPPI 4.1 compliant DMA controller. Upon receiving a zero length USB packet, the XFER DMA sends a data block to the DMA controller with the byte count of zero, the EOP bit set, and the zero byte packet bit of INFO Word 1 set. The DMA controller then performs normal EOP termination of the packet, without transferring data. To the packet transmission case, if a packet has the SOP and EOP packet has the zero byte packet bit of the INFO Word 1 of the CFIFO interface set, the XFER DMA ignores the CPPI packet size and sends a zero byte packet to the USB core. It is not valid to have the zero byte packet bit of the INFO Word 1 set without the SOP and EOP bits of INFO word 0 also being set.

20.1.3.2.10.5 Teardown

Teardown operation of an endpoint requires three operations. The teardown register in the CPPI DMA must be written, the corresponding endpoint bit in the teardown register of the USB module must be set, and the FlushFIFO bit in the Mentor USB controller CSR register must be set. Writing to the USB OTG controller teardown register resets the CPPI FIFO occupancy value and pointers to 0. It also resets the current state of the XDMA for the endpoint selected, after any current operations have been completed.. Note that due to VBUSP bridge latency, the CPPI FIFO occupancy values are not reset immediately upon writing of the teardown register.

20.1.3.2.11 PDR Interrupts

The interrupt interface provided by the core is not PDR compliant. The core only provides one interrupt signal. The CPU would have to read registers inside the core to decide what the source of the interrupt actually is. While there are enable (mask) registers and the ability to clear interrupts, there is no ability to set interrupts as required by PDR. This sub-module creates a PDR compliant interrupt interface for the OTG module. The interrupt block accepts the core interrupt, reads the core registers to determine the source(s), and provides the required interface. This includes a new interrupt source set register, and the end-of-interrupt (EOI) register. The original interrupt from the core will also be sent out for those systems that do not need the PDR interrupt interface. These interrupts are in addition to those provided by the DMA controller. The interrupts that are output by this sub-module are then aggregated by the Intd module in a PDR compliant manner.

There is also an extra interrupt mode, for systems that do not need external interrupt aggregation logic or PDR interrupts altogether. The system can utilize the aggregator logic inside the Mentor core, and not use any part of the PDR interrupt style for the USB interrupts. All USB interrupt interaction must be with the Mentor core registers in this mode, since the PDR interrupt logic is disabled. This is only for the USB interrupts, though, as the DMA interrupts only support PDR mode.

20.1.3.2.12 VBUSP Retiming

To enhance the timing of the VBUSP interfaces to the system, a VBUSP retiming bridge is used to isolate the internal timings from the external system. This retiming bridge simply takes in a VBUSP bus and outputs a VBUSP bus. These bridges are located on the incoming VBUSP configuration slave bus and the outgoing Master from the module SCR.

20.1.3.2.13 PDR Clocking and IP Generics

This sub-module implements a PDR 3.5 compliant clock interface. This consists of a gated clock, clock stop request, and clock stop acknowledge signals. When a clock stop request is asserted, this block waits until all DMAs inside the module have ceased, and sends the clock stop acknowledge. Then the system may stop the gated clock. Software should disable all the necessary functions inside the DMA and core so that there is no continued activity when a clock stop is requested.

The module also performs the required PDR 3.5 emulation interface. Setting the emulation register sets the method the module uses to perform emulation stops based on the VBUSP emulation suspend signals.

There is a register bit in the control register that allows for fast responding to the clock stop request. If this bit is enabled, this module returns a clock stop ack in the next clock cycle and does not wait for current DMAs to finish. The system needs to be tolerant of a possible loss of the DMA request if this occurs.

20.1.3.2.14 Switched Central Resource (SCR)

SCRs allow connecting more than two VBUSP ports together onto a single bus. By using VBUSP ports for connecting not only to the system, but also to local RAMs or FIFOs, a generic interface is provided that can be upgraded without affecting the other sub-modules. An SCR is used to connect all the internal and external interfaces together. These interfaces include: the core's AHB slave through the V2A bridge, the XFER DMA, and all the configuration ports.

The USB 2.0 OTG SCR has been enhanced to allow the SSRAM port to be non-bursting, while the remainder support bursting.

20.1.4 Interrupt Conditions

20.1.4.1 CPU Interrupts

The following interrupts are generated by the USB 2.0 OTG controller:

Table 20-4. CPU Interrupts

Interrupt	Description
TX_ENDP[15:1]	TX endpoint ready or error for endpoints 16 to 0 (endpoint 0 is for TX and RX)
RX_ENDP[15:1]	RX endpoint ready or error for endpoints 16 to 0 (there is no interrupt for endpoint 0)
USB[8:0]	Nine USB conditions
USB_INT	Single interrupt signal from the core

20.1.4.2 Interrupt Description

20.1.4.2.1 USB Core PDR Interrupts

There are 4 general-purpose TX endpoints supported in addition to control endpoint 0, whose interrupt encapsulates both TX and RX readiness. The core generates a corresponding TX_ENDP interrupt when that TX endpoint successfully completes transmitting a packet and the buffer is empty and ready to send another packet or when the TX endpoint had an error condition. If the TXCSR is setup as required for the DMA (with DMAReqEnab and DMAReqMode set), then the TX endpoint interrupt only generates error condition interrupts, and not packet completion interrupts, which would allow software to keep the interrupt enabled for monitoring error conditions. The TX endpoint interrupts are:

Table 20-5. TX Endpoint Interrupts

Interrupt	Description
TX_ENDP[15]	TX endpoint ready or error for endpoint 15
TX_ENDP[14]	TX endpoint ready or error for endpoint 14
TX_ENDP[13]	TX endpoint ready or error for endpoint 13
TX_ENDP[12]	TX endpoint ready or error for endpoint 12
TX_ENDP[11]	TX endpoint ready or error for endpoint 11
TX_ENDP[10]	TX endpoint ready or error for endpoint 10
TX_ENDP[9]	TX endpoint ready or error for endpoint 9
TX_ENDP[8]	TX endpoint ready or error for endpoint 8
TX_ENDP[7]	TX endpoint ready or error for endpoint 7
TX_ENDP[6]	TX endpoint ready or error for endpoint 6
TX_ENDP[5]	TX endpoint ready or error for endpoint 5
TX_ENDP[4]	TX endpoint ready or error for endpoint 4
TX_ENDP[3]	TX endpoint ready or error for endpoint 3
TX_ENDP[2]	TX endpoint ready or error for endpoint 2
TX_ENDP[1]	TX endpoint ready or error for endpoint 1
TX_ENDP[0]	TX or RX endpoint ready or error for endpoint 0

There are 4 general-purpose RX endpoints supported. The core generates a corresponding RX_ENDP interrupt when that RX endpoint successfully receives a packet and all the data is ready in the RX endpoint FIFO or when a receive error occurs. If the RXCSR is setup as required for the DMA (with DMAReqEnab set and DMAReqMode clear), then the RX endpoint interrupt only generates error condition interrupts, and not packet readiness interrupts, which would allow software to keep the interrupt enabled for monitoring error conditions. The RX endpoint interrupts are:

Table 20-6. RX Endpoint Interrupts

Interrupt	Description
RX_ENDP[15]	RX endpoint ready or error for endpoint 15
RX_ENDP[14]	RX endpoint ready or error for endpoint 14
RX_ENDP[13]	RX endpoint ready or error for endpoint 13
RX_ENDP[12]	RX endpoint ready or error for endpoint 12
RX_ENDP[11]	RX endpoint ready or error for endpoint 11
RX_ENDP[10]	RX endpoint ready or error for endpoint 10
RX_ENDP[9]	RX endpoint ready or error for endpoint 9
RX_ENDP[8]	RX endpoint ready or error for endpoint 8
RX_ENDP[7]	RX endpoint ready or error for endpoint 7
RX_ENDP[6]	RX endpoint ready or error for endpoint 6
RX_ENDP[5]	RX endpoint ready or error for endpoint 5
RX_ENDP[4]	RX endpoint ready or error for endpoint 4
RX_ENDP[3]	RX endpoint ready or error for endpoint 3
RX_ENDP[2]	RX endpoint ready or error for endpoint 2
RX_ENDP[1]	RX endpoint ready or error for endpoint 1

The USB OTG module also generates 9 USB interrupts based on noteworthy USB conditions. The USB interrupts are:

Table 20-7. USB Interrupts

Interrupt	Description
USB[8]	DRVVBUS level change
USB[7]	VBUS < VBUS valid threshold
USB[6]	SRP detected
USB[5]	Device disconnected (host mode)
USB[4]	Device connected (host mode)
USB[3]	SOF started
USB[2]	Reset signaling detected (peripheral mode) Babble detected (host mode)
USB[1]	Resume signaling detected
USB[0]	Suspend signaling detected

Interrupts USB[0] to USB[7] are generated by the Mentor core, and converted to PDR 2.0 format by the PDR interrupt sub-module. For more details on these interrupts, see the Mentor core documentation. Interrupt USB[8] is generated outside the Mentor core by the PDR interrupt sub-module whenever the level of DRVVBUS changes. This is usually due to entering or leaving Host mode, or entering or leaving power saving mode.

The USB20OTG interrupts are also aggregated by the Interrupt Distributor module. The table below provides a list of USB Interrupts that INTD processes. Only one USB interrupt signals is provided on the outside and INTD provides the interrupt aggregation logic for all USB interrupts.

When the EOI register in the USB module is written to, the eoi_wr and eoi_wr_vector information is sent to INTD which in turn re-evaluates the interrupt and triggers it if necessary.

20.1.4.2.2 USB Core Non-PDR Interrupts

If the non-PDR interrupt mode is selected for the USB interrupts, then the previous list of USB PDR sources are not valid, and the core generates the CPU interrupt directly. The pin from the core is connected to the module output, USB_INT, and all interactions to clear and mask interrupts must be done with the core registers. This interrupt is an active high signal.

The additional USB interrupt for DRVVBUS level changes is not part of the core interrupt set, and therefore is not available when using non-PDR interrupt mode.

20.1.4.2.3 Completion Queue Interrupts

The pending bits for queues 92-95 are aggregated in the INTD controller. These bits correspond to dedicated completion queues used by the CDMA for returning completed transmit and receive packets. The pending signals are level sensitive and when low, indicate that there are no packets on the queue. When a packet is added to one of the completion queues, the pending bit goes to high, indicating packets are available on the queue. When the low to high transition occurs, an interrupt is generated by the INTD module on the cdma_int pin. The Host software must read the queue status register for the completion queues to determine which queue caused the interrupt. Once all packets have been removed from the queue, the corresponding pending signal returns to 0. Software may choose to use one of the unassigned queues if an interrupt is not wanted upon the completion of a packet.

20.1.4.3 Interrupt Condition Control

20.1.4.3.1 USB Core Interrupts

The USB core interrupts are originally generated inside the Mentor core. The core provides enable registers for each of the three types of interrupts. The INTRTXE register allows enabling of each TX endpoint interrupt The INTRRXE register allows enabling of each RX endpoint interrupt. The INTRUSBE register allows enabling of each general USB interrupt. Any enabled interrupt generates CPU interrupts in the form of the corresponding interrupt source, the module generated interrupt pulse, and the interrupt pulse from the Mentor core directly.

To comply with PDR interrupts, additional logic is built around the core. This adds a mask register along with mask set and mask clear registers. The USB Interrupt Mask register shows the current mask value, where each bit enables the corresponding interrupt source in the USB Interrupt Source register. The USB Interrupt Mask Set register allows writing a ‘1’ in bit positions to enable the corresponding interrupt source. Those bits written with a ‘0’ will not be modified. The USB Interrupt Mask Clear register allows writing a ‘1’ in bit positions to disable the corresponding interrupt source. Those bit written with a ‘0’ will not be modified. The mask is used to enable interrupt sources and generate the masked interrupt sources which are used to generate the CPU interrupt or by external logic to generate CPU interrupts.

20.1.4.4 Interrupt Handling

20.1.4.4.1 USB Core Interrupts

The Mentor core generates the interrupts and the additional logic generates the PDR 3.5 interrupt sources. In order to generate the interrupt sources, the additional logic must read the interrupt source registers inside the core. When these registers are read, the core clears all the interrupt source bits, so the CPU will not have to do this. Therefore, the CPU simply needs to handle the data for TX or RX endpoint interrupts, or the USB interrupts and acknowledge the interrupt in the usual method. The interrupt source can be cleared by writing a ‘1’ in the corresponding bit position to the USB Interrupt Source Clear register. Also, the End of Interrupt register functions as required by PDR To facilitate debugging, writing a ‘1’ to bit positions in the USB Interrupt Source Set register sets the corresponding interrupt source even if it wasn’t generated by the core.

20.1.5 I/O Description

20.1.5.1 Module I/O

20.1.5.1.1 Reset Interface

Pin Name	Type	Function
vbusp_rst_n	In	Main reset for CDMA/Queue Manager/INTD
vbusp_usb0_rst_n	In	Main reset for USB controller 0 submodule
vbusp_usb1_rst_n	In	Main reset for USB controller 1 submodule

20.1.5.1.2 Queue Manager Event Interface

Pin Name	Type	Function
qmgr_event_queue[6:0]	Out	Queue on which a push/pop operation was performed. See CPPI 4.1.7 spec for hookup details.
qmgr_event_strobe	Out	Queue Event Valid Indicator. See CPPI 4.1.7 spec for hookup details.
qmgr_event_type	Out	Queue Push/Pop Type indicator with push operation indicated by a 1 and pop by a 0. See CPPI 4.1.7 spec for hookup details.

20.1.5.1.3 USB Interface

Pin Name	Type	Function
PHY_DP	Inout	USB I/O pin
PHY_DM	Inout	USB I/O pin
PHY_VBUS	Inout	USB I/O pin
PHY_ID	Inout	USB I/O pin
PHY_UTMI_DRVVBUS	Out	Drive VBUS connection to external charge pump
PHY_UTMI_DRVVBUS_OE_N	Out	Output enable for drive VBUS

20.1.5.1.4 Interrupt Interface

Pin Name	Type	Function
Usb_int	Out	USB Interrupt request
cdma_int	Out	CDMA Starvation Interrupt Out

20.1.5.2 External Pins

The only external pin is for an external charge pump. The controller outputs DRVVBUS to indicate whether the charge pump should drive the VBUS pin or not. This pin need not be present when using an internal charge pump. There are no timing requirements for DRVVBUS since it is basically a static signal that only changes at major USB actions, such as going into suspend mode, or becoming a Host.

20.1.5.2.1 External Pin Table

Table 20-8. External Pin Information

Name	Dir	Description	I/O Type [1]	Timing Parameter	Minimum Value	Maximum Value
PHY_UTMI_DRVVBUS	O	Drive VBUS	LV-TTL			

(1) I/O Types: 5 V-Tol = 5 Volt Tolerant, TTL-3S = TTL Output Level Tri-Stateable, LV-TTL Low Voltage TTL

20.1.6 Teardown Procedure

In order to fully teardown a TX or RX channel, the following procedure must be followed. Software should guarantee that all steps have been completed before assuming teardown has completed successfully.

20.1.6.1 Transmit Teardown

First, the CPPI DMA teardown procedure must be implemented. This is to ensure that the DMA controller is aware of the teardown and can respond properly. See the CDMA and CPPI 4.1 specifications for more details on DMA transmit teardown.

Next, once the CDMA teardown is complete and any current packets the CDMA is transmitting on the channel have completed, the USB controller Teardown register must be written, setting the corresponding bit of the endpoint that is to be torn down set. This makes sure the CPPI FIFO interface is reset properly and no spurious data remains in the FIFOs.

Lastly, the FlushFIFO bit of the CSR Register of the endpoint that is to be torn down must be set. Make certain that the necessary bits are kept set if the entire register is written. See the Mentor core documentation for more information.

After these steps are complete, the DMA channel and endpoint must go through the normal reconfiguration procedure and then can resume normal operation.

20.1.6.2 Receive Teardown

Set the FlushFIFO bit of the CSR register of the endpoint that is to be torn down. See the Mentor core documentation for more detailed information.

Next, set the USB controller Teardown register for the designated endpoint. Lastly, initiate the CDMA receive teardown procedure as described in the CDMA specification. To ensure that no data corrution has occurred in the Mentor core FIFO, the FlushFIFO bit of the CSR register should be written a second time after the CDMA channel teardown. To ensure good operation, the Host should wait at least 100 VBUSP_CLK clock cycles after CDMA channel teardown and the second write of the FlushFIFO bit. At this time the endpoint has completed teardown and can be reinitialized according to the standard endpoint setup procedure and normal operation resumed.

20.1.7 USB Bus Reset Handling

After a USB bus reset, the Mentor core interrupts as well as clears many of its registers. This requires the software to teardown any TX DMA channels currently in use before setting up the DMA and core again. Failure to perform the teardown procedure as described previously may result in erratic behavior or unwanted TX data.

20.1.8 VBUSP Slave Port Bursting

The USB 2.0 OTG subsystem does not support bursting on the VBUSP slave port. All accesses should be single word (or smaller) accesses, and the first and last signals should always be high. Sending bursts to registers can result in unpredictable behavior.

20.1.9 Core Register Type Mixing

Software must take care when reading core registers larger than their original size to optimize reads. If the registers read are from different types (common, endpoint, or FIFO) then the core only reads the registers in the type of the initial address, and not the upper address registers in another type. An example is the DEVCTL common register at 0x460 with the TXFIFOSZ and RXFIFOSZ endpoint registers at 0x462 and 0x463. The core only reads the DEVCTL register in this case. To read the TXFIFOSZ and RXFIFOSZ registers, software must read starting at 0x462.

20.1.10 Zero Byte Packet Parameters

Zero-byte packets must be generated with the zero-byte bit of Info register 1 set (see the CPPI FIFO Info register 1) and the packet size should be 1 byte. The actual packet data is not used.

20.1.11 Interrupt Usage

The expected usage of the interrupts when using the DMA is as follows. The DMA generates interrupts after each CPPI packet is processed. With the expects TXCSR and RXCSR setup as required for the DMA, the TX endpiont and RX endpoint interrupts will only be generated when there is an error or stall condition from the core. Therefore, the CPU should not disable and ignore the endpoint interrupts unless error detection is not desired or polling is used.

20.1.12 Powerdown Handling

In order to safely powerdown the USB 2.0 OTG module, the SUSPENDM signal should be monitored. When low, the USB bus is going into suspend mode and the system can safely powerdown the OTG module. This includes disabling the clock, and disabling the PHY. Please refer to the PHY spec on how to safely powerdown the PHY.

In order to wakeup after suspend from a bus resume, the PHY must be powered on to drive the LINESTATE, VBUSVALID, and AVALID UTMI signals. The OTG controller uses these to asynchronously generate SUSPENDM, so it can drive SUSPENDM high even without a clock. When SUSPENDM goes high, the USB 2.0 OTG module and the PHY should be powered up again so that the USB operations can resume.

20.1.13 Clock Stop and Emulation Suspend

The USB module supports clock stop operation and CBA compliant emulation suspend. It should be noted that for clock stop operation, the UTMI clock is unaffected and the Mentor core continues to interact with the phy device unless the phy is powered down at the time. Similarly for emulation suspend, the XDMA state machine halts once the current packet processing is complete, but the Mentor controller is unaffected by the suspend unless the phy is also powered down at the same time.

The clock stop interface allows the DMA to be gracefully commanded to enter into a stopped clock state. When the *clkstop_req* input is asserted, the DMA suspend state is entered and, the DMA stops processing receive and transmit packets for each channel at the next packet boundary. Any packet currently in reception or transmission is completed normally without suspension. The *clkstop_ack* output is asserted when the DMA enters the IDLE state and all of the individual channels have indicated that they are stopped and are no longer in the middle of transmitting or receiving a packet. The *clkstop_ack* indicates that the clock, *vbusp_gclk*, may be stopped.

20.1.14 Registers

Table 20-9 lists the memory-mapped registers for the high speed universal serial bus (USB).

Table 20-9. Universal Serial Bus (USB) Registers

VBUS Slave Address Offset	Register	Section
0x000	Reserved	
0x004	USB Control Register	Section 20.1.14.1
0x008	USB Status Register	Section 20.1.14.2
0x00C	Reserved	
0x010	Reserved	
0x014	USB Auto Req Register	Section 20.1.14.3
0x018	Reserved	
0x01C	USB Teardown Register	Section 20.1.14.4
0x020	USB Endpoint Interrupt Source Register	Section 20.1.14.5
0x024	USB Endpoint Interrupt Source Set Register	Section 20.1.14.6
0x028	USB Endpoint Interrupt Source Clear Register	Section 20.1.14.7
0x02C	USB Endpoint Interrupt Mask Register	Section 20.1.14.8
0x030	USB Endpoint Interrupt Mask Set Register	Section 20.1.14.9
0x034	USB Endpoint Interrupt Mask Clear Register	Section 20.1.14.10
0x038	USB Endpoint Interrupt Source Masked Register	Section 20.1.14.11
0x03C	Reserved	
0x040	USB Core Interrupt Source Register	Section 20.1.14.12
0x044	USB Core Interrupt Source Set Register	Section 20.1.14.13
0x048	USB Core Interrupt Source Clear Register	Section 20.1.14.14
0x04C	USB Core Interrupt Mask Register	Section 20.1.14.15
0x050	USB Core Interrupt Mask Set Register	Section 20.1.14.16
0x054	USB Core Interrupt Mask Clear Register	Section 20.1.14.17
0x058	USB Core Interrupt Source Masked Register	Section 20.1.14.18
0x05C	Reserved	
0x060	USB End of Interrupt Register	Section 20.1.14.19
0x064	MOP/SOP/ Interrupt Enable Register	Section 20.1.14.20
0x068 – 0x06C	Reserved	
0x70	USB Tx Mode Register	Section 20.1.14.21
0x74	USB Rx Mode Register	Section 20.1.14.22
0x78	EP Count Mode Register	Section 20.1.14.23
0x07C	Reserved	
0x080	USB Generic RNDIS Size EP1 Register	Section 20.1.14.24
0x084	USB Generic RNDIS Size EP2 Register	Section 20.1.14.24
0x088	USB Generic RNDIS Size EP3 Register	Section 20.1.14.24
0x08C	USB Generic RNDIS Size EP4 Register	Section 20.1.14.24
0x090	USB Generic RNDIS Size EP5 Register	Section 20.1.14.24
0x094	USB Generic RNDIS Size EP6 Register	Section 20.1.14.24
0x098	USB Generic RNDIS Size EP7 Register	Section 20.1.14.24
0x09C	USB Generic RNDIS Size EP8 Register	Section 20.1.14.24
0x0A0	USB Generic RNDIS Size EP9 Register	Section 20.1.14.24
0x0A4	USB Generic RNDIS Size EP10 Register	Section 20.1.14.24
0x0A8	USB Generic RNDIS Size EP11 Register	Section 20.1.14.24
0x0AC	USB Generic RNDIS Size EP12 Register	Section 20.1.14.24
0x0B0	USB Generic RNDIS Size EP13 Register	Section 20.1.14.24

Table 20-9. Universal Serial Bus (USB) Registers (continued)

VBUS Slave Address Offset	Register	Section
0x0B4	USB Generic RNDIS Size EP14 Register	Section 20.1.14.24
0x0B8	USB Generic RNDIS Size EP15 Register	Section 20.1.14.24
0x0C0	USB Queue Interrupt Threshold Enable Register	Section 20.1.14.25
0x0C4	USB Queue Threshold Register 0	Section 20.1.14.26
0x0C8	USB Queue Interrupt Clear Register 0	Section 20.1.14.27
0x0D4	USB Queue Threshold Register 1	Section 20.1.14.28
0x0D8	USB Queue Interrupt Clear Register 1	Section 20.1.14.29
0x0DC-0x3FF	Reserved	
0x400 – 0x59C	USB Mentor Core Registers/FIFOs	Section 20.1.14.30
0x1800	CDMA Tx Channel 0 Global Configuration Register	Section 20.1.14.30.1
0x1804	Reserved	
0x1808	CDMA Rx Channel 0 Global Configuration Register	Section 20.1.14.30.2
0x180C	CDMA Rx Channel 0 Host Packet Configuration Register A	Section 20.1.14.30.3
0x1810	CDMA Rx Channel 0 Host Packet Configuration Register B	Section 20.1.14.30.4
0x1814-0x181C	Reserved	
0x1820	Tx Channel 1 Global Configuration Register	Section 20.1.14.30.1
0x1824	Reserved	
0x1828	Rx Channel 1 Global Configuration Register	Section 20.1.14.30.2
0x182C	Rx Channel 1 Host Packet Configuration Register A	Section 20.1.14.30.3
0x1830	Rx Channel 1 Host Packet Configuration Register B	Section 20.1.14.30.4
0x1834 - 0x183C	Reserved	
0x1840	Tx Channel 2 Global Configuration Register	Section 20.1.14.30.1
0x1844	Reserved	
0x1848	Rx Channel 2 Global Configuration Register	Section 20.1.14.30.2
0x184C	Rx Channel 2 Host Packet Configuration Register A	Section 20.1.14.30.3
0x1850	Rx Channel 2 Host Packet Configuration Register B	Section 20.1.14.30.4
0x1854-0x185F	Reserved	
0x1860	Tx Channel 3 Global Configuration Register	Section 20.1.14.30.1
0x1864	Reserved	
0x1868	Rx Channel 3 Global Configuration Register	Section 20.1.14.30.2
0x186C	Rx Channel 3 Host Packet Configuration Register A	Section 20.1.14.30.3
0x1860	Rx Channel 3 Host Packet Configuration Register B	Section 20.1.14.30.4
0x1880 - 0x1CDF	...	
0x1CA0	Tx Channel 29 Global Configuration Register	Section 20.1.14.30.1
0x1CA4	Reserved	
0x1CA8	Rx Channel 29 Global Configuration Register	Section 20.1.14.30.2
0x1CAC	Rx Channel 29 Host Packet Configuration Register A	Section 20.1.14.30.3
0x1CA0	Rx Channel 29 Host Packet Configuration Register B	Section 20.1.14.30.4
0x1CE4-0x1FFF	Reserved	
0x2000	CDMA Scheduler Control Register	Section 20.1.14.30.5
0x2804-0x28FF	Reserved	
0x2800	CDMA Scheduler Table Word 0 Register	Section 20.1.14.30.6
0x2804	CDMA Scheduler Table Word 1 Register	Section 20.1.14.30.6
…	…	
0x28F8	CDMA Scheduler Table Word 62 Register	Section 20.1.14.30.6
0x28FC	CDMA Scheduler Table Word 63 Register	Section 20.1.14.30.6
0x28FF-0x2FFF	Reserved	

Table 20-9. Universal Serial Bus (USB) Registers (continued)

VBUS Slave Address Offset	Register	Section
0x3000	INTD Revision Register	Section 20.1.14.31
0x3004 - 0x3008	Reserved	
0x3010	INTD EOI Register (legacy)	Section 20.1.14.32
0x3014	INTD EOI Interrupt Vector Register	Section 20.1.14.33
0x3200 – 0x32FC	INTD Status Registers	Section 20.1.14.34
0x3300-0x3FFF	Reserved	
0x4000	Queue Manager Revision Register	Section 20.1.14.39
0x4008	Queue Manager Queue Diversion Register	Section 20.1.14.40
0x4020	Queue Manager Free Descriptor/Buffer Starvation Count Register 0	Section 20.1.14.41
0x4024	Queue Manager Free Descriptor/Buffer Starvation Count Register 1	Section 20.1.14.42
0x4028	Queue Manager Free Descriptor/Buffer Starvation Count Register 2	Section 20.1.14.43
0x402c	Queue Manager Free Descriptor/Buffer Starvation Count Register 3	Section 20.1.14.44
0x4030	Queue Manager Free Descriptor/Buffer Starvation Count Register 4	Section 20.1.14.45
0x4034	Queue Manager Free Descriptor/Buffer Starvation Count Register 5	Section 20.1.14.46
0x4038	Queue Manager Free Descriptor/Buffer Starvation Count Register 6	Section 20.1.14.47
0x403c	Queue Manager Free Descriptor/Buffer Starvation Count Register 7	Section 20.1.14.48
0x4030-0x407c	Reserved	
0x4080	Queue Manager Linking RAM Region 0 Base Address Register	Section 20.1.14.49
0x4084	Queue Manager Linking RAM Region 0 Size Register	Section 20.1.14.50
0x4088	Queue Manager Linking RAM Region 1 Base Address Register	Section 20.1.14.51
0x4090	Queue Manager Queue Pending Register 0	Section 20.1.14.52
0x4094	Queue Manager Queue Pending Register 1	Section 20.1.14.53
0x4098	Queue Manager Queue Pending Register 2	Section 20.1.14.54
0x409c-0x4FFFF	Reserved	
0x5000 + 16xR	Queue Manager Memory Region R Base Address Register	Section 20.1.14.55
0x5000 + 16xR + 4	Queue Manager Memory Region R Control Register	Section 20.1.14.56
0x6028-0x7FFF	Reserved	
0x6000 + 16xN	Queue Manager Queue N Register A	Section 20.1.14.57
0x6000 + 16xN + 4	Queue Manager Queue N Register B	Section 20.1.14.58
0x6000 + 16xN + 8	Queue Manager Queue N Register C	Section 20.1.14.59
0x6000 + 16xN + C	Queue Manager Queue N Register D	Section 20.1.14.60
0x4400-0x47FF	Reserved	
0x6800 + 16xN	5.61Queue Manager Queue N Status Register A	Section 20.1.14.61
0x6800 + 16xN + 4	Queue Manager Queue N Status Register B	Section 20.1.14.62
0x6800 + 16xN + 8	Queue Manager Queue N Status Register C	Section 20.1.14.63

20.1.14.1 USB Control Register (Base Address + 0x0004)

The control register allows the CPU to control various aspects of the module.

The USB control register is shown in Figure 20-4 and described in Table 20-10.

Figure 20-4. USB Control Register

31	30	29														16
dis_deb	dis_srp	Reserved														
R/W	R/W	R														
15												5	4	3	2	1	0
Reserved													rndis	uint	RSVD	clkfack	Reset
R													R/W	R/W	R	R/W	R/W-1

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

Table 20-10. USB Control Register Field Descriptions

Bit	Field	Value	Description
31	dis_deb		Disable the VBUS debouncer circuit fix
30	dis_srp		Disable the SRP a_valid circuit fix
29-5	Reserved	0	Always read as 0. Writes have no affect.
4	rndis		Global RNDIS mode enable for all endpoints.
3	uint		USB non-PDR interrupt enable
2	Reserved	0	Always read as 0. Writes have no effect.
1	clkfack		Clock stop fast ack enable.
0	Reset		Soft reset. Writing a 1 starts a module reset.

20.1.14.2 USB Status Register (Base Address + 0x0008)

The status register allows the CPU to check various aspects of the module.

The USB status register is shown in Figure 20-5 and described in Table 20-11.

Figure 20-5. USB Status Register

31	1	0
Reserved		drvvbus
		R

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

Table 20-11. USB Status Register Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Always read as 0. Writes have no affect.
0	drvvbus		Current DRVVBUS value

20.1.14.3 USB Auto Req Register (Base Address + 0x0014)

The auto req register allows the CPU to enable an automatic IN token request generation for the Host mode RX operation per each RX endpoint. This features has the DMA set the ReqPkt bit in the RXCSR when it clears the RxPktRdy bit after reading out a packet. The ReqPkt bit is used by the core to generate an IN token to receive data. By using this feature, the Host can automatically generate an IN token after the DMA finishes receiving data and empties an endpoint buffer, thus receiving the next data packet as soon as possible from the connected device. Without this feature, the CPU has to manually set the ReqPkt bit for every USB packet. There are two modes that Auto Req can function: always or all except an end of packet (EOP). The always mode sets the ReqPkt bit after every USB packet the DMA receives generating a new IN token after each USB packet. The EOP mode sets the ReqPkt bit after every USB packet that isn't an EOP in the CPPI descriptor. For RNDIS, CDC, and generic RNDIS modes, the auto req stops when the EOP is received (either via a short packet for RNDIS,CDC, and generic RNDIS or the count is reached for generic RNDIS), making it useful for starting a large RNDIS packet and having it auto generate IN tokens until the end of the RNDIS packet. For transparent mode, every USB packet is an EOP CPPI packet so the auto req never functions and acts like it is disabled.

The USB auto req register is shown in Figure 20-6 and described in Table 20-12.

Figure 20-6. USB Auto Req Register

31	30	29	28	27	26	25	24
Reserved		Rx(N+14)_autoreq		Rx(N+13)_autoreq		Rx(N+12)_autoreq	
R		R/W		R/W		R/W	
23	22	21	20	19	18	17	16
Rx(N+11)_autoreq		Rx(N+10)_autoreq		Rx(N+9)_autoreq		Rx(N+8)_autoreq	
R/W		R/W		R/W		R/W	
15	14	13	12	11	10	9	8
Rx(N+7)_autoreq		Rx(N+6)_autoreq		Rx(N+5)_autoreq		Rx(N+4)_autoreq	
R/W		R/W		R/W		R/W	
7	6	5	4	3	2	1	0
Rx(N+3)_autoreq		Rx(N+2)_autoreq		Rx(N+1)_autoreq		Rx(N)_autoreq	
R/W		R/W		R/W		R/W	

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

Table 20-12. USB Auto Req Register Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no affect.
29-28	Rx(N+14)_autoreq		RX endpoint N+14 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
27-26	Rx(N+13)_autoreq		RX endpoint N+13 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
25-24	Rx(N+12)_autoreq		RX endpoint N+12 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always

Table 20-12. USB Auto Req Register Field Descriptions (continued)

Bit	Field	Value	Description
23-22	Rx(N+11)_autoreq		RX endpoint N+11 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
21-20	Rx(N+10)_autoreq		RX endpoint N+10 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
19-18	Rx(N+9)_autoreq		RX endpoint N+9 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
17-16	Rx(N+8)_autoreq		RX endpoint N+8 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
15-14	Rx(N+7)_autoreq		RX endpoint N+7 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
13-12	Rx(N+6)_autoreq		RX endpoint N+6 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
11-10	Rx(N+5)_autoreq		RX endpoint N+5 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
9-8	Rx(N+4)_autoreq		RX endpoint N+4 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
7-6	Rx(N+3)_autoreq		RX endpoint N+3 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always

Table 20-12. USB Auto Req Register Field Descriptions (continued)

Bit	Field	Value	Description
5-4	Rx(N+2)_autoreq		RX endpoint N+2 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
3-2	Rx(N+1)_autoreq		RX endpoint N+1 Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always
1-0	Rx(N)_autoreq		RX endpoint N Auto Req enable
		00	No auto req
		01	Auto req on all but EOP
		10	Reserved
		11	Auto req always

20.1.14.4 USB Teardown Register (Base Address + 0x001C)

The teardown register controls the tearing down of rx and tx FIFOs in the USB controller. When a '1' is written to a valid bit in this register, the CPPI FIFO pointers for that endpoint are cleared. This register must be used in conjunction with the CPPI DMA teardown mechanism. The Host should also write the FlushFIFO bits in the TXCSR and RXCSR Mentor USB controller registers to ensure a complete teardown of the endpoint. See the Mentor specification for details.

The USB teardown register is shown in Figure 20-7 and described in Table 20-13.

Figure 20-7. USB Teardown Register

31	17	16	15	1	0
tx_tdown		Reserved	rx_tdown		Reserved
R/W			R/W		

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

Table 20-13. USB Teardown Register Field Descriptions

Bit	Field	Value	Description
31-17	tx_tdown		Tx Endpoint Teardown. Write '1' to corresponding bit to set. Read as '0'. Bit 31 = Endpoint 15 ... Bit 17 = Endpoint 1
16	Reserved	0	Always read as 0. Writes have no affect.
15-1	rx_tdown		RX Endpoint Teardown. Write '1' to corresponding bit to set. Read as '0'. Bit 15 = Endpoint 15 ... Bit 1 = Endpoint 1
0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.5 USB Endpoint Interrupt Source Register (Base Address + 0x0020)

The USB interrupt source register contains the status of the interrupt sources generated by the USB core (not the DMA).

The USB endpoint interrupt source register is shown in Figure 20-8 and described in Table 20-14.

Figure 20-8. USB Endpoint Interrupt Source Register

31	17	16	5	0
rx		Reserved	tx	
R			R	

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

Table 20-14. USB Endpoint Interrupt Source Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		RX endpoint interrupt sources (N+15 to N).
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		TX endpoint interrupt sources (N+15 to N+1 plus endpoint 0).

20.1.14.6 USB Endpoint Interrupt Source Set Register (Base Address + 0x0024)

The USB interrupt source set register allows the USB interrupt sources to be manually triggered. A read of this register returns the USB interrupt source register value.

The USB endpoint interrupt source set register is shown in Figure 20-9 and described in Table 20-15.

Figure 20-9. USB Endpoint Interrupt Source Set Register

31	17	16	5	0
rx		Reserved	tx	
R/W			R/W	

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

Table 20-15. USB Endpoint Interrupt Source Set Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		Write a 1 to set equivalent RX endpoint interrupt source.
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		Write a 1 to set equivalent TX endpoint interrupt source.

20.1.14.7 USB Endpoint Interrupt Source Clear Register (Base Address + 0x0028)

The USB interrupt source clear register allows the CPU to acknowledge an interrupt source and turn it off. A read of this register returns the USB interrupt source register value.

The USB endpoint interrupt source clear register is shown in Figure 20-10 and described in Table 20-16.

Figure 20-10. USB Endpoint Interrupt Source Clear Register

31	17	16	5	0
rx		Reserved	tx	
R/W			R/W	

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

Table 20-16. USB Endpoint Interrupt Source Clear Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		Write a 1 to clear equivalent RX endpoint interrupt source.
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		Write a 1 to clar equivalent TX endpoint interrupt source.

20.1.14.8 USB Endpoint Interrupt Mask Register (Base Address + 0x002C)

The USB interrupt mask register contains the masks of the interrupt sources generated by the USB core (not the DMA). These masks are used to enable or disable interrupt sources generated on the masked source interrupts (the raw source interrupts are never masked). The bit positions are maintained in the same position as the interrupt sources in the USB interrupt source register.

The USB endpoint interrupt mask register is shown in Figure 20-11 and described in Table 20-17.

Figure 20-11. USB Endpoint Interrupt Mask Register

31	17	16	5	0
rx		Reserved	tx	
R			R	

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

Table 20-17. USB Endpoint Interrupt Mask Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		Write a 1 to clear equivalent RX endpoint interrupt source.
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		Write a 1 to clar equivalent TX endpoint interrupt source.

20.1.14.9 USB Endpoint Interrupt Mask Set Register (Base Address + 0x0030)

The USB interrupt mask set register allows the USB interrupt masks to be individually enabled. A read to this register returns the USB interrupt mask register value.

The USB endpoint interrupt mask set register is shown in Figure 20-12 and described in Table 20-18.

Figure 20-12. USB Endpoint Interrupt Mask Set Register

31	17	16	5	0
rx		Reserved	tx	
R/W			R/W	

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

Table 20-18. USB Endpoint Interrupt Mask Set Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		Write a 1 to set equivalent RX endpoint interrupt mask.
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		Write a 1 to set equivalent TX endpoint interrupt mask.

20.1.14.10 USB Endpoint Interrupt Mask Clear Register (Base Address + 0x0034)

The USB interrupt mask clear register allows the USB interrupt masks to be individually disabled. A read to this register returns the USB interrupt mask register value.

The USB endpoint interrupt mask clear register is shown in Figure 20-13 and described in Table 20-19.

Figure 20-13. USB Endpoint Interrupt Mask Clear Register

31	17	16	5	0
rx		Reserved	tx	
R/W			R/W	

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

Table 20-19. USB Endpoint Interrupt Mask Clear Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		Write a 1 to clear equivalent RX endpoint interrupt mask.
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		Write a 1 to clear equivalent TX endpoint interrupt mask.

20.1.14.11 USB Endpoint Interrupt Source Masked Register (Base Address + 0x0038)

The USB endpoint interrupt source masked register contains the status of the interrupt sources generated by the USB core masked by the USB interrupt mask register values.

The USB endpoint interrupt source masked register is shown in Figure 20-14 and described in Table 20-20.

Figure 20-14. USB Endpoint Interrupt Source Masked Register

31	17	16	5	0
rx		Reserved	tx	
R/W			R/W	

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

Table 20-20. USB Endpoint Interrupt Source Masked Register Field Descriptions

Bit	Field	Value	Description
31-17	rx		RX endpoint interrupt sources masked.
16	Reserved	0	Always read as 0. Writes have no affect.
15-0	tx		TX endpoint interrupt sources masked.

20.1.14.12 USB Core Interrupt Source Register (Base Address + 0x0040)

The USB interrupt source register contains the status of the interrupt sources generated by the USB core (not the DMA).

The USB core interrupt source register is shown in Figure 20-15 and described in Table 20-21.

Figure 20-15. USB Core Interrupt Source Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R			

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

Table 20-21. USB Core Interrupt Source Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		USB interrupt sources.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.13 USB Core Interrupt Source Set Register (Base Address + 0x0044)

The USB interrupt source set register allows the USB interrupt sources to be manually triggered. A read of this register returns the USB interrupt source register value.

The USB core interrupt source set register is shown in Figure 20-16 and described in Table 20-22.

Figure 20-16. USB Core Interrupt Source Set Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R/W			

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

Table 20-22. USB Core Interrupt Source Set Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		Write a 1 to set equivalent USB interrupt source.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.14 USB Core Interrupt Source Clear Register (Base Address + 0x0048)

The USB interrupt source clear register allows the CPU to acknowledge an interrupt source and turn it off. A read of this register returns the USB interrupt source register value.

The USB core interrupt source clear register is shown in Figure 20-17 and described in Table 20-23.

Figure 20-17. USB Core Interrupt Source Clear Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R/W			

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

Table 20-23. USB Core Interrupt Source Clear Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		Write a 1 to clear equivalent USB interrupt source.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.15 USB Core Interrupt Mask Register (Base Address + 0x004C)

The USB interrupt mask register contains the masks of the interrupt sources generated by the USB core (not the DMA). These masks are used to enable or disable interrupt sources generated on the masked source interrupts (the raw source interrupts are never masked). The bit positions are maintained in the same position as the interrupt sources in the USB interrupt source register.

The USB core interrupt mask register is shown in Figure 20-18 and described in Table 20-24.

Figure 20-18. USB Core Interrupt Mask Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R			

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

Table 20-24. USB Core Interrupt Mask Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		USB interrupt source masks.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.16 USB Core Interrupt Mask Set Register (Base Address + 0x0050)

The USB interrupt mask set register allows the USB interrupt masks to be individually enabled. A read to this register returns the USB interrupt mask register value.

The USB core interrupt mask set register is shown in Figure 20-19 and described in Table 20-25.

Figure 20-19. USB Core Interrupt Mask Set Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R/W			

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

Table 20-25. USB Core Interrupt Mask Set Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		Write a 1 to set equivalent USB interrupt mask.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.17 USB Core Interrupt Mask Clear Register (Base Address + 0x0054)

The USB interrupt mask clear register allows the USB interrupt masks to be individually disabled. A read to this register returns the USB interrupt mask register value.

The USB core interrupt mask clear register is shown in Figure 20-20 and described in Table 20-26.

Figure 20-20. USB Core Interrupt Mask Clear Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R/W			

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

Table 20-26. USB Core Interrupt Mask Clear Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		Write a 1 to clear equivalent USB interrupt mask.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.18 USB Core Interrupt Source Masked Register (Base Address + 0x0058)

The USB interrupt source masked register contains the status of the interrupt sources generated by the USB core masked by the USB interrupt mask register values.

The USB core interrupt source masked register is shown in Figure 20-21 and described in Table 20-27.

Figure 20-21. USB Core Interrupt Source Masked Register

31	25	24	16	15	0
Reserved		usb		Reserved	
		R			

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

Table 20-27. USB Core Interrupt Source Masked Register Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Always read as 0. Writes have no affect.
24-16	usb		USB interrupt sources masked.
15-0	Reserved	0	Always read as 0. Writes have no affect.

20.1.14.19 USB End of Interrupt Register (Base Address + 0x0060)

The USB end of interrupt register allows the CPU to acknowledge completion of an interrupt by writing to the EOI. An eoi_write signal is generated, the EOI vector is updated to the written value, and another interrupt is triggered if interrupt sources remain.

The USB end of interrupt register is shown in Figure 20-22 and described in Table 20-28.

Figure 20-22. USB End of Interrupt Register

31	8	7	0
Reserved		eoi_vector	
		R/W	

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

Table 20-28. USB End of Interrupt Register Field Descriptions

Bit	Field	Value	Description
318	Reserved	0	Always read as 0. Writes have no affect.
7-0	eoi_vector		EOI vector

20.1.14.20 USB MOP/SOP Interrupt Enable Register (Base Address + 0x0064)

This register must be set to enable the DMA buffer starvation interrupt. When bit for the mop or sop interrupt is enabled, the interrupt to the INTD module triggers the first time the DMA issues that interrupt and then automatically disables itself, blocking future interrupts until the Host re-enables via this register.

The USB MOP/SOP interrupt enable register is shown in Figure 20-23 and described in Table 20-29.

Figure 20-23. USB MOP/SOP Interrupt Enable Register

31	2	1	0
Reserved		sop_en	mop_en
		R/W	R/W

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

Table 20-29. USB MOP/SOP Interrupt Enable Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no affect.
1	sop_en		When 1, DMA sop starvation interrupt is enabled.
0	mop_en		When 1, DMA mop starvation interrupt is enabled.

20.1.14.21 USB Tx Mode Register (Base Address + 0x0070)

The mode register allows the CPU to individually enable RNDIS/generic/CDC modes for each endpoint. Using the global RNDIS enable in the control register overrides this register and enables RNDIS mode for all endpoints.

The USB Tx mode register is shown in Figure 20-24 and described in Table 20-30.

Figure 20-24. USB Tx Mode Register

31 30	29 28	27 26	25 24
Reserved	Tx(N+14)_mode	Tx(N+13)_mode	Tx(N+12)_mode
R	R/W	R/W	R/W
23 22	**21 20**	**19 18**	**17 16**
Tx(N+11)_mode	Tx(N+10)_mode	Tx(N+9)_mode	Tx(N+8)_mode
R/W	R/W	R/W	R/W
15 14	**13 12**	**11 10**	**9 8**
Tx(N+7)_mode	Tx(N+6)_mode	Tx(N+5)_mode	Tx(N+4)_mode
R/W	R/W	R/W	R/W
7 6	**5 4**	**3 2**	**1 0**
Tx(N+3)_mode	Tx(N+2)_mode	Tx(N+1)_mode	Tx(N)_mode
R/W	R/W	R/W	R/W

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

Table 20-30. USB Tx Mode Register Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no affect.
29-28	Tx(N+14)_mode	00	Transparent Mode on TX endpoint N+14
		01	RNDIS MODE on TX endpoint N+14
		10	CDC Mode on TX endpoint N+14
		11	Generic RNDIS Mode on TX endpoint N+14
27-26	Tx(N+13)_mode	00	Transparent Mode on TX endpoint N+13
		01	RNDIS MODE on TX endpoint N+13
		10	CDC Mode on TX endpoint N+13
		11	Generic RNDIS Mode on TX endpoint N+13
25-24	Tx(N+12)_mode	00	Transparent Mode on TX endpoint N+12
		01	RNDIS MODE on TX endpoint N+12
		10	CDC Mode on TX endpoint N+12
		11	Generic RNDIS Mode on TX endpoint N+12
23-22	Tx(N+11)_mode	00	Transparent Mode on TX endpoint N+11
		01	RNDIS MODE on TX endpoint N+11
		10	CDC Mode on TX endpoint N+11
		11	Generic RNDIS Mode on TX endpoint N+11
21-20	Tx(N+10)_mode	00	Transparent Mode on TX endpoint N+10
		01	RNDIS MODE on TX endpoint N+10
		10	CDC Mode on TX endpoint N+10
		11	Generic RNDIS Mode on TX endpoint N+10
19-18	Tx(N+9)_mode	00	Transparent Mode on TX endpoint N+9
		01	RNDIS MODE on TX endpoint N+9
		10	CDC Mode on TX endpoint N+9
		11	Generic RNDIS Mode on TX endpoint N+9

Table 20-30. USB Tx Mode Register Field Descriptions (continued)

Bit	Field	Value	Description
17-16	Tx(N+8)_mode	00	Transparent Mode on TX endpoint N+8
		01	RNDIS MODE on TX endpoint N+8
		10	CDC Mode on TX endpoint N+8
		11	Generic RNDIS Mode on TX endpoint N+8
15-14	Tx(N+7)_mode	00	Transparent Mode on TX endpoint N+7
		01	RNDIS MODE on TX endpoint N+7
		10	CDC Mode on TX endpoint N+7
		11	Generic RNDIS Mode on TX endpoint N+7
13-12	Tx(N+6)_mode	00	Transparent Mode on TX endpoint N+6
		01	RNDIS MODE on TX endpoint N+6
		10	CDC Mode on TX endpoint N+6
		11	Generic RNDIS Mode on TX endpoint N+6
11-10	Tx(N+5)_mode	00	Transparent Mode on TX endpoint N+5
		01	RNDIS MODE on TX endpoint N+5
		10	CDC Mode on TX endpoint N+5
		11	Generic RNDIS Mode on TX endpoint N+5
9-8	Tx(N+4)_mode	00	Transparent Mode on TX endpoint N+4
		01	RNDIS MODE on TX endpoint N+4
		10	CDC Mode on TX endpoint N+4
		11	Generic RNDIS Mode on TX endpoint N+4
7-6	Tx(N+3)_mode	00	Transparent Mode on TX endpoint N+3
		01	RNDIS MODE on TX endpoint N+3
		10	CDC Mode on TX endpoint N+3
		11	Generic RNDIS Mode on TX endpoint N+3
5-4	Tx(N+2)_mode	00	Transparent Mode on TX endpoint N+2
		01	RNDIS MODE on TX endpoint N+2
		10	CDC Mode on TX endpoint N+2
		11	Generic RNDIS Mode on TX endpoint N+2
3-2	Tx(N+1)_mode	00	Transparent Mode on TX endpoint N+1
		01	RNDIS MODE on TX endpoint N+1
		10	CDC Mode on TX endpoint N+1
		11	Generic RNDIS Mode on TX endpoint N+1
1-0	Tx(N)_mode	00	Transparent Mode on TX endpoint N
		01	RNDIS MODE on TX endpoint N
		10	CDC Mode on TX endpoint N
		11	Generic RNDIS Mode on TX endpoint N

20.1.14.22 USB Rx Mode Register (Base Address + 0x0074)

The mode register allows the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS enable in the control register overrides this register and enables RNDIS mode for all endpoints.

The USB Rx mode register is shown in Figure 20-25 and described in Table 20-31.

Figure 20-25. USB Rx Mode Register

31 30	29 28	27 26	25 24
Reserved	Rx(N+14)_mode	Rx(N+13)_mode	Rx(N+12)_mode
R	R/W	R/W	R/W
23 22	**21 20**	**19 18**	**17 16**
Rx(N+11)_mode	Rx(N+10)_mode	Rx(N+9)_mode	Rx(N+8)_mode
R/W	R/W	R/W	R/W
15 14	**13 12**	**11 10**	**9 8**
Rx(N+7)_mode	Rx(N+6)_mode	Rx(N+5)_mode	Rx(N+4)_mode
R/W	R/W	R/W	R/W
7 6	**5 4**	**3 2**	**1 0**
Rx(N+3)_mode	Rx(N+2)_mode	Rx(N+1)_mode	Rx(N)_mode
R/W	R/W	R/W	R/W

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

Table 20-31. USB Rx Mode Register Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no affect.
29-28	Rx(N+14)_mode	00	Transparent Mode on RX endpoint N+14
		01	RNDIS MODE on RX endpoint N+14
		10	CDC Mode on RX endpoint N+14
		11	Generic RNDIS Mode on RX endpoint N+14
27-26	Rx(N+13)_mode	00	Transparent Mode on RX endpoint N+13
		01	RNDIS MODE on RX endpoint N+13
		10	CDC Mode on RX endpoint N+13
		11	Generic RNDIS Mode on RX endpoint N+13
25-24	Rx(N+12)_mode	00	Transparent Mode on RX endpoint N+12
		01	RNDIS MODE on RX endpoint N+12
		10	CDC Mode on RX endpoint N+12
		11	Generic RNDIS Mode on RX endpoint N+12
23-22	Rx(N+11)_mode	00	Transparent Mode on RX endpoint N+11
		01	RNDIS MODE on RX endpoint N+11
		10	CDC Mode on RX endpoint N+11
		11	Generic RNDIS Mode on RX endpoint N+11
21-20	Rx(N+10)_mode	00	Transparent Mode on RX endpoint N+10
		01	RNDIS MODE on RX endpoint N+10
		10	CDC Mode on RX endpoint N+10
		11	Generic RNDIS Mode on RX endpoint N+10
19-18	Rx(N+9)_mode	00	Transparent Mode on RX endpoint N+9
		01	RNDIS MODE on RX endpoint N+9
		10	CDC Mode on RX endpoint N+9
		11	Generic RNDIS Mode on RX endpoint N+9

Table 20-31. USB Rx Mode Register Field Descriptions (continued)

Bit	Field	Value	Description
17-16	Rx(N+8)_mode	00	Transparent Mode on RX endpoint N+8
		01	RNDIS MODE on RX endpoint N+8
		10	CDC Mode on RX endpoint N+8
		11	Generic RNDIS Mode on RX endpoint N+8
15-14	Rx(N+7)_mode	00	Transparent Mode on RX endpoint N+7
		01	RNDIS MODE on RX endpoint N+7
		10	CDC Mode on RX endpoint N+7
		11	Generic RNDIS Mode on RX endpoint N+7
13-12	Rx(N+6)_mode	00	Transparent Mode on RX endpoint N+6
		01	RNDIS MODE on RX endpoint N+6
		10	CDC Mode on RX endpoint N+6
		11	Generic RNDIS Mode on RX endpoint N+6
11-10	Rx(N+5)_mode	00	Transparent Mode on RX endpoint N+5
		01	RNDIS MODE on RX endpoint N+5
		10	CDC Mode on RX endpoint N+5
		11	Generic RNDIS Mode on RX endpoint N+5
9-8	Rx(N+4)_mode	00	Transparent Mode on RX endpoint N+4
		01	RNDIS MODE on RX endpoint N+4
		10	CDC Mode on RX endpoint N+4
		11	Generic RNDIS Mode on RX endpoint N+4
7-6	Rx(N+3)_mode	00	Transparent Mode on RX endpoint N+3
		01	RNDIS MODE on RX endpoint N+3
		10	CDC Mode on RX endpoint N+3
		11	Generic RNDIS Mode on RX endpoint N+3
5-4	Rx(N+2)_mode	00	Transparent Mode on RX endpoint N+2
		01	RNDIS MODE on RX endpoint N+2
		10	CDC Mode on RX endpoint N+2
		11	Generic RNDIS Mode on RX endpoint N+2
3-2	Rx(N+1)_mode	00	Transparent Mode on RX endpoint N+1
		01	RNDIS MODE on RX endpoint N+1
		10	CDC Mode on RX endpoint N+1
		11	Generic RNDIS Mode on RX endpoint N+1
1-0	Rx(N)_mode	00	Transparent Mode on RX endpoint N
		01	RNDIS MODE on RX endpoint N
		10	CDC Mode on RX endpoint N
		11	Generic RNDIS Mode on RX endpoint N

20.1.14.23 USB EP Count Mode Register (Base Address + 0x0078)

The EP count mode register is a special feature used only in conjunction with transparent mode that allows the XDMA to receive a preprogrammed number of USB packets and then halting until the count is reprogrammed. When the corresponding bit of this register is set and the endpoint is in transparent mode, the EP size register for that endpoint becomes a USB packet count register. After the XDMA receives a USB packet, it decrements this register by one. If auto request on all is enabled, it sets the packet request bit in the endpoint's CSR register in the Mentor controller. When the XDMA receives a packet and the count value is 1, the XDMA will not issue a packet request to the Mentor controller, but waits for the count value to be reprogrammed before continuing.

The USB EP count mode register is shown in Figure 20-26 and described in Table 20-32.

Figure 20-26. USB EP Count Mode Register

31	30	29	28	27	26	25	24
Reserved			EP15_Count_Mode	Reserved	EP14_Count_Mode	Reserved	EP13_Count_Mode
			R/W		R/W		R/W
23	22	21	20	19	18	17	16
Reserved	EP12_Count_Mode	Reserved	EP11_Count_Mode	Reserved	EP10_Count_Mode	Reserved	EP9_Count_Mode
	R/W		R/W		R/W		R/W
15	14	13	12	11	10	9	8
Reserved	EP8_Count_Mode	Reserved	EP7_Count_Mode	Reserved	EP6_Count_Mode	Reserved	EP5_Count_Mode
	R/W		R/W		R/W		R/W
7	6	5	4	3	2	1	0
Reserved	EP4_Count_Mode	Reserved	EP3_Count_Mode	Reserved	EP2_Count_Mode	Reserved	EP1_Count_Mode
	R/W		R/W		R/W		R/W

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

Table 20-32. USB EP Count Mode Register Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Always read as 0. Writes have no affect.
28	EP15_Count_Mode	0	EP Count Mode Disabled for EP 15
		1	EP Count Mode Enabled for EP 15
27	Reserved	0	Always read as 0. Writes have no affect.
26	EP14_Count_Mode	0	EP Count Mode Disabled for EP 14
		1	EP Count Mode Enabled for EP 14
25	Reserved	0	Always read as 0. Writes have no affect.
24	EP13_Count_Mode	0	EP Count Mode Disabled for EP 13
		1	EP Count Mode Enabled for EP 13
23	Reserved	0	Always read as 0. Writes have no affect.
22	EP12_Count_Mode	0	EP Count Mode Disabled for EP 12
		1	EP Count Mode Enabled for EP 12
21	Reserved	0	Always read as 0. Writes have no affect.
20	EP11_Count_Mode	0	EP Count Mode Disabled for EP 11
		1	EP Count Mode Enabled for EP 11
19	Reserved	0	Always read as 0. Writes have no affect.
18	EP10_Count_Mode	0	EP Count Mode Disabled for EP 10
		1	EP Count Mode Enabled for EP 10

Table 20-32. USB EP Count Mode Register Field Descriptions (continued)

Bit	Field	Value	Description
17	Reserved	0	Always read as 0. Writes have no affect.
16	EP9_Count_Mode	0	EP Count Mode Disabled for EP 9
		1	EP Count Mode Enabled for EP 9
15	Reserved	0	Always read as 0. Writes have no affect.
14	EP8_Count_Mode	0	EP Count Mode Disabled for EP 8
		1	EP Count Mode Enabled for EP 8
13	Reserved	0	Always read as 0. Writes have no affect.
12	EP7_Count_Mode	0	EP Count Mode Disabled for EP 7
		1	EP Count Mode Enabled for EP 7
11	Reserved	0	Always read as 0. Writes have no affect.
10	EP6_Count_Mode	0	EP Count Mode Disabled for EP 6
		1	EP Count Mode Enabled for EP 6
9	Reserved	0	Always read as 0. Writes have no affect.
8	EP5_Count_Mode	0	EP Count Mode Disabled for EP 5
		1	EP Count Mode Enabled for EP 5
7	Reserved	0	Always read as 0. Writes have no affect.
6	EP4_Count_Mode	0	EP Count Mode Disabled for EP 4
		1	EP Count Mode Enabled for EP 4
5	Reserved	0	Always read as 0. Writes have no affect.
4	EP3_Count_Mode	0	EP Count Mode Disabled for EP 3
		1	EP Count Mode Enabled for EP 3
3	Reserved	0	Always read as 0. Writes have no affect.
2	EP2_Count_Mode	0	EP Count Mode Disabled for EP 2
		1	EP Count Mode Enabled for EP 2
1	Reserved	0	Always read as 0. Writes have no affect.
0	EP1_Count_Mode	0	EP Count Mode Disabled for EP 1
		1	EP Count Mode Enabled for EP 1

20.1.14.24 USB Generic RNDIS EP N Size Register (Base Address + 0x0080)

The generic RNDIS EP1 size register is programmed with a RNDIS packet size in bytes. When EP1 is in generic RNDIS mode, the received USB packets collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a “short” packet is received. Note that this register must be programmed with a value that is an integer multiple of the endpoint size. The maximum value this register can be programmed with is 0x10000, or 65536. When the controller is set in transparent EP count mode, this register is known as the EP count register and the number programmed is the number of USB packets to receive before halting.

The USB generic RNDIS EP N size register is shown in Figure 20-27 and described in Table 20-33??.

Figure 20-27. USB Generic RNDIS EP N Size Register

31	17	16	0
Reserved		Ep(N)_size	
		R/W	

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

Table 20-33. USB Generic RNDIS EP N Size Register Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Always read as 0. Writes have no affect.
16-0	Ep(N)_size		Generic RNDIS packet size.

20.1.14.25 USB Queue Interrupt Threshold Enable Register (Base Address + 0x00C0)

The queue interrupt threshold enable register enables the threshold counts that are available for two of the completion queues. When the bit is set, the interrupt threshold mechanism for that queue is enabled.

The USB queue interrupt threshold enable register is shown in Figure 20-28 and described in Table 20-34.

Figure 20-28. USB Queue Interrupt Threshold Enable Register

31	2	1	0
Reserved		Queue65_en	Queue63_en
		R/W	R/W

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

Table 20-34. USB Queue Interrupt Threshold Enable Register Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no affect.
1	Queue65_en		
		0	Threshold Disabled for Completion Queue 65
		1	Threshold Enabled for Completion Queue 65
0	Queue63_en		
		0	Threshold Disabled for Completion Queue 63
		1	Threshold Enabled for Completion Queue 63

20.1.14.26 USB Queue Threshold Register 0 (Base Address + 0x00C4)

The queue threshold register provides a programmable method to delay the completion interrupt until a number of interrupts have been detected. The Host software programs this register with a value between 1 and 255. When the DMA completes a packet designated for completion queue 63, an internal counter is incremented. When this counter reaches the value in this threshold register, an the interrupt and subsequent interrupts are enable to the INTD module, until the interrupt clear register is written and the internal count is reset.

The USB queue threshold register 0 is shown in Figure 20-29 and described in Table 20-35.

Figure 20-29. USB Queue Threshold Register 0

31	8	7	0
Reserved		thrsh_q63	
		R/W	

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

Table 20-35. USB Queue Threshold Register 0 Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Always read as 0. Writes have no affect.
7-0	thrsh_q63		Programmable interrupt threshold.

20.1.14.27 USB Interrupt Clear Register 0 (Base Address + 0x00C8)

When this register is written, the internal threshold count register is cleared. The Host should clear this register whenever the threshold count register value is reached, and an interrupt is generated.

The USB interrupt clear register 0 is shown in Figure 20-30 and described in Table 20-36.

Figure 20-30. USB Interrupt Clear Register 0

31	0
Queue63_en	
W	

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

Table 20-36. USB Interrupt Clear Register 0 Field Descriptions

Bit	Field	Value	Description
31-0	Queue63_en		Clears internal threshold count.

20.1.14.28 USB Queue Threshold Register 1 (Base Address + 0x00D4)

The queue threshold register provides a programmable method to delay the completion interrupt until a number of interrupts have been detected. The Host software programs this register with a value between 1 and 255. When the DMA completes a packet designated for completion queue 65, an internal counter is incremented. When this counter reaches the value in this threshold register, an the interrupt and subsequent interrupts are enable to the INTD module, until the interrupt clear register is written and the internal count is reset.

The USB queue threshold register 1 is shown in Figure 20-31 and described in Table 20-37.

Figure 20-31. USB Queue Threshold Register 1

31	8	7	0
Reserved		thrsh_q65	
		R/W	

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

Table 20-37. USB Queue Threshold Register 1 Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Always read as 0. Writes have no affect.
7-0	thrsh_q65		Programmable interrupt threshold.

20.1.14.29 USB Interrupt Clear Register 1 (Base Address + 0x00D8)

When this register is written, the internal threshold count register is cleared. The Host should clear this register whenever the threshold count register value is reached, and an interrupt is generated.

The USB interrupt clear register 1 is shown in Figure 20-32 and described in Table 20-38.

Figure 20-32. USB Interrupt Clear Register 1

31	0
Queue65_en	
W	

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

Table 20-38. USB Interrupt Clear Register 1 Field Descriptions

Bit	Field	Value	Description
31-0	Queue65_en		Clears internal threshold count.

20.1.14.30 USB Mentor Core Registers/FIFOs (Base Address + 0x400 – 0x59C)

A description of the Mentor core registers is available in the Mentor specification. The core registers are described starting at address 0x000, but they are located in the USB 2.0 OTG module starting at 0x400. To convert, just add 0x400 to the register address in the Mentor core specification.

*20.1.14.30.1 CDMA Tx Channel N Global Configuration Register (Base Address + 0x0800 + 32*N)*

The Tx channel configuration registers are used to initialize the behavior of each of the Tx DMA channels.

The CDMA Tx channel N global configuration register is shown in Figure 20-33 and described in Table 20-39.

Figure 20-33. CDMA Tx Channel N Global Configuration Register

31	30	29	14
tx_enable	tx_teardown	Reserved	
R/W	R/W		

13	12	11	0
tx_default_qmgr		tx_default_qnum	
W		W	

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

Table 20-39. CDMA Tx Channel N Global Configuration Register Field Descriptions

Bit	Field	Value	Description
31	tx_enable		This field enables or disables the channel
		0	Channel is disabled
		1	Channel is enabled
			This field is cleared after a channel teardown is complete.
30	tx_teardown		Setting this bit will request the channel to be torn down. This field remains set after a channel teardown is complete.
29-14	Reserved	0	Always read as 0. Writes have no affect.
13-12	tx_default_qmgr		This field controls the default queue manager number that is used to queue teardown descriptors back to the Host.
11-0	tx_default_qnum		This field controls the default queue number within the selected queue manager onto which teardown descriptors are queued back to the Host.

20.1.14.30.2 CDMA Rx Channel N Global Configuration Register (Base Address + 0x0808 + 32*N)

The Rx channel global configuration registers are used to initialize the global (non descriptor type specific) behavior of each of the Rx DMA channels. If the enable bit is being set, the Rx channel global configuration register should only be written after all of the other Rx configuration registers have been initialized.

The CDMA Rx channel N global N configuration register is shown in Figure 20-34 and described in Table 20-40.

Figure 20-34. CDMA Rx Channel N Global Configuration Register

31	30	29	28	27	26	25	24
rx_enable	rx_teardown	Reserved					rx_error_handling
R/W	R/W						W

23	22	21	20	19	18	17	16
rx_sop_offset							
W							

15	14	13	12	11	10	9	8
rx_default_desc_type		rx_default_rq_qmgr		rx_default_rq_qnum			
W		W		W			

7	6	5	4	3	2	1	0
rx_default_rq_qnum							
W							

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

Table 20-40. CDMA Rx Channel N Global Configuration Register Field Descriptions

Bit	Field	Value	Description
31	rx_enable		This field enables or disables the channel.
		0	Channel is disabled
		1	Channel is enabled
			This field is cleared after a channel teardown is complete.
30	rx_teardown		This field indicates whether or not an Rx teardown operation is complete. This field should be cleared when a channel is initialized. This field is set after a channel teardown is complete.
29-25	Reserved	0	Always read as 0. Writes have no affect.
24	rx_error_handling		This bit controls the error handling mode for the channel and is only used when channel errors (i.e., descriptor or buffer starvation occurs):
		0	Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to
		1	Starvation errors result in subsequent re-try of the descriptor allocation operation. In this mode, the DMA returns to the IDLE state without saving it's internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface. This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.
			Regardless of the value of this bit, the DMA asserts the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when **??**
23-16	rx_sop_offset		This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload. This value must be less than the minimum size of a buffer in the system. Valid values are 0 – 255 bytes.

Table 20-40. CDMA Rx Channel N Global Configuration Register Field Descriptions (continued)

Bit	Field	Value	Description
15-14	rx_default_desc_type		This field indicates the default descriptor type to use:
		0	Reserved
		1	Host
		2	Reserved
		3	Reserved
			The actual descriptor type that is used for reception can be overridden by information provided in the CPPI FIFO data block.
13-12	rx_default_rq_qmgr		This field indicates the default receive queue manager that this channel should use. The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.
11-0	rx_default_rq_qnum		This field indicates the default receive queue that this channel should use. The actual receive queue that is used for reception can be overridden by information provided in the CPPI FIFO data block.

20.1.14.30.3 CDMA Rx Channel N Host Packet Configuration Register A (Base Address + 0x080C + 32*N)

The Rx channel Host packet configuration registers are used to initialize the behavior of each of the Rx DMA channels for reception of Host type packets.

The CDMA Rx channel N Host packet configuration register A is shown in Figure 20-35 and described in Table 20-41.

Figure 20-35. CDMA Rx Channel N Host Packet Configuration Register A

31	30	29	28	27	16
Reserved		rx_host_fdq1_qmgr		rx_host_fdq1_qnum	
		W		W	

15	14	13	12	11	0
Reserved		rx_host_fdq0_qmgr		rx_host_fdq0_qnum	
		W		W	

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

Table 20-41. CDMA Rx Channel N Host Packet Configuration Register A Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no affect.
29-28	rx_host_fdq1_qmgr		This field specifies which Buffer Manager should be used for the second Rx buffer in a Host type packet.
27-16	rx_host_fdq1_qnum		This field specifies which Free Descriptor / Buffer Pool should be used for the second Rx buffer in a Host type packet.
15-14	Reserved	0	Always read as 0. Writes have no affect.
13-12	rx_host_fdq0_qmgr		This field specifies which Buffer Manager should be used for the first Rx buffer in a Host type packet.
11-0	rx_host_fdq0_qnum		This field specifies which Free Descriptor / Buffer Pool should be used for the first Rx buffer in a Host type packet.

20.1.14.30.4 CDMA Rx Channel N Host Packet Configuration Register B (Base Address + 0x0810 + 32*N)

The Rx channel Host packet configuration registers are used to initialize the behavior of each of the Rx DMA channels for reception of Host type packets.

The CDMA Rx channel N Host packet configuration register B is shown in Figure 20-36 and described in Table 20-42.

Figure 20-36. CDMA Rx Channel N Host Packet Configuration Register B

31	30	29	28	27	16
Reserved		rx_host_fdq3_qmgr		rx_host_fdq3_qnum	
		W		W	

15	14	13	12	11	0
Reserved		rx_host_fdq2_qmgr		rx_host_fdq2_qnum	
		W		W	

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

Table 20-42. CDMA Rx Channel N Host Packet Configuration Register B Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no affect.
29-28	rx_host_fdq1_qmgr		This field specifies which Buffer Manager should be used for the fourth or later Rx buffers in a Host type packet.
27-16	rx_host_fdq1_qnum		This field specifies which Free Descriptor queue should be used for the fourth or later Rx buffers in a Host type packet.
15-14	Reserved	0	Always read as 0. Writes have no affect.
13-12	rx_host_fdq0_qmgr		This field specifies which Buffer Manager should be used for the third Rx buffer in a Host type packet.
11-0	rx_host_fdq0_qnum		This field specifies which Free Descriptor / Buffer Pool should be used for the third Rx buffer in a Host type packet.

20.1.14.30.5 CDMA Scheduler Control Register (Base Address + 0x0C00)

The revision register contains the major and minor revisions for the module.

The CDMA scheduler control register is shown in Figure 20-37 and described in Table 20-43??.

Figure 20-37. CDMA Scheduler Control Register

31	30	8	7	0
Enable	Reserved		last_entry	
R/W			R/W	

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

Table 20-43. CDMA Scheduler Control Register Field Descriptions

Bit	Field	Value	Description
31	Enable		This is the enable bit for the scheduler and is encoded as follows:
		0	Scheduler is disabled and will no longer fetch entries from the scheduler table or pass credits to the DMA controller
		1	Scheduler is enabled
			This bit should only be set after the table has been initialized.
30-8	Reserved	0	Always read as 0. Writes have no affect.
7-0	last_entry		This field indicates the last valid entry in the scheduler table. There are 64 words in the table and there are 4 entries in each word. The table can be programmed with any integer number of entries from 1 to 256. The corresponding encoding for this field is as follows:
		0	1 entry
		1	2 entries
		…	
		254	255 entries
		255	256 entries

20.1.14.30.6 CDMA Scheduler Table Word N Registers (Base Address + 0x0D00:0DC00 + 4*N)

The Tx channel configuration registers are used to initialize the behavior of each of the Tx DMA channels.

The CDMA scheduler table word N registers are shown in Figure 20-38 and described in Table 20-44.

Figure 20-38. CDMA Scheduler Table Word N Registers

31	30	28	27	24
entry3_rxtx	Reserved		entry3_channel	
W			W	
23	22	20	19	16
entry2_rxtx	Reserved		entry2_channel	
W			W	
15	14	12	11	8
entry1_rxtx	Reserved		entry1_channel	
W			W	
7	6	5 4	3	0
entry0_rxtx	Reserved		entry0_channel	
W			W	

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

Table 20-44. CDMA Scheduler Table Word N Registers Field Descriptions

Bit	Field	Value	Description
31	entry3_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
30-28	Reserved	0	Always read as 0. Writes have no affect.
27-24	entry3_channel		This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA is presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA is presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit is the channel number that is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
23	entry2_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
22-20	Reserved	0	Always read as 0. Writes have no affect.
19-16	entry2_channel		This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA is presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA is presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit is the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
15	entry1_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel
14-12	Reserved	0	Always read as 0. Writes have no affect.
11-8	entry1_channel		This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA is presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA is presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit is the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
7	entry0_rxtx		This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:
		0	Tx Channel
		1	Rx Channel

Table 20-44. CDMA Scheduler Table Word N Registers Field Descriptions (continued)

Bit	Field	Value	Description
6-4	Reserved	0	Always read as 0. Writes have no affect.
3-0	entry0_channel		This field indicates the channel # that is to be given an opportunity to transfer data. If this is a Tx entry, the DMA is presented with a scheduling 'credit' for that exact Tx channel. If this is an Rx entry, the DMA is presented with a scheduling credit for the Rx FIFO that is associated with this channel. For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit is the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.

20.1.14.31 INTD Revision Register (Base Address + 0x3000)

The revision register contains the ID and revision information.

The INTD revision register is shown in Figure 20-39 and described in Table 20-45.

Figure 20-39. INTD Revision Register

31	30	29	16
scheme		modID	
R		R	

15	11	10	8	7	0
revrtl		revmaj		revmin	
R		R		R	

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

Table 20-45. INTD Revision Register Field Descriptions

Bit	Field	Value	Description
31-30	scheme		Always read as 1. Writes have no affect.
29-16	modID		Module ID field.
15-11	revrtl		RTL revision. Will vary depending on release.
10-8	revmaj		Major revision.
7-0	revmin		Minor revision.

20.1.14.32 INTD EOI Register (Base Address + 0x3010)

The EOI register allows the software to perform an end of interrupt handshake to interrupts with an IP that does not have an EOI register. The EOI handshake allows system interrupts to be re-evaluated for the associate logic with the EOI vector value to allow multiple logic paths to re-evaluate individually without separate EOI registers. The EOI occurs the cycle after the register is written.

The INTD EOI register is shown in Figure 20-40 and described in Table 20-46.

Figure 20-40. INTD EOI Register

31	8	7	0
Reserved		eoi_vector	
		R/W	

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

Table 20-46. INTD EOI Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Always read as 0. Writes have no affect.
7-0	eoi_vector		EOI Vector value. This should be written with the value associate to the interrupt and Host combination given in the cp_intd configuration.

20.1.14.33 INTD EOI Interrupt Vector Register (Base Address + 0x3014)

The EOI interrupt vector register captures the active and prioritized interrupts so that software can quickly read the values rather than check every bit in the IP interrupt source register. The exact definition of this register is configuration dependent.

The INTD EOI interrupt vector register is shown in Figure 20-41 and described in Table 20-47.

Figure 20-41. INTD EOI Interrupt Vector Register

31	0
intr_vector	
R	

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

Table 20-47. INTD EOI Interrupt Vector Register

Bit	Field	Value	Description
31-0	intr_vector		EOI interrupt vector value. This has the latest prioritized interrupt values. Dependent on configuration.

20.1.14.34 INTD Status Register 0 (Base Address + 0x3200)

The status register indicates which IP interrupts are active for the Host. There is at least one register per Host for up to 32 interrupts. The exact interrupt mapping is determined by the configuration and then those registers, could be more then 1, are replicated for each Host. The mapping of bits in this register matches that of the enable registers. For level interrupts the status bit is read only and is always determined by the input interrupt level.

The INTD status register 0 is shown in Figure 20-42 and described in Table 20-48.

Figure 20-42. INTD Status Register 0

31	2	1	0
Reserved		rx_sbuf	rx_sdes
		R	R

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

Table 20-48. INTD Status Register 0 Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no affect.
1	rx_sbuf		Rx MOP Descriptor Starvation Status
		0	Not pending
		1	Pending
0	rx_sdes		Rx SOP Descriptor Starvation Status
		0	Not pending
		1	Pending

20.1.14.35 INTD Status Register 1 (Base Address + 0x3204)

The status register indicates which IP interrupts are active for the Host. There is at least one register per Host for up to 32 interrupts. The exact interrupt mapping is determined by the configuration and then those registers, could be more then 1, are replicated for each Host. The mapping of bits in this register matches that of the enable registers. For level interrupts the status bit is read only and is always determined by the input interrupt level.

The INTD status register 1 is shown in Figure 20-43 and described in Table 20-49.

Figure 20-43. INTD Status Register 1

31						10	9	8
Reserved							USB	usb_int8
							R	R
7	6	5	4	3	2	1	0	
usb_int7	usb_int6	usb_int5	usb_int4	usb_int3	usb_int2	usb_int1	usb_int0	
R	R	R	R	R	R	R	R	

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

Table 20-49. INTD Status Register 1 Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Always read as 0. Writes have no effect.
9	USB		USB Core Interrupt Status
		0	Not pending
		1	Pending
8	usb_int8		USB Interrupt 8 Status
		0	Not pending
		1	Pending
7	usb_int7		USB Interrupt 7 Status
		0	Not pending
		1	Pending
6	usb_int6		USB Interrupt 6 Status
		0	Not pending
		1	Pending
5	usb_int5		USB Interrupt 5 Status
		0	Not pending
		1	Pending
4	usb_int4		USB Interrupt 4 Status
		0	Not pending
		1	Pending
3	usb_int3		USB Interrupt 3 Status
		0	Not pending
		1	Pending
2	usb_int2		USB Interrupt 2 Status
		0	Not pending
		1	Pending
1	usb_int1		USB Interrupt 1 Status
		0	Not pending
		1	Pending
0	usb_int0		USB Interrupt 0 Status
		0	Not pending
		1	Pending

20.1.14.36 INTD Status Register 2 (Base Address + 0x3208)

The status register indicates which IP interrupts are active for the Host. There is at least one register per Host for up to 32 interrupts. The exact interrupt mapping is determined by the configuration and then those registers, could be more then 1, are replicated for each Host. The mapping of bits in this register matches that of the enable registers. For level interrupts the status bit is read only and is always determined by the input interrupt level.

The INTD status register 2 is shown in Figure 20-44 and described in Table 20-50.

Figure 20-44. INTD Status Register 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	rx_ep15	rx_ep14	rx_ep13	rx_ep12	rx_ep11	rx_ep10	rx_ep9	rx_ep8	rx_ep7	rx_ep6	rx_ep5	rx_ep4	rx_ep3	rx_ep2	rx_ep1
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tx_ep15	tx_ep14	tx_ep13	tx_ep12	tx_ep11	tx_ep10	tx_ep9	tx_ep8	tx_ep7	tx_ep6	tx_ep5	tx_ep4	tx_ep3	tx_ep2	tx_ep1	tx_ep0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

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

Table 20-50. INTD Status Register 2 Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Always read as 0. Writes have no effect.
30	rx_ep15		USB RX EP15 Interrupt Status
		0	Not pending
		1	Pending
29	rx_ep14		USB RX EP14 Interrupt Status
		0	Not pending
		1	Pending
28	rx_ep13		USB RX EP13 Interrupt Status
		0	Not pending
		1	Pending
27	rx_ep12		USB RX EP12 Interrupt Status
		0	Not pending
		1	Pending
26	rx_ep11		USB RX EP11 Interrupt Status
		0	Not pending
		1	Pending
25	rx_ep10		USB RX EP10 Interrupt Status
		0	Not pending
		1	Pending
24	rx_ep9		USB RX EP9 Interrupt Status
		0	Not pending
		1	Pending
23	rx_ep8		USB RX EP8 Interrupt Status
		0	Not pending
		1	Pending
22	rx_ep7		USB RX EP7 Interrupt Status
		0	Not pending
		1	Pending

Table 20-50. INTD Status Register 2 Field Descriptions (continued)

Bit	Field	Value	Description
21	rx_ep6		USB RX EP6 Interrupt Status
		0	Not pending
		1	Pending
20	rx_ep5		USB RX EP5 Interrupt Status
		0	Not pending
		1	Pending
19	rx_ep4		USB RX EP4 Interrupt Status
		0	Not pending
		1	Pending
18	rx_ep3		USB RX EP3 Interrupt Status
		0	Not pending
		1	Pending
17	rx_ep2		USB RX EP2 Interrupt Status
		0	Not pending
		1	Pending
16	rx_ep1		USB RX EP1 Interrupt Status
		0	Not pending
		1	Pending
15	tx_ep15		USB TX EP16 Interrupt Status
		0	Not pending
		1	Pending
14	tx_ep14		USB TX EP16 Interrupt Status
		0	Not pending
		1	Pending
13	tx_ep13		USB TX EP16 Interrupt Status
		0	Not pending
		1	Pending
12	tx_ep12		USB TX EP16 Interrupt Status
		0	Not pending
		1	Pending
11	tx_ep11		USB TX EP16 Interrupt Status
		0	Not pending
		1	Pending
10	tx_ep10		USB TX EP15 Interrupt Status
		0	Not pending
		1	Pending
9	tx_ep9		USB TX EP14 Interrupt Status
		0	Not pending
		1	Pending
8	tx_ep8		USB TX EP13 Interrupt Status
		0	Not pending
		1	Pending
7	tx_ep7		USB TX EP12 Interrupt Status
		0	Not pending
		1	Pending

Table 20-50. INTD Status Register 2 Field Descriptions (continued)

Bit	Field	Value	Description
6	tx_ep16		USB TX EP11 Interrupt Status
		0	Not pending
		1	Pending
5	tx_ep15		USB TX EP10 Interrupt Status
		0	Not pending
		1	Pending
4	tx_ep4		USB TX EP9 Interrupt Status
		0	Not pending
		1	Pending
3	tx_ep3		USB TX EP8 Interrupt Status
		0	Not pending
		1	Pending
2	tx_ep2		USB TX EP7 Interrupt Status
		0	Not pending
		1	Pending
1	tx_ep1		USB TX EP6 Interrupt Status
		0	Not pending
		1	Pending
0	tx_ep0		USB TX EP5 Interrupt Status
		0	Not pending
		1	Pending

20.1.14.37 INTD Status Register 3 (Base Address + 0x320C)

The status register indicates which IP interrupts are active for the Host. There is at least one register per Host for up to 32 interrupts. The exact interrupt mapping is determined by the configuration and then those registers, could be more then 1, are replicated for each Host. The mapping of bits in this register matches that of the enable registers. For level interrupts the status bit is read only and is always determined by the input interrupt level.

The INTD status register 3 is shown in Figure 20-45 and described in Table 20-51.

Figure 20-45. INTD Status Register 3

31 16	15 14	13 2	1 0
Reserved	Comp_int	Reserved	Comp_int
	R		R

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

Table 20-51. INTD Status Register 3 Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Always read as 0. Writes have no effect.
15-14	Comp_int		Completion Interrupts for Queues 66 and 65 Status
		0	Not pending
		1	Pending
13-2	Reserved	0	Always read as 0. Writes have no effect.
1-0	Comp_int		Completion Interrupts for Queues 63 and 64 Status
		0	Not pending
		1	Pending

20.1.14.38 INTD Status Clear Register 0 (Base Address + 0x280)

The status clear register allows software to clear the pending status. This register is only valid and functional for pulsed interrupts and has no effect for level interrupts. There is at least one register per Host for up to 32 interrupts. The mapping of bits in this register matches that of the enable registers.

The INTD status clear register 0 is shown in Figure 20-46 and described in Table 20-52.

Figure 20-46. INTD Status Clear Register 0

31	2	1	0
Reserved		status_clr	status_clr
		W	W

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

Table 20-52. INTD Status Clear Register 0 Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Always read as 0. Writes have no affect.
1	status_clr		RX MOP Descriptor interrupt status.
		0	Not pending
		1	Pending
			Write a 1 in a bit position to clear that bit. Writing a 0 has no effect.
0	status_clr		RX SOP Descriptor Starve interrupt status.
		0	Not pending
		1	Pending
			Write a 1 in a bit position to clear that bit. Writing a 0 has no effect.

20.1.14.39 Queue Manager Revision Register (0x4000)

The revision register contains the major and minor revisions for the module. It does not support byte accesses.

The queue manager revision register is shown in Figure 20-47 and described in Table 20-53.

Figure 20-47. Queue Manager Revision Register

31	30	29	28	27	16
scheme		Reserved		function	
R				R	

15	11	10	8	7	6	5	0
revrtl		revmaj		revcustom		revmin	
R		R		R		R	

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

Table 20-53. Queue Manager Revision Register

Bit	Field	Value	Description
31-30	scheme		Scheme that this register is compliant with
29-28	Reserved	0	Always read as 0. Writes have no effect.
27-16	function		Function
15-11	revrtl		RTL revision
10-8	revmaj		Major revision
7-6	revcustom		Custom revision
5-0	revmin		Minor revision

20.1.14.40 Queue Manager Queue Diversion Register (0x4008)

The queue diversion register is used to transfer the contents of one queue onto another queue. It does not support byte accesses.

The queue manager queue diversion register is shown in Figure 20-48 and described in Table 20-54.

Figure 20-48. Queue Manager Queue Diversion Register

31	30	29	16
head_tail	Reserved	dest_qnum	
W		W	

15	14	13	0
Reserved		source_qnum	
		W	

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

Table 20-54. Queue Manager Queue Diversion Register Field Descriptions

Bit	Field	Value	Description
31	head_tail		head_tail indicates whether the queue contents should be merged onto the head or the tail of the destination queue. Clear this field for head and set for tail.
30	Reserved	0	Always read as 0. Writes have no effect.
29-16	dest_qnum		Destination Queue Number
15-14	Reserved	0	Always read as 0. Writes have no effect.
13-0	source_qnum		Source Queue Number

20.1.14.41 Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (0x4020)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register is shown in Figure 20-49 and described in Table 20-55.

Figure 20-49. Queue Manager Free Descriptor/Buffer Starvation Count Register 0

31 24	23 16	8	0
fdbq3_starve_cnt	fdbq2_starve_cnt	fdbq1_starve_cnt	fdbq0_starve_cnt
COR	COR	COR	COR

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

Table 20-55. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq3_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 3 is read while it is empty. This field is cleared when read.
23-16	fdbq2_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 2 is read while it is empty. This field is cleared when read.
15-8	fdbq1_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 1 is read while it is empty. This field is cleared when read.
7-0	fdbq0_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 0 is read while it is empty. This field is cleared when read.

20.1.14.42 Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (0x4024)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 1 is shown in Figure 20-50 and described in Table 20-56.

Figure 20-50. Queue Manager Free Descriptor/Buffer Starvation Count Register 1

31 24	23 16	8	0
fdbq7_starve_cnt	fdbq6_starve_cnt	fdbq5_starve_cnt	fdbq4_starve_cnt
COR	COR	COR	COR

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

Table 20-56. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq7_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 7 is read while it is empty. This field is cleared when read.
23-16	fdbq6_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 6 is read while it is empty. This field is cleared when read.
15-8	fdbq5_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 5 is read while it is empty. This field is cleared when read.
7-0	fdbq4_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 4 is read while it is empty. This field is cleared when read.

20.1.14.43 Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (0x4028)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 2 is shown in Figure 20-51 and described in Table 20-57.

Figure 20-51. Queue Manager Free Descriptor/Buffer Starvation Count Register 2

31 24	23 16	8	0
fdbq7_starve_cnt	fdbq6_starve_cnt	fdbq5_starve_cnt	fdbq4_starve_cnt
COR	COR	COR	COR

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

Table 20-57. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq7_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 7 is read while it is empty. This field is cleared when read.
23-16	fdbq6_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 6 is read while it is empty. This field is cleared when read.
15-8	fdbq5_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 5 is read while it is empty. This field is cleared when read.
7-0	fdbq4_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 4 is read while it is empty. This field is cleared when read.

20.1.14.44 Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (0x402c)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 3 is shown in Figure 20-52 and described in Table 20-58.

Figure 20-52. Queue Manager Free Descriptor/Buffer Starvation Count Register 3

31 24	23 16	8	0
fdbq15_starve_cnt	fdb14_starve_cnt	fdbq13_starve_cnt	fdbq12_starve_cnt
COR	COR	COR	COR

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

Table 20-58. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq15_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 15 is read while it is empty. This field is cleared when read.
23-16	fdbq14_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 14 is read while it is empty. This field is cleared when read.
15-8	fdbq13_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 13 is read while it is empty. This field is cleared when read.
7-0	fdbq12_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 12 is read while it is empty. This field is cleared when read.

20.1.14.45 Queue Manager Free Descriptor/Buffer Starvation Count Register 4 (0x4030)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 4 is shown in Figure 20-53 and described in Table 20-59.

Figure 20-53. Queue Manager Free Descriptor/Buffer Starvation Count Register 4

31 24	23 16	8	0
fdbq19_starve_cnt	fdb18_starve_cnt	fdbq17_starve_cnt	fdbq16_starve_cnt
COR	COR	COR	COR

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

Table 20-59. Queue Manager Free Descriptor/Buffer Starvation Count Register 4 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq19_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 19 is read while it is empty. This field is cleared when read.
23-16	fdbq18_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 18 is read while it is empty. This field is cleared when read.
15-8	fdbq17_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 17 is read while it is empty. This field is cleared when read.
7-0	fdbq16_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 16 is read while it is empty. This field is cleared when read.

20.1.14.46 Queue Manager Free Descriptor/Buffer Starvation Count Register 5 (0x4034)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 5 is shown in Figure 20-54 and described in Table 20-60.

Figure 20-54. Queue Manager Free Descriptor/Buffer Starvation Count Register 5

31 24	23 16	8	0
fdbq23_starve_cnt	fdb22_starve_cnt	fdbq21_starve_cnt	fdbq20_starve_cnt
COR	COR	COR	COR

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

Table 20-60. Queue Manager Free Descriptor/Buffer Starvation Count Register 5 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq23_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 23 is read while it is empty. This field is cleared when read.
23-16	fdbq22_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 22 is read while it is empty. This field is cleared when read.
15-8	fdbq21_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 21 is read while it is empty. This field is cleared when read.
7-0	fdbq20_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 20 is read while it is empty. This field is cleared when read.

20.1.14.47 Queue Manager Free Descriptor/Buffer Starvation Count Register 6 (0x4038)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 6 is shown in Figure 20-55 and described in Table 20-61.

Figure 20-55. Queue Manager Free Descriptor/Buffer Starvation Count Register 6

31 24	23 16	8	0
fdbq27_starve_cnt	fdb26_starve_cnt	fdbq25_starve_cnt	fdbq24_starve_cnt
COR	COR	COR	COR

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

Table 20-61. Queue Manager Free Descriptor/Buffer Starvation Count Register 6 Field Descriptions

Bit	Field	Value	Description
31-24	fdbq27_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 27 is read while it is empty. This field is cleared when read.
23-16	fdbq26_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 26 is read while it is empty. This field is cleared when read.
15-8	fdbq25_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 25 is read while it is empty. This field is cleared when read.
7-0	fdbq24_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 24 is read while it is empty. This field is cleared when read.

20.1.14.48 Queue Manager Free Descriptor/Buffer Starvation Count Register 7 (0x403C)

The free descriptor/buffer queue starvation count register provides statistics about how many starvation events are occurring on the Rx free descriptor/buffer queues. It does not support byte accesses.

The queue manager free descriptor/buffer starvation count register 6 is shown in Figure 20-56 and described in Table 20-62.

Figure 20-56. Queue Manager Free Descriptor/Buffer Starvation Count Register 7

31 24	23 16	8	0
fdbq31_starve_cnt	fdb30_starve_cnt	fdbq29_starve_cnt	fdbq28_starve_cnt
COR	COR	COR	COR

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

Table 20-62. Queue Manager Free Descriptor/Buffer Starvation Count Register 7 Field Descriptions

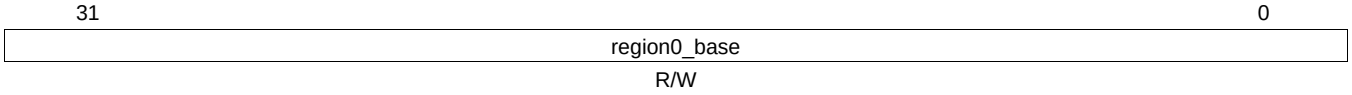
Bit	Field	Value	Description
31-24	fdbq31_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 31 is read while it is empty. This field is cleared when read.
23-16	fdbq30_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 30 is read while it is empty. This field is cleared when read.
15-8	fdbq29_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 29 is read while it is empty. This field is cleared when read.
7-0	fdbq28_starve_cnt		This field increments each time the Free Descriptor/Buffer Queue 28 is read while it is empty. This field is cleared when read.

20.1.14.49 Queue Manager Linking RAM Region 0 Base Address Register (0x4080)

The linking RAM region 0 base address register is used to set the base address for the first portion of the linking RAM. This address must be 32-bit aligned. It is used by the queue manager to calculate the 32-bit linking address for a given descriptor index. It does not support byte accesses.

The queue manager linking RAM region 0 base address register is shown in Figure 20-57 and described in Table 20-63.

Figure 20-57. Queue Manager Linking RAM Region 0 Base Address Register



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

Table 20-63. Queue Manager Linking RAM Region 0 Base Address Register Field Descriptions

Bit	Field	Value	Description
31-0	region0_base		This field stores the base address for the first region of the linking RAM. This can be anywhere in 32-bit address space but would be typically located in on-chip memory.

20.1.14.50 Queue Manager Linking RAM Region 0 Size Register (0x4084)

The linking RAM region 0 size register is used to set the size of the array of linking pointers that are located in region 0 of linking RAM. The size specified the number of descriptors for which linking information is stored in this region. It does not support byte accesses.

The queue manager linking RAM region 0 size register is shown in Figure 20-58 and described in Table 20-64.

Figure 20-58. Queue Manager Linking RAM Region 0 Size Register

15	14	13	0
Reserved		region0_size	
		R/W	

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

Table 20-64. Queue Manager Linking RAM Region 0 Size Register Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Always read as 0. Writes have no effect.
13-0			This field indicates the number of entries that are contained in the linking RAM region 0. A descriptor with index less than region0_size value has its linking location in region 0. A descriptor with index greater than region0_size has its linking location in region 1. The queue manager adds the index (left shifted by 2 bits) to the appropriate regionX_base_addr to get the absolute 32-bit address to the linking location for a descriptor.

20.1.14.51 Queue Manager Linking RAM Region 1 Base Address Register (0x4088)

The linking RAM region 1 base address register is used to set the base address for the second portion of the linking RAM. This base address is used by the queue manager to calculate the 32-bit linking address from the descriptor index. All descriptors with index higher than that given in linking RAM 0 size register have linking information stored in linking RAM region 1. It does not support byte accesses.

The queue manager linking RAM region 1 base address register is shown in Figure 20-59 and described in Table 20-65.

Figure 20-59. Queue Manager Linking RAM Region 1 Base Address Register

31	0
region1_base	
R/W	

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

Table 20-65. Queue Manager Linking RAM Region 1 Base Address Register Field Descriptions

Bit	Field	Value	Description
31-0	region1_base		This field stores the base address for the second region of the linking RAM. This can be anywhere in 32-bit address space but would be typically located in on-chip memory.

20.1.14.52 Queue Manager Queue Pending Register 0 (0x4090)

The queue pending register 0 can be read to find the pending status for queues 31 to 0. It does not support byte accesses.

The queue manager queue pending register 0 is shown in Figure 20-60 and described in Table 20-66.

Figure 20-60. Queue Manager Queue Pending Register 0

31	0
qpend0	
R/W	

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

Table 20-66. Queue Manager Queue Pending Register 0 Field Descriptions

Bit	Field	Value	Description
31-0	qpend0		This field indicates the queue pending status for queues[31:0].

20.1.14.53 Queue Manager Queue Pending Register 1 (0x4094)

The queue pending register 1 can be read to find the pending status for queues 63 to 32. It does not support byte accesses.

The queue manager queue pending register 1 is shown in Figure 20-61 and described in Table 20-67.

Figure 20-61. Queue Manager Queue Pending Register 1

31	0
qpend1	
R/W	

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

Table 20-67. Queue Manager Queue Pending Register 1 Field Descriptions

Bit	Field	Value	Description
31-0	qpend1		This field indicates the queue pending status for queues[63:32].

20.1.14.54 Queue Manager Queue Pending Register 2 (0x4098)

The queue pending register 2 can be read to find the pending status for queues 95 to 64. It does not support byte accesses.

The queue manager queue pending register 2 is shown in Figure 20-62 and described in Table 20-68.

Figure 20-62. Queue Manager Queue Pending Register 2

31	0
qpend2	
R	

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

Table 20-68. Queue Manager Queue Pending Register 2 Field Descriptions

Bit	Field	Value	Description
31-0	qpend2		This field indicates the queue pending status for queues[95:64].

20.1.14.55 Queue Manager Memory Region R Base Address Register (0x5000 + 16xR)

The memory region R base address register is written by the Host to set the base address of memory region R. This memory region stores a number of descriptors of a particular size as determined by the memory region R control register. It does not support byte accesses.

The queue manager memory region R base address register is shown inFigure 20-63 and described inTable 20-69.

Figure 20-63. Queue Manager Memory Region R Base Address Register

31	0
reg	
R/W	

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

Table 20-69. Queue Manager Memory Region R Base Address Register Field Descriptions

Bit	Field	Value	Description
31-0	reg		This field contains the base address of the memory region R.

20.1.14.56 Queue Manager Memory Region R Control Register (0x5000 + 16xR + 4)

The memory region R control register is written by the Host to configure various parameters of this memory region. It does not support byte accesses.

The queue manager memory region R control register is shown in Figure 20-64 and described in Table 20-70.

Figure 20-64. Queue Manager Memory Region R Control Register

31	30	29	16
Reserved		start_index	
		R/W	

15	12	11	8	7	3	2	0
Reserved		desc_size		Reserved		reg_size	
		R/W				R/W	

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

Table 20-70. Queue Manager Memory Region R Control Register Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Always read as 0. Writes have no effect.
29-16	start_index		This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.
15-12	Reserved	0	Always read as 0. Writes have no effect.
11-8	desc_size		This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as 2^(5+desc_size) number of bytes. The settings of desc_size from 9-15 are reserved.
7-3	Reserved	0	Always read as 0. Writes have no effect.
2-0	reg_size		This field indicates the size of the memory region (in terms of number of descriptors). It is an encoded value that specifies region size as 2^(5+reg_size) number of descriptors.

The following sections describe each of the four register locations that may be present for each queue in the queues region. For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index. Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet add or a single burst read for each packet pop operation. The length of a burst to add or pop a packet varies depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes. Queue N register D must always be written/read in the burst but the preceding words are optional depending on the required queue functionality.

20.1.14.57 Queue Manager Queue N Register A (0x6000 + 16xN)

The queue N register A is an optional register that is only implemented for a queue if the queue supports entry/byte count feature. The entry count feature provides a count of the number of entries that are currently valid in the queue. It does not support byte accesses.

The queue manager queue N register is shown in Figure 20-65 and described in Table 20-71.

Figure 20-65. Queue Manager Queue N Register A

31	14	13	0
Reserved		queue_entry_count	
		R	

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

Table 20-71. Queue Manager Queue N Register A Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Always read as 0. Writes have no effect.
13-0	queue_entry_count		This field indicates how many packets are currently queued on the queue. This count is incremented by 1 whenever a packet is added to the queue. This count is decremented by 1 whenever a packet is popped from the queue.

20.1.14.58 Queue Manager Queue N Register B (0x6000 + 16xN + 4)

The queue N register B is an optional register that is only implemented for a queue if the queue supports a total byte count feature. The total byte count feature provides a count of the total number of bytes in all of the packets that are currently valid in the queue. This register must be read prior to reading queue N register D during packet pop operation if the total size information is desired. It does not support byte accesses.

The queue manager queue N register B is shown in Figure 20-66 and described in Table 20-72.

Figure 20-66. Queue Manager Queue N Register B

31 28	27 0
Reserved	queue_byte_count
	R

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

Table 20-72. Queue Manager Queue N Register B Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Always read as 0. Writes have no effect.
27-0	queue_byte_count		This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.

20.1.14.59 Queue Manager Queue N Register C (0x6000 + 16xN + 8)

The queue N register C is used to provide additional information about the packet that is being pushed or popped from the queue. This register provides an option for the packet to be pushed onto either the tail of the queue (default) or the head of the queue (override). This register must be written prior to writing the queue N register D during packet write operations. This register must be read prior to reading queue N register D during pop operations if the packet size information is desired. It does not support byte accesses.

The queue manager queue N register c is shown in Figure 20-67 and described in Table 20-73.

Figure 20-67. Queue Manager Queue N Register C

31	30 14	13 0
head_tail	Reserved	packet_size
W		R/W

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

Table 20-73. Queue Manager Queue N Register C Field Descriptions

Bit	Field	Value	Description
31	head_tail		Head/Tail Push Control. Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.
30-14	Reserved	0	Always read as 0. Writes have no effect.
13-0	packet_size		This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation.

20.1.14.60 Queue Manager Queue N Register D (0x6000 + 16xN + C)

The queue N register D is written to add a packet to the queue and read to pop a packets off a queue. The packet is only pushed or popped to/from the queue when the queue register D is written. It does not support byte accesses.

The queue manager queue N register D is shown in Figure 20-68 and described in Table 20-74.

Figure 20-68. Queue Manager Queue N Register D

31	5	4	0
desc_ptr		desc_size	
		R/W	

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

Table 20-74. Queue Manager Queue N Register D Field Descriptions

Bit	Field	Value	Description
31-5	desc_ptr	0	Descriptor pointer. It is read as zero if the queue is empty. It indicates a 32-bit aligned address that points to a descriptor when the queue is not empty.
4-0	desc_size		Descriptor Size. It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes. This field returns a 0x0 when an empty queue is read.

20.1.14.61 Queue Manager Queue N Status Register A (0x6800 + 16xN)

The queue N status register A is an optional register that is only implemented for a queue if the queue supports entry/byte count feature. The entry count feature provides a count of the number of entries that are currently valid in the queue. It does not support byte accesses.

The queue manager queue N status register A is shown in Figure 20-69 and described in Table 20-75.

Figure 20-69. Queue Manager Queue N Status Register A

31	14	13	0
Reserved		queue_entry_count	
		R	

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

Table 20-75. Queue Manager Queue N Status Register A Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Always read as 0. Writes have no effect.
13-0	queue_entry_count		This field indicates how many packets are currently queued on the queue.

20.1.14.62 Queue Manager Queue N Status Register B (0x6800 + 16xN + 4)

The queue N status register B is an optional register that is only implemented for a queue if the queue supports a total byte count feature. The total byte count feature provides a count of the total number of bytes in all of the packets that are currently valid in the queue. It does not support byte accesses.

The queue manager queue N status register B is shown in Figure 20-70 and described in Table 20-76.

Figure 20-70. Queue Manager Queue N Status Register B

31 28	27 0
Reserved	queue_byte_count
	R

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

Table 20-76. Queue Manager Queue N Status Register B Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Always read as 0. Writes have no effect.
27-0	queue_byte_count		This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.

20.1.14.63 Queue Manager Queue N Status Register C (0x6800 + 16xN + 8)

The queue N status register C specifies the packet size for the head element of a queue. It does not support byte accesses.

The queue manager queue N status requester C is shown in Figure 20-71 and described in Table 20-77.

Figure 20-71. Queue Manager Queue N Status Register C

31 14	13 0
Reserved	packet_size
	R

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

Table 20-77. Queue Manager Queue N Status Register C Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Always read as 0. Writes have no effect.
13-0	packet_size		This field indicates packet size of the head element of a queue.

20.2 High-Speed USB Host Subsystem

20.2.1 High-Speed USB Host Subsystem Overview

The high-speed universal serial bus (USB) host subsystem is composed of the high-speed multiport USB host controller and the USBTLL module.

The USB controller is a high-speed multiport USB2.0 host controller. It contains two independent, 3-port host controllers that operate in parallel:

- The EHCI controller, based on the *Enhanced Host Controller Interface (EHCI) specification for USB Release 1.0*, is in charge of high-speed traffic (480M bit/s), over the ULPI/UTMI interface
- The OHCI controller, based on the *Open Host Controller Interface (OHCI) specification for USB Release 1.0a*, is in charge of full-speed/low-speed traffic (12/1.5M bit/s, respectively), over a serial interface

Each of the three external ports is owned by exactly one of the controllers at any time.

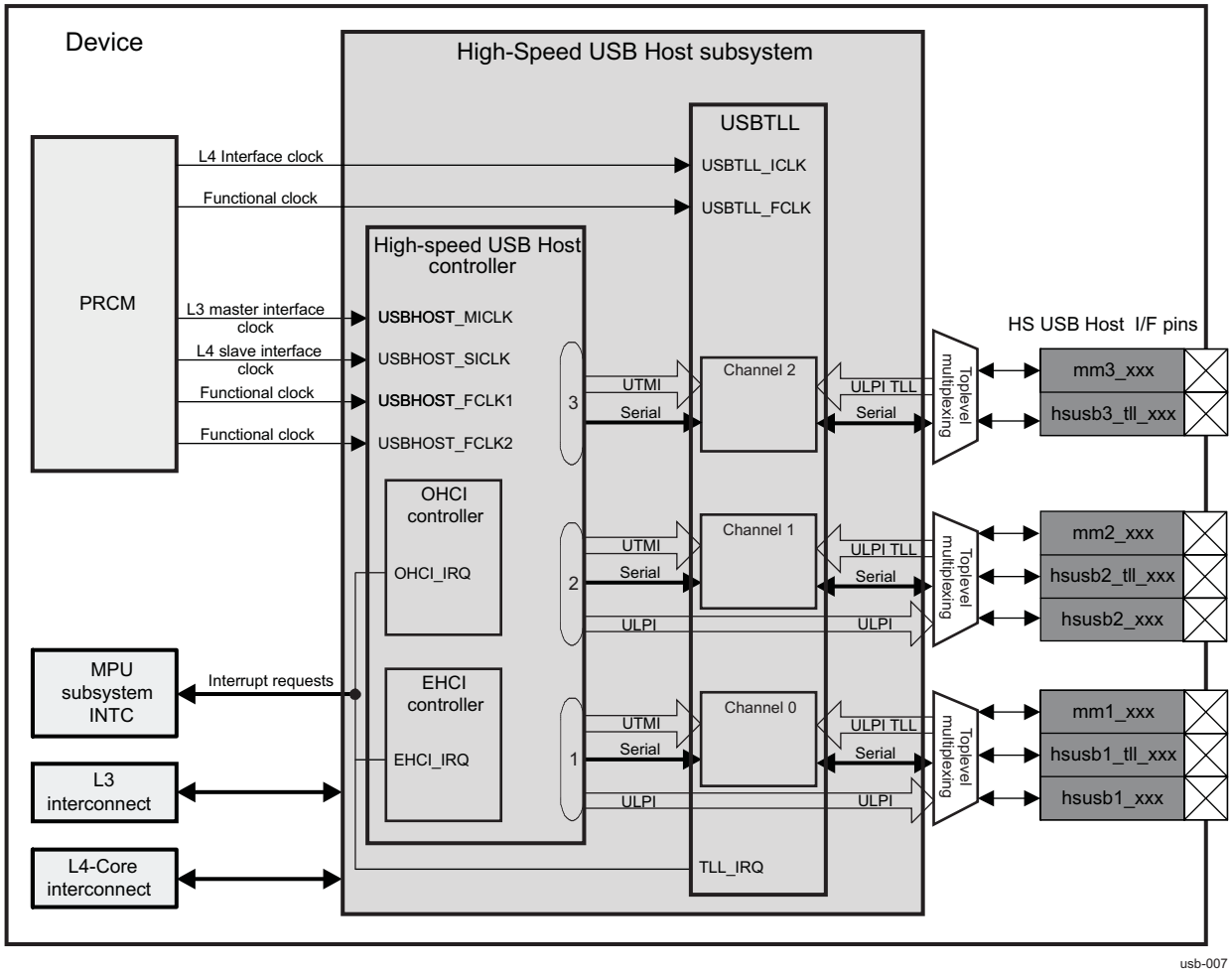
NOTE: If one port is configured as a high-speed ULPI transceiver interface, then all other ports must be likewise configured.

The USBTLL module is a high-speed USB UTMI low-pin interface (ULPI) transceiverless link logic (TLL) adapter. It implements a TLL compatible with a number of USB standard interface protocols. It consists of three channels, defined as independent USB path through the TLL module, which always converts the UTMI+ PHY interface protocol coming from the high-speed USB host controller.

Each USB port (1, 2, and 3) can connect either to an external chip USB transceiver or directly using a transceiverless link to an external integrated circuit (IC) supporting the same TLL protocol.

Figure 20-72 highlights the high-speed USB host subsystem.

Figure 20-72. High-Speed USB Host Subsystem Highlight



20.2.1.1 Main Features

The high-speed USB host subsystem includes the following features:

- Multiport high-speed USB host controller:
  - Complies with the USB 2.0 standard for high-speed (480M bit/s) functions
  - USB 2.0 low-speed (1.5M bit/s) and full-speed (12M bit/s) over serial interface
  - High-speed (480M bit/s) operations over ULPI
  - Three downstream ports (3-port root hub)
  - Complies with EHCI (high-speed host controller)
  - Complies with OHCI (low-speed/full-speed host controller)
  - Supports suspend/resume and remote wakeup
  - Interface with ULPI PHYs (transceivers) on two ports
    - 12-pin/8-bit data single data rate (SDR) mode
    - 60-MHZ clock, generated by the host: ULPI "input" clocking mode

CAUTION

The HS USB host subsystem only supports PHYs that can accept a 60 MHZ input clock.

The HS USB host subsystem can only support the external charge pump of PHY (no support of internal charge pump for ULPI PHY).

  - Hardware-driven save-and-restore of the suspended host hardware context
  - Two interrupt lines
- USBTLL module
  - Three channels
  - Three ports (A, C, and D) by channel
  - Port A: PHY-side UTMI+ port. Connects to the local link controller. The UTMI “local” port is used in all configurations, (that is, the entire channel can be seen as a protocol converted from that port to one of the other, “remote” ports).
    - Compliant with UTMI+ (USB 2.0 Transceiver Macrocell Interface) version 1.0
    - 8-data-bit, 60-MHZ UTMI (HS/FS/LS-capable)
    - UTMI+ Level 3 extensions
    - Vcontrol/Vstatus (from UTMI)
    - Serial FS/LS “6-pin” mode
  - Port C: PHY-side ULPI port. Connects to a remote (off-chip) ULPI link controller through I/O pads.
    - SDR and dual data rate (DDR) ULPI capable (8/4-bit data width modes)
    - Supports optional 6-pin/3-pin serial modes
    - Supports optional input clocking mode
  - Port D: Serial multimode port. Connects to either a serial link controller (TLL modes) or a serial PHY (PHY interface modes).
    - Supports 6-pin unidirectional, 4-pin bidirectional, 3-pin bidirectional, 2-pin bidirectional modes
    - All modes are supported for TLL or PHY interface configuration.
    - Supports sideband signals (pullup/down control, speed/suspend enable, etc)
  - An interrupt line
  - OCP target slave interface (L4) for configuration
- USB port signal pins interface supporting:
  - External USB transceivers
    - ULPI interface: 12-pin/8-bit data SDR version supporting "input" clocking mode
    - Serial 6-pin PHY (transceiver) interfaces: 6-pin mode (TX: DAT/SE0 or TX: DP/DM unidirectional mode), 4-pin mode (DP/DM bidirectional mode), and 3-pin mode (DAT/SE0 bidirectional mode)
  - TLL mode
    - ULPI TLL interfaces: 12-pin/8-bit data SDR and 8-pin/4-bit data DDR versions
    - Serial 6-pin TLL interfaces: 6-pin mode (DAT/SE0 and DP/DM unidirectional modes), 4-pin mode, (DP/DM bidirectional mode), 3-pin mode (DAT/SE0 bidirectional mode) and 2-pin mode (DAT/SE0 and DP/DM bidirectional modes)

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

Table 20-78. USB Connectivity Modes

USB Connectivity Modes	Port 1	Port 2	Port 3
ULPI interface	√	√	
Serial 6-pin transceiver interfaces	√	√	√
ULPI TLL interfaces	√	√	√
Serial 6-pin TLL interfaces	√	√	√

NOTE: If either port 1 or port 2 is configured as a ULPI interface, then both ports must be configured as a ULPI interface.

Only FS/LS USB transceivers can be connected to port 3; HS ULPI transceivers are not supported on port 3.

20.2.2 High-Speed USB Host Subsystem Environment

The high-speed USB host controller provides two kinds of interfaces for connection:

- ULPI interfaces for high-speed data transactions (up to 480M bit/s)
- Serial interfaces (with the use of the USBTLL module) for full- and low-speed data transactions (up to 12M bit/s)

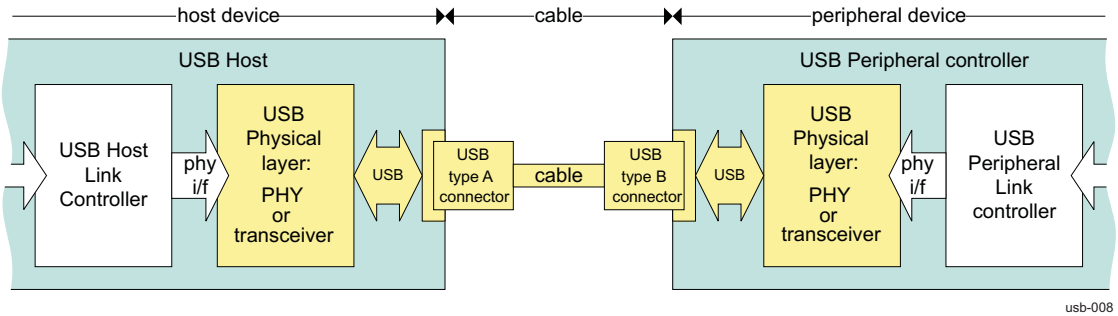
The high-speed USB host controller connects to either controllers (TLL modes) and/or transceivers. It supports the following configurations with the serial interfaces and ULPI interfaces:

- External USB transceiver configurations
  - ULPI interface: 12-pin/8-bit data SDR version supporting "input" clocking mode
  - Serial 6-pin PHY (transceiver) interfaces: 6-pin mode (TX: DAT/SE0 or TX: DP/DM unidirectional mode), 4-pin mode (DP/DM bidirectional mode) and 3-pin mode (DAT/SE0 bidirectional mode)
- TLL configurations
  - ULPI TLL interfaces: 12-pin/8-bit data SDR and 8-pin/4-bit data DDR versions
  - Serial 6-pin TLL interfaces: 6-pin mode (DAT/SE0 and DP/DM unidirectional modes), 4-pin mode (DP/DM bidirectional mode), 3-pin mode (DAT/SE0 bidirectional mode), and 2-pin mode (DAT/SE0 and DP/DM bidirectional modes)

20.2.2.1 Standard USB Implementation: Transceiver Connection

From a logical point of view, a point-to-point USB connection is composed of several blocks, organized in protocol layers, and shown in Figure 20-73.

Figure 20-73. USB Connection



The host system (USB master) and the peripheral system (USB slave) connected through the USB cable include a link or controller (link layer) and a PHY or transceiver (physical layer). Each system talks to its own controller, which talks to its own transceiver, which is connected to the opposite side (transceiver) through an assembly of connectors, receptacles, and cable.

20.2.2.2 TLL Connection

The TLL feature enables connection of the high-speed USB host subsystem to an external, onboard USB peripheral controller, without using USB transceivers or associated circuitry. When TLL is used, the following components are removed from the system:

- Both USB transceivers
- The series resistors
- Pullup and pulldown resistors
- VBUS switching components
- USB connectors and cables, typically used between a USB host controller and the downstream USB peripheral controller

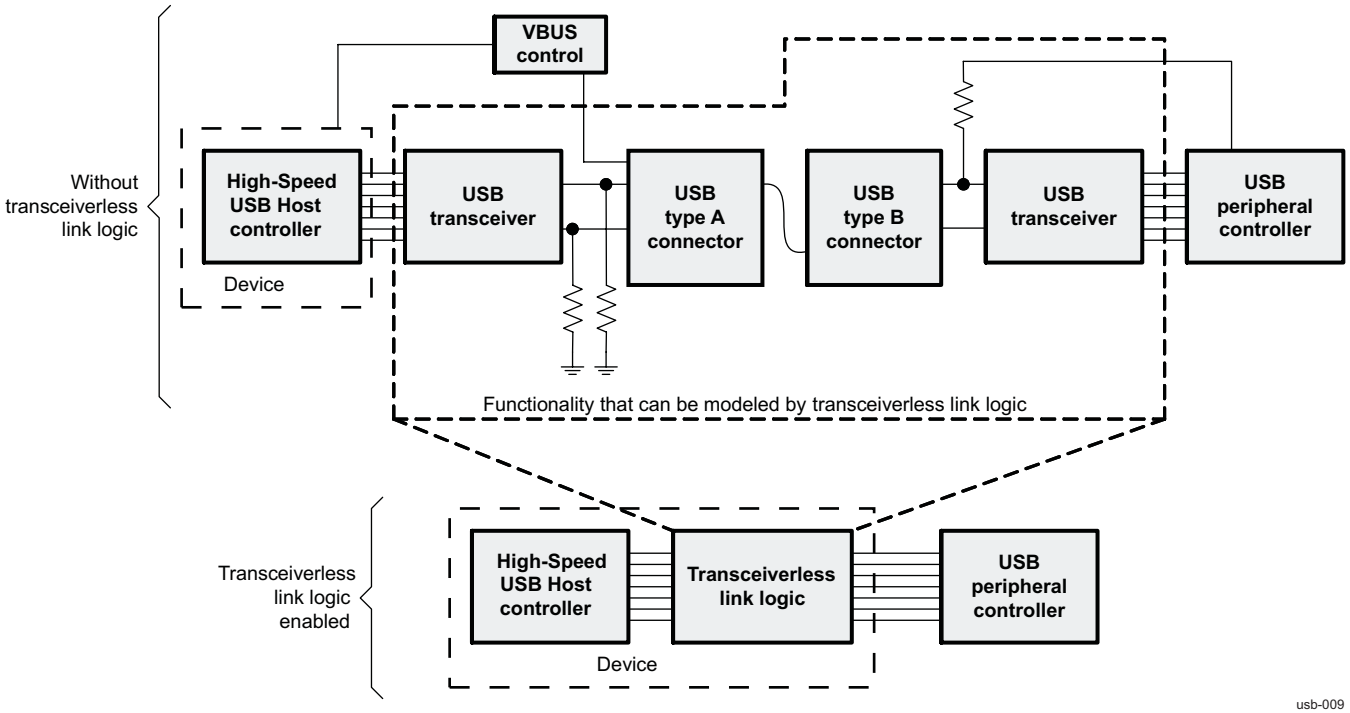
The TLL signaling system is not suitable for use across a cable. It is intended only for use when the device is used with an external USB integrated circuit (IC) that is on the same board.

When using the TLL, signals of the external USB IC pins, which typically connect to a USB transceiver, instead directly connect to the device pins. Signaling on these pins use CMOS levels. TLL logic can be used with external devices that support ULPI TLL interface and serial 6-pin TLL interface connectivity.

The TLL function in the device interprets the transmit control signals from the external USB IC and similar signals from the device USB host controller, and computes the equivalent USB differential-pair state. The computed differential-pair state is interpreted and the appropriate transceiver output signals are provided to the external USB IC and to the device USB host controller.

Figure 20-74 shows the device and how TLL can be compared to a typical USB implementation. The top portion of Figure 20-74 shows the a transceiver-based solution, and the bottom portion shows a transceiverless solution using TLL.

Figure 20-74. High-Speed USB Host Controller Connection—With and Without TLL



NOTE: The USB bus lines (D+/D–) no longer appear in subsequent figures: They are emulated by the TLL.

20.2.2.3 ULPI Interfaces

The high-speed USB host subsystem supports the following configurations with the ULPI interfaces:

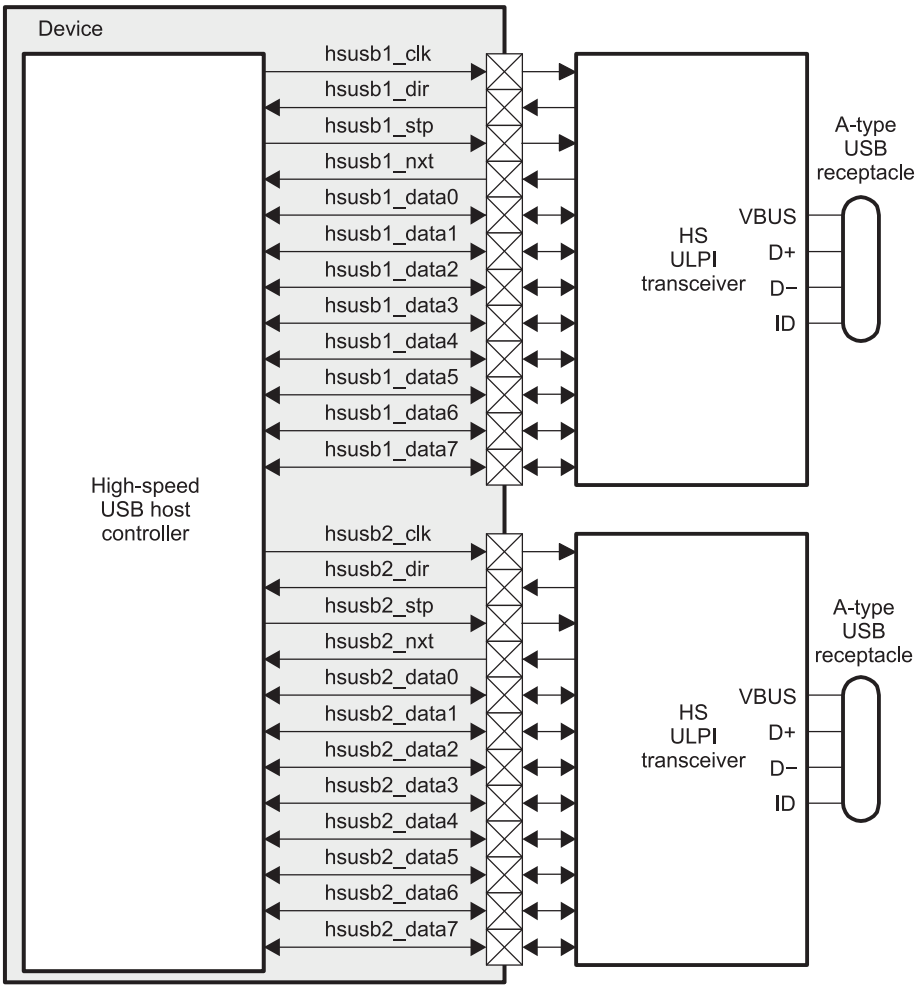
- External USB transceiver
  - ULPI interfaces: 12-pin/8-bit data SDR version supporting "input" clocking mode
- TLL
  - ULPI TLL interfaces: 12-pin/8-bit data SDR and 8-pin/4-bit data DDR versions

Figure 20-75 and Figure 20-76 show typical applications using the high-speed USB host subsystem with the ULPI and with the ULPI TLL, respectively.

The high-speed USB host subsystem supports USB ports, which use the ULPI interface mode to connect to an off-chip high-speed ULPI transceiver (12-pin/8-bit data SDR mode) for high-speed data transactions (up to 480M bit/s). Using the ULPI interfaces in 12-pin/8-bit data version, the device and the transceiver achieve the USB function. An A-type external receptacle allows the connection of an external device.

The device supports TLL logic interfaces on its ports in the ULPI interface mode. TLL modes enable glueless interconnect to another USB device port without a costly transceiver.

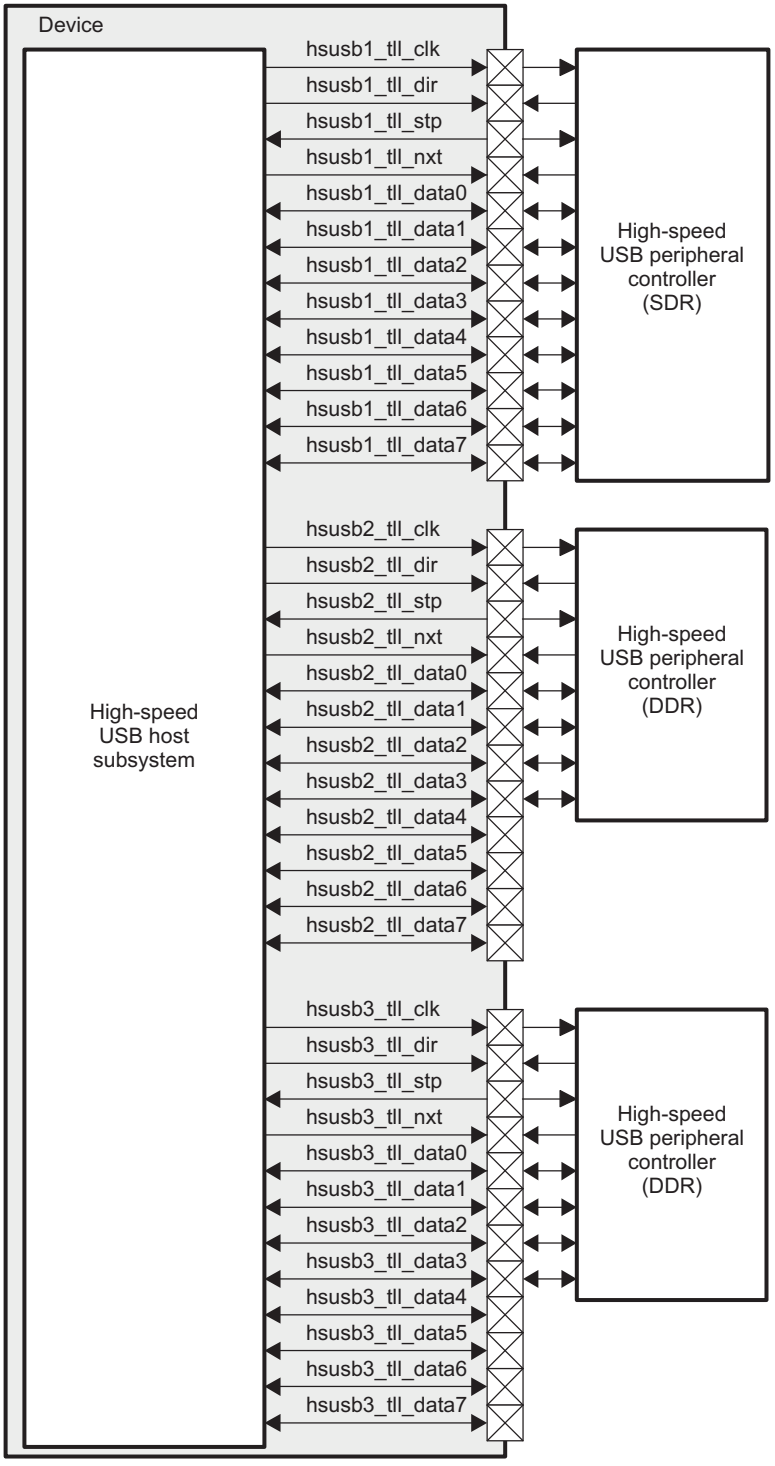
Figure 20-75. High-Speed USB Host Controller Typical Application System – ULPI Interfaces



ULPI Interfaces - USBTLL is bypassed and high-speed USB host controller ports 1 and 2 are connected directly to external transceivers

usb–010

Figure 20-76. High-Speed USB Host Subsystem Typical Application System - ULPI TLL Interfaces



ULPI TLL Interfaces -The high-speed USB host controller is coupled with the USBTLL module to compose the ULPITLL interface modes usb−032

The current implementation of the ULPI includes a method for manual, software-controlled generation of PHY-side register accesses. This implies support of the following features:

- Access to vendor-specific or optional PHY-side registers
- Access to vendor ID, product ID, and scratch and debug registers

The 12-pin ULPI interface uses an 8-bit data bus with data synchronous to the rising edge of the PHY (transceiver) clock (SDR mode), whereas the 8-pin ULPI uses a 4-bit data bus with data generated on both the rising and falling clock edges (DDR mode).

20.2.2.3.1 Transceiver Interface Configurations

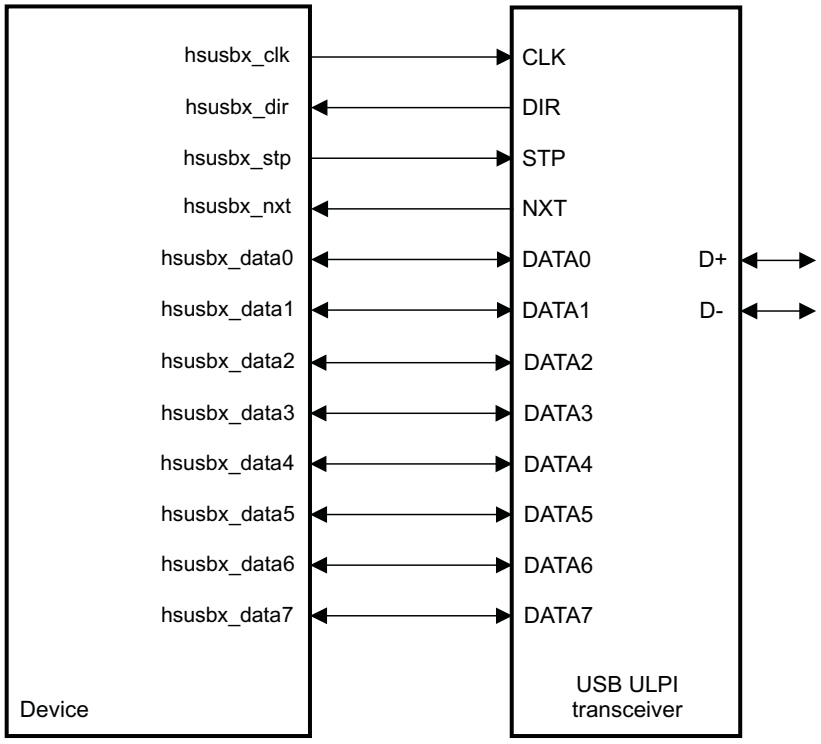
The high-speed USB host subsystem supports only the 12-pin/8-bit data SDR version of the ULPI interface mode.

NOTE: In the device, only the ULPI ports 1 and 2 of the high-speed USB host controller are mapped and can be connected directly to external transceivers.

The ULPI transceiver must support "input" clocking mode, that is, it must accept a 60 MHZ input clock.

Figure 20-77 shows USB ports using the 12-pin/8-bit data SDR version of the ULPI interface mode.

Figure 20-77. ULPI Interfaces – 12-Pin/8-Bit Data SDR Version



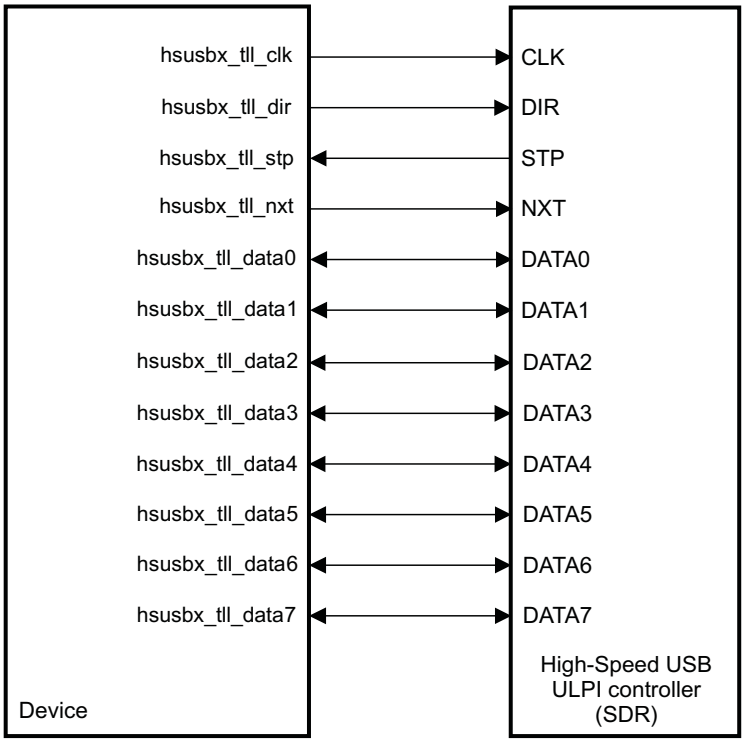
20.2.2.3.2 TLL Configurations

The high-speed USB host controller is coupled with the USBTLL module to compose the ULPI TLL interface modes.

The high-speed USB host subsystem supports the 12-pin/8-bit data SDR and 8-pin/4-bit data DDR versions of the ULPI TLL interface mode.

Figure 20-78 shows USB ports using the 12-pin/8-bit data SDR version of the ULPI TLL interface mode.

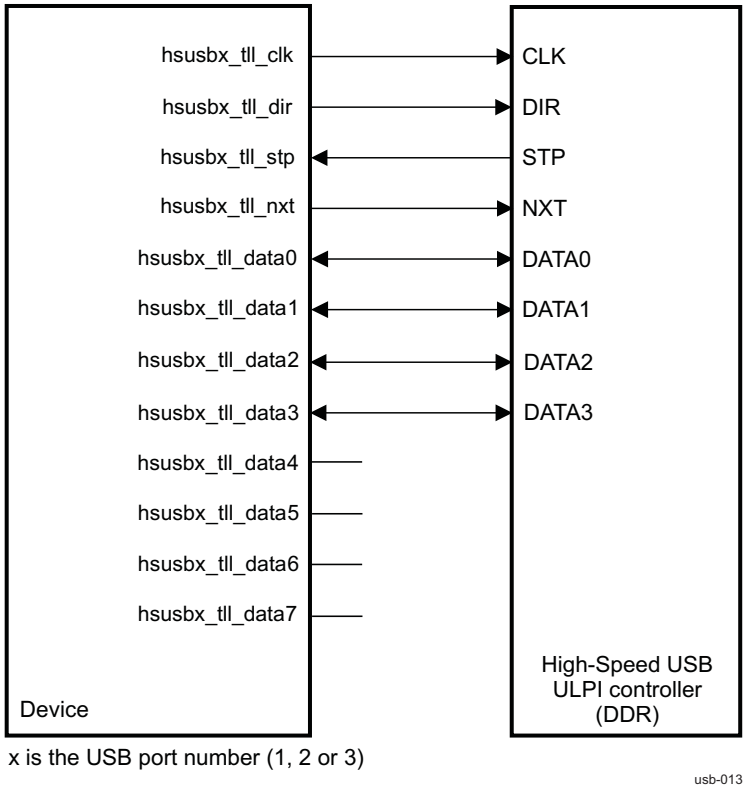
Figure 20-78. ULPI TLL Interfaces –12-Pin/8-Bit Data SDR Version



x is the USB port number (1, 2 or 3)

Figure 20-79 shows USB ports using the 8-pin/4-bit data DDR version of the ULPI TLL interface mode.

Figure 20-79. ULPI TLL Interfaces – 8-Pin/4-Bit Data DDR Version

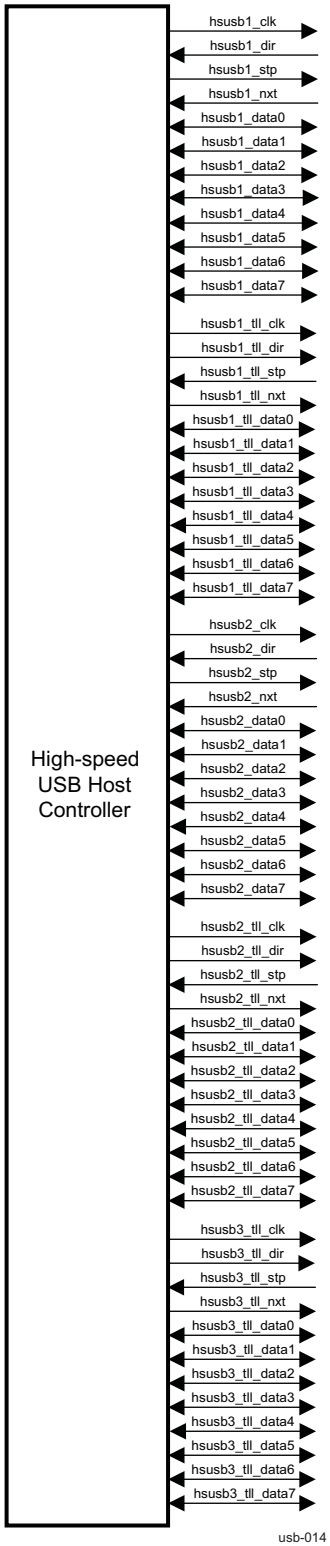


20.2.2.3.3 High-Speed USB Host Subsystem Functional Interfaces

20.2.2.3.3.1 Basic High-Speed USB Host Subsystem Pins

Figure 20-80 shows the high-speed USB host controller ULPI functional interfaces.

Figure 20-80. High-Speed USB Host Subsystem Functional Interface Signals



20.2.2.3.3.2 High-Speed USB Host Subsystem Interface Description

Table 20-79 describes the I/O of the high-speed USB host subsystem ULPI interfaces. The ULPI (PHY) Interfaces and the ULPI TLL Interfaces can not be used together, either the ULPI (PHY) Interfaces or the ULPI TLL Interfaces are selected.

Table 20-79. I/O Description

Signal Name	I/O[1]	Description	Reset Value
HSUSB1			
hsusb1_clk	O	60-MHZ clock output to ULPI transceiver[2]	0
hsusb1_dir	I	Data direction control from ULPI transceiver	Unknown
hsusb1_stp	O	Stop signal to ULPI transceiver	1
hsusb1_nxt	I	Next signal from ULPI transceiver	Unknown
hsusb1_data0	I/O	Bidirectional DATA0	Unknown
hsusb1_data1	I/O	Bidirectional DATA1	Unknown
hsusb1_data2	I/O	Bidirectional DATA2	Unknown
hsusb1_data3	I/O	Bidirectional DATA3	Unknown
hsusb1_data4	I/O	Bidirectional DATA4	Unknown
hsusb1_data5	I/O	Bidirectional DATA5	Unknown
hsusb1_data6	I/O	Bidirectional DATA6	Unknown
hsusb1_data7	I/O	Bidirectional DATA7	Unknown
HSUSB1 TLL			
hsusb1_tll_clk	O	60-MHZ clock output to ULPI transceiver[2]	0
hsusb1_tll_dir	O	Data direction control from ULPI transceiver	0
hsusb1_tll_stp	I	Stop signal to ULPI transceiver	Unknown
hsusb1_tll_nxt	O	Next signal from ULPI transceiver	0
hsusb1_tll_data0	I/O	Bidirectional DATA0	0
hsusb1_tll_data1	I/O	Bidirectional DATA1	0
hsusb1_tll_data2	I/O	Bidirectional DATA2	0
hsusb1_tll_data3	I/O	Bidirectional DATA3	0
hsusb1_tll_data4	I/O	Bidirectional DATA4	0
hsusb1_tll_data5	I/O	Bidirectional DATA5	0
hsusb1_tll_data6	I/O	Bidirectional DATA6	0
hsusb1_tll_data7	I/O	Bidirectional DATA7	0
HSUSB2			
hsusb2_clk	O	60-MHZ clock output to ULPI transceiver[2]	0
hsusb2_dir	I	Data direction control from ULPI transceiver	Unknown
hsusb2_stp	O	Stop signal to ULPI transceiver	1
hsusb2_nxt	I	Next signal from ULPI transceiver	Unknown
hsusb2_data0	I/O	Bidirectional DATA0	Unknown
hsusb2_data1	I/O	Bidirectional DATA1	Unknown
hsusb2_data2	I/O	Bidirectional DATA2	Unknown
hsusb2_data3	I/O	Bidirectional DATA3	Unknown
hsusb2_data4	I/O	Bidirectional DATA4	Unknown
hsusb2_data5	I/O	Bidirectional DATA5	Unknown
hsusb2_data6	I/O	Bidirectional DATA6	Unknown
hsusb2_data7	I/O	Bidirectional DATA7	Unknown

(1) I = Input, O = Output

(2) This output signal is also used as re-timing input.

Table 20-79. I/O Description (continued)

Signal Name	I/O[1]	Description	Reset Value
HSUSB2 TLL			
hsusb2_tll_clk	O	60-MHZ clock output to ULPI transceiver[3]	0
hsusb2_tll_dir	O	Data direction control from ULPI transceiver	0
hsusb2_tll_stp	I	Stop signal to ULPI transceiver	Unknown
hsusb2_tll_nxt	O	Next signal from ULPI transceiver	0
hsusb2_tll_data0	I/O	Bidirectional DATA0	0
hsusb2_tll_data1	I/O	Bidirectional DATA1	0
hsusb2_tll_data2	I/O	Bidirectional DATA2	0
hsusb2_tll_data3	I/O	Bidirectional DATA3	0
hsusb2_tll_data4	I/O	Bidirectional DATA4	0
hsusb2_tll_data5	I/O	Bidirectional DATA5	0
hsusb2_tll_data6	I/O	Bidirectional DATA6	0
hsusb2_tll_data7	I/O	Bidirectional DATA7	0
HSUSB3 TLL			
hsusb3_tll_clk	O	60-MHZ clock output to ULPI transceiver[3]	0
hsusb3_tll_dir	O	Data direction control from ULPI transceiver	0
hsusb3_tll_stp	I	Stop signal to ULPI transceiver	Unknown
hsusb3_tll_nxt	O	Next signal from ULPI transceiver	0
hsusb3_tll_data0	I/O	Bidirectional DATA0	0
hsusb3_tll_data1	I/O	Bidirectional DATA1	0
hsusb3_tll_data2	I/O	Bidirectional DATA2	0
hsusb3_tll_data3	I/O	Bidirectional DATA3	0
hsusb3_tll_data4	I/O	Bidirectional DATA4	0
hsusb3_tll_data5	I/O	Bidirectional DATA5	0
hsusb3_tll_data6	I/O	Bidirectional DATA6	0
hsusb3_tll_data7	I/O	Bidirectional DATA7	0

(3) This output signal is also used as re-timing input.

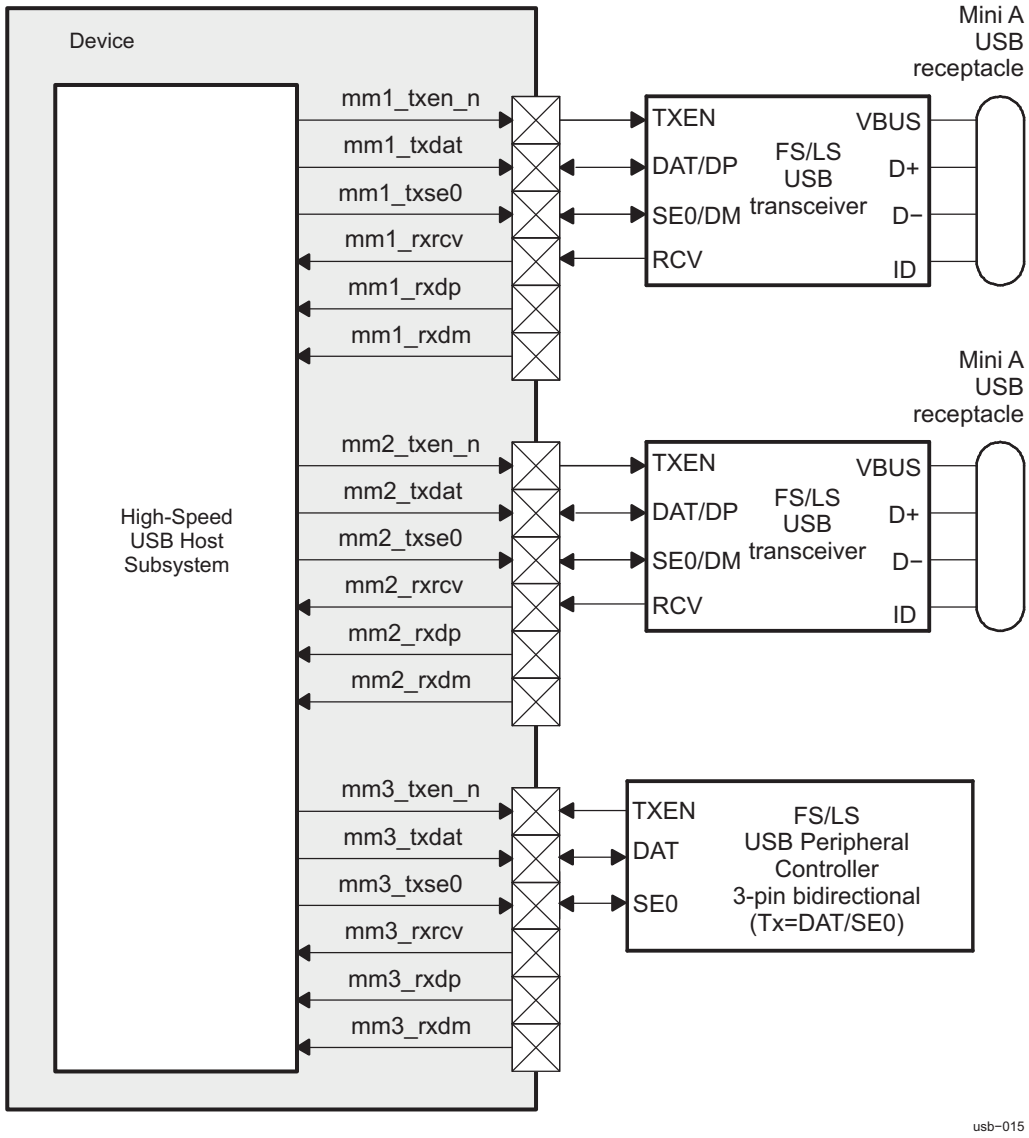
20.2.2.4 Serial Interfaces

The high-speed USB host subsystem supports the following configurations with the serial interfaces:

- External USB transceiver
  - Serial 6-pin PHY (transceiver) interfaces: 6-pin mode (TX: DAT/SE0 or TX: DP/DM unidirectional mode), 4-pin mode (DP/DM bidirectional mode), and 3-pin mode (DAT/SE0 bidirectional mode)
- TLL
  - Serial 6-pin TLL interfaces: 6-pin mode (DAT/SE0 and DP/DM unidirectional modes), 4-pin mode (DP/DM bidirectional mode), 3-pin mode (DAT/SE0 bidirectional mode), and 2-pin mode (DAT/SE0 and DP/DM bidirectional modes)

Figure 20-81 shows a typical application using the high-speed USB host subsystem with the serial interfaces.

Figure 20-81. High-Speed USB Host Subsystem Typical Application System



The high-speed USB host controller is coupled with the USBTLL module to compose the serial interface modes. The USBTLL module translates the parallel, synchronous UTMI+ (Level 3) protocol to a serial, asynchronous one. The benefit of the conversion is a simplified interface to the transceiver.

CAUTION

Only full- and low-speed data transactions are possible in serial mode. Transceiver interface is serial (its frequency is that of the actual USB line) and combinatorial (no clock is passed).

Whether in TLL or transceiver configuration, the serial interface follows the same principles. It is limited to full/low speed, and that high speed requires a parallel interface.

20.2.2.4.1 Encoding in Serial Mode

20.2.2.4.1.1 Unidirectional

When a USB transceiver is connected to the device and used in 6-pin unidirectional DAT/SE0 encoding mode, the encoding described in Table 20-80 is used.

Table 20-80. Signaling Between High-Speed USB Host Subsystem and 6-Pin Unidirectional USB Transceiver (DAT/SE0 Signaling)

Logical Signal Name	Device Pin Direction	Transceiver Pin Direction	Description				
TXEN	Output	Input	When low, the USB transceiver drives D+ and D-.				
DAT and SE0	Output	Input	Controls the values output by the USB transceiver on D+ and D- when TXEN is low; ignored when TXEN is high.				
			TXEN	DAT	SE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
			1	X	X	Undriven	Undriven
RCV	Input	Output	Output from transceiver differential receiver				
			D+	D-	RCV		
			0	0	X		
			0	1	0		
			1	0	1		
			1	1	X		
DP	Input	Output	Output from transceiver single-ended D+ signal receiver				
			D+	DP			
			0	0			
			1	1			
DM	Input	Output	Output from transceiver single-ended D- signal receiver				
			D-	DM			
			0	0			
			1	1			

When a USB transceiver is connected to the device and used in 6-pin unidirectional DP/DM encoding mode, the encoding described in Table 20-81 is used.

Table 20-81. Signaling Between High-Speed USB Host Subsystem and 6-Pin Unidirectional USB Transceiver (DP/DM Signaling)

Logical Signal Name	Device Pin Direction	Transceiver Pin Direction	Description				
TXEN	Output	Input	When low, the USB transceiver drives D+ and D-.				
DAT and SE0	Output	Input	Controls the values output by the USB transceiver on D+ and D- when TXEN is low; ignored when TXEN is high.				
			TXEN	DAT	SE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
			1	X	X	Undriven	Undriven

Table 20-81. Signaling Between High-Speed USB Host Subsystem and 6-Pin Unidirectional USB Transceiver (DP/DM Signaling) (continued)

Logical Signal Name	Device Pin Direction	Transceiver Pin Direction	Description		
RCV	Input	Output	Output from transceiver differential receiver		
			D+	D-	RCV
			0	0	X
			0	1	0
			1	0	1
			1	1	X
DP	Input	Output	Output from transceiver single-ended D+ signal receiver		
			D+	DP	
			0	0	
			1	1	
DM	Input	Output	Output from transceiver single-ended D- signal receiver		
			D-	DM	
			0	0	
			1	1	

20.2.2.4.1.2 Bidirectional

When a USB or USB OTG transceiver is connected to the device and is used in 3-pin bidirectional DAT/SE0 encoding mode, the encoding described in Table 20-82 is used.

Table 20-82. Signaling Between High-Speed USB Host Subsystem and 3-Pin Bidirectional USB Transceiver Using DAT/SE0 Signaling

Logical Signal Name	Device Pin Direction	Transceiver Pin Direction	Description				
TXEN	Output	Input	When low, USB transceiver drives D+ and D-.				
DAT and SE0	Output	Input	When TXEN is low, the device drives DAT and SE0 and the transceiver drives D+ and D- based on the values of DAT and SE0.				
	Output	Input	TXEN	DAT	SE0	D+	D-
			0	0	0	0	1
				1	0	1	0
				X	1	0	0
	Input	Output	TXEN	D+	D-	DAT	SE0
			1	0	0	0	1
				0	1	0	0
				1	0	1	0
				1	1	Undefined	Undefined

NOTE: The device does not support 3-wire bidirectional signaling using DP/DM signals.

When a USB or USB OTG transceiver is connected to the device and is used in 4-pin bidirectional DP/DM encoding mode, the encoding described in Table 20-83 is used.

Table 20-83. Signaling Between High-Speed USB Host Subsystem and 4-Pin Bidirectional USB Transceiver Using DP/DM Signaling

Logical Signal Name	Device Pin Direction	Transceiver Pin Direction	Description		
TXEN	Output	Input	When low, USB transceiver drives D+ and D-.		
DM			Value driven to or received from D-		
	Output	Input	TXEN	DM	D-
			0	0	0
			0	1	1
	Input	Output	TXEN	D-	DM
			1	0	0
			1	1	1
DP			Value driven to or received from D+		
	Output	Input	TXEN	DP	D+
			0	0	0
			0	1	1
	Input	Output	TXEN	D+	DP
			1	0	0
			1	1	1
RCV	Input	Output	Output from transceiver single-ended D- signal receiver		
			D+	D-	RCV
			0	0	X
			0	1	0
			1	0	1
			1	1	X

NOTE: The device does not support 4-pin bidirectional signaling using DAT/SE0 signals.

20.2.2.4.2 Sideband Signals for Serial Modes

Serial interfaces only carry the USB data information. Sideband control and status (respectively, to/from the transceiver/TLL or to the bus lines themselves) require additional signals, which are usually implemented in a case-by-case, ad hoc way.

- Sideband control examples: FS/LS (slew rate control), transceiver suspend, connect (D+/D– pullup), pulldown enable, VBUS drive, etc.
- Sideband status example: VBUS level (VBUS valid, session valid, session end), etc.
- Sideband signal implementations: dedicated lines (one per sideband information bit), serial bus + interrupt line with register-mapped control/status (I2C, UART, etc.)

Figure 20-82 and Figure 20-83 show system integration for sideband signals for two logically identical USB connections: one in transceiver configuration, and one in TLL configuration. Although the sideband (purple) arrows are all oriented from controller to transceiver in the two figures, the sideband information flow is bidirectional (that is, it flows from controller to transceiver [control] but also from transceiver to controller [status]).

Figure 20-82 shows the transceiver configuration, where each side connects the sideband signals to its own transceiver. On the device (containing the USBTLL module), the sideband is decoded/re-encoded. The sideband signals available at the device boundary (and the USBTLL module) are decoded from the standard UTMI+ interface.

- Sideband output signals from the USBTLL module (speed, suspend, puen, etc.)
- The software-driven VBUS reporting procedure is described in Section 20.2.4.2.4.1.1.

Figure 20-82. Serial Interface Sideband Integration - Transceiver Configuration

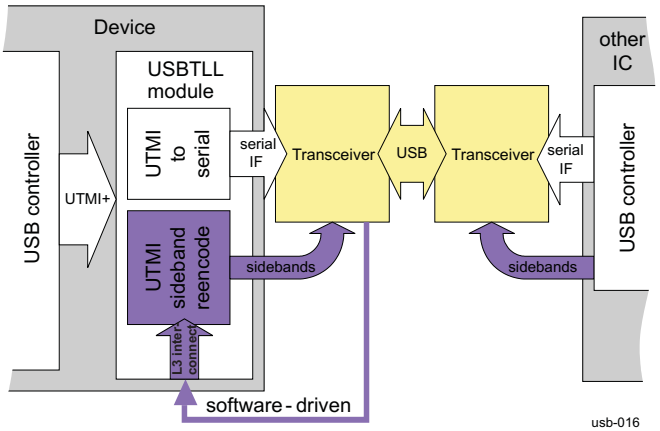
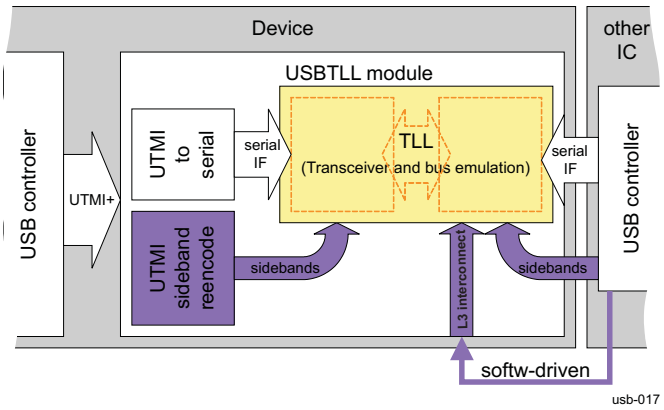


Figure 20-83 shows the TLL configuration, where both transceivers are actually emulated inside the USBTLL module.

- The transceiver of the local controller (left) is working with the sideband information to/from the UTMI+ port. This is internal to the USBTLL module.
- The transceiver of the remote controller (right) must communicate with its controller, located on another IC. This is done in two ways:
  - Sideband input signals at the TLL module boundary (tllpuen, tlldrvvbus, tllvbusvalid, etc.)
  - The software-driven VBUS control procedure is described in Section 20.2.4.2.4.2.2

Figure 20-83. Serial Interface Sideband Integration - TLL Configuration



20.2.2.4.3 Transceiver Interface Configurations

An external USB transceiver is required for each USB port used in the system. It converts between appropriate signaling for the high-speed USB host subsystem and appropriate signaling for the USB wire.

The serial interface mode of the high-speed USB host subsystem includes support for several types of USB transceivers. It provides signaling to up to three external USB transceivers.

Several types of external transceiver signaling are supported. Signaling between the high-speed USB subsystem in the serial-interface mode and the external USB transceiver for monitoring and controlling the differential USB signal can be done through a 6-, 4-, or 3-wire signaling interface, with two or more control signals provided either by additional signals or through an I^2C link.

The following subsections describe the transceiver interface modes supported by the high-speed USB host subsystem in the serial interface mode. In each case, the subsystem is connected to external transceivers, on the other side of which are the actual USB lines (D+/D-).

20.2.2.4.3.1 Unidirectional Transceiver Interface Modes: 6-Pin

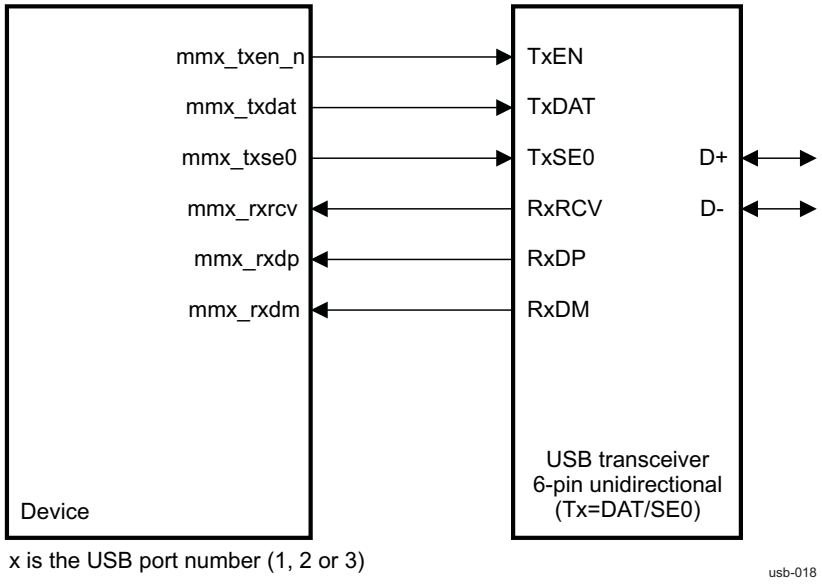
The 6-pin modes are the "natural" transceiver interface modes for the full-speed transceivers in the sense that they mirror the internal makeup of the transceivers.

Two encodings exist for TX: DAT/SE0 or DP/DM.

When a USB is connected to the device and used in 6-pin unidirectional DAT/SE0 signaling mode, the signaling described in Table 20-80 is used.

Figure 20-84 shows a USB port using DAT/SE0 encoding.

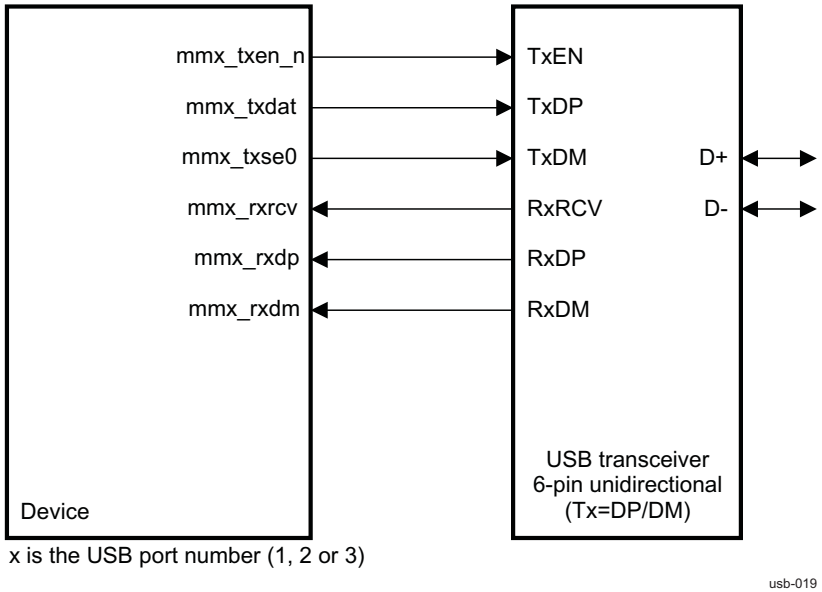
Figure 20-84. 6-Pin Unidirectional Using DAT/SE0 Signaling



When a USB is connected to the device and used in 6-pin unidirectional DP/DM signaling mode, the signaling described in Table 20-81 is used.

Figure 20-85 shows a USB port using DP/DM encoding.

Figure 20-85. 6-Pin Unidirectional Using DP/DM Signaling



20.2.2.4.3.2 Bidirectional Transceiver Interface Modes: 3-Pin, 4-Pin

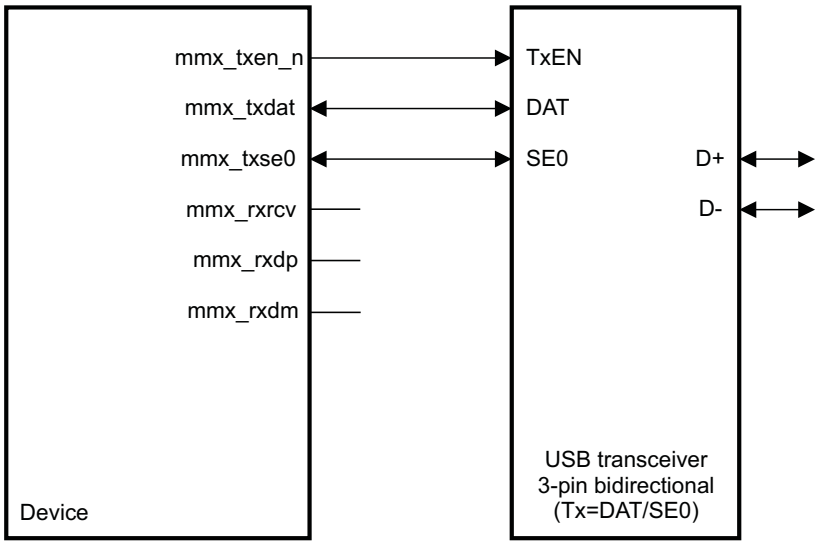
The bidirectional transceiver interface modes are pin-count optimizations of the unidirectional modes. They take advantage of the fact that a USB port is either sending or receiving at any given time, but never both. The TX and RX paths of the unidirectional mode can be multiplexed on bidirectional lines. To prevent glitches at TX/RX turnaround, the same encoding is used for both directions (DAT/SE0 or DP/DM).

The signaling listed in Table 20-82 is used when a USB transceiver is connected to the device and is used in 3-pin bidirectional DAT/SE0 signaling mode.

NOTE: The device does not support 3-wire bidirectional signaling using DP/DM signals.

Figure 20-86 shows a USB port using DAT/SE0 encoding.

Figure 20-86. 3-Pin Bidirectional Using DAT/SE0 Signaling

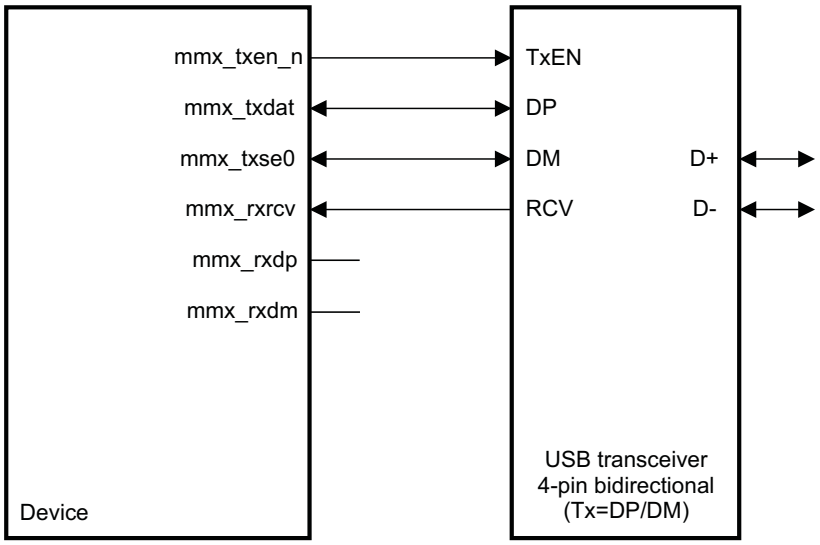


The signaling listed in Table 20-83 is used when a USB transceiver is connected to the device and is used in 4-pin bidirectional DP/DM signaling mode.

NOTE: The device does not support 4-pin bidirectional signaling using DAT/SE0 signals.

Figure 20-87 shows a USB port using DP/DM encoding.

Figure 20-87. 4-Pin Bidirectional Using DP/DM Signaling



20.2.2.4.4 TLL Configurations

The high-speed USB host subsystem supports unidirectional and bidirectional TLL logic interfaces on its ports. The TLL modes enable glueless interconnect to the USB device port of another device without needing a costly transceiver.

Serial interface modes are full- or low-speed only. Transceiver interface is serial (its frequency is that of the actual USB line) and combinatorial (no clock is passed).

20.2.2.4.4.1 Unidirectional TLL Modes

The 6-pin TLL configurations are mirror images of the 6-pin transceiver configurations presented above. The same signals are mapped on the same physical pins, but in the opposite directions.

Two possible modes exist, depending on the TX data encoding used by the external device.

Figure 20-88 shows an external device using DAT/SE0 encoding.

Figure 20-88. 6-Pin Unidirectional TLL Using DAT/SE0 Signaling

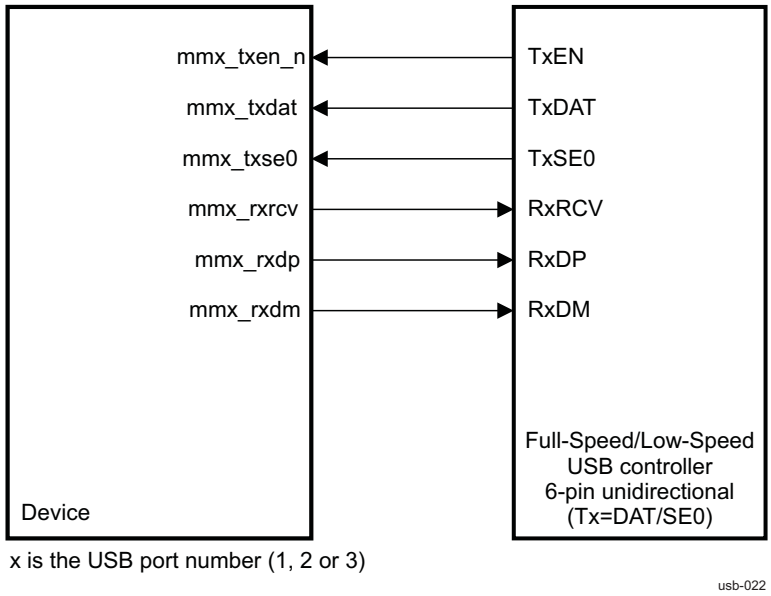
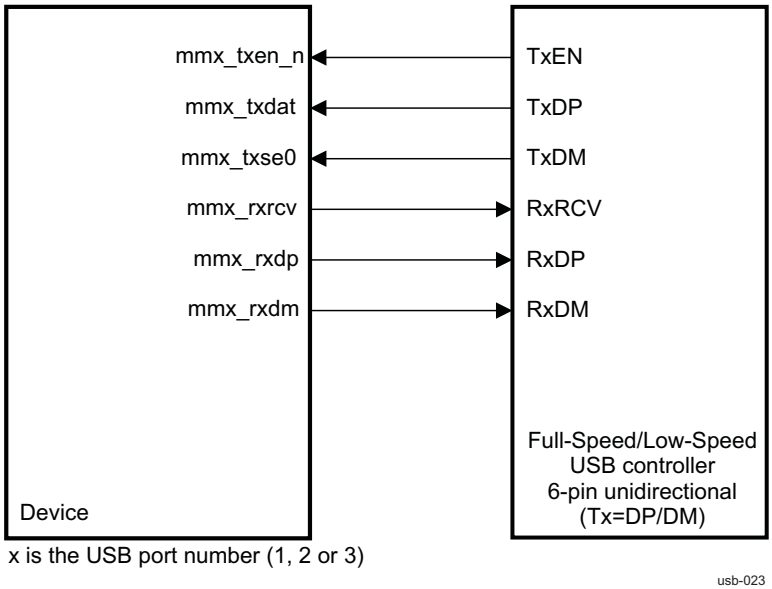


Figure 20-89 shows an external device using DP/DM encoding.

Figure 20-89. 6-Pin Unidirectional TLL Using DP/DM Signaling



20.2.2.4.4.2 Bidirectional TLL Modes

The 3-pin/4-pin TLL configurations are mirror images of the 3-pin/4-pin transceiver configurations presented above. The same signals are mapped on the same physical pins, but in the opposite directions (bidirectional lines remain bidirectional).

Two possible modes exist, depending on the TX data encoding used by the external device.

Figure 20-90 shows an external device using DAT/SE0 encoding.

Figure 20-90. 3-Pin Bidirectional TLL Using DAT/SE0 Signaling

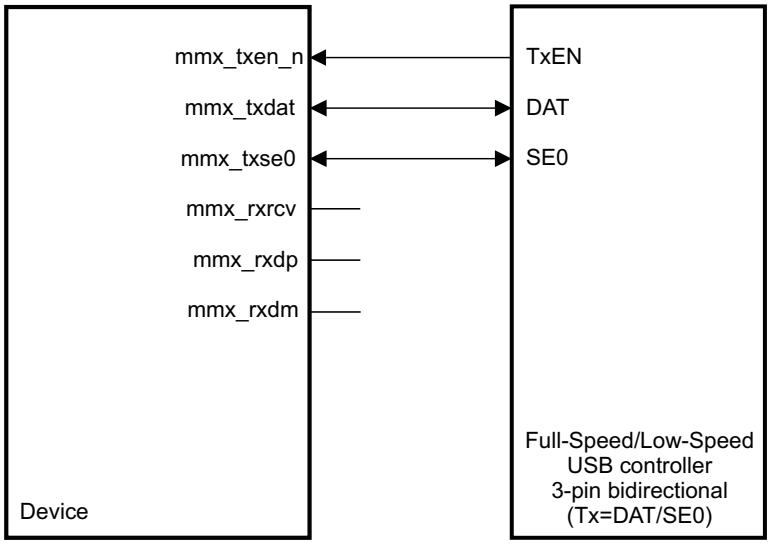
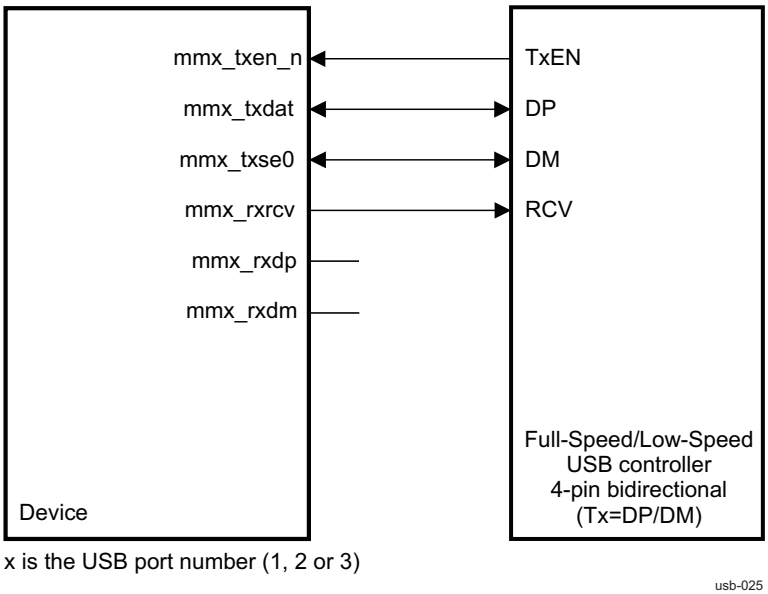


Figure 20-91 shows an external device using DP/DM encoding.

Figure 20-91. 4-Pin Bidirectional TLL Using DP/DM Signaling



The 2-pin TLL configurations have unique specifications:

- They require pullups/pulldowns to operate, because the bidirectional lines are not driven at all times like the other serial transceiver interfaces described above. The connection of pull resistors depends on the speed of the controller.
- The module supports explicit 2-pin TLL modes, with either DAT/SE0 or DP/DM encoding.
- Non-TLL modes (that is, transceiver configuration mode) can be used to implement the 2-pin functionality, using a specific connectivity.

Figure 20-92 shows USB port using DP/DM encoding.

Figure 20-92. 2-Pin Bidirectional TLL Using DP/DM Encoding, With 4-Pin Bidirectional USB Device

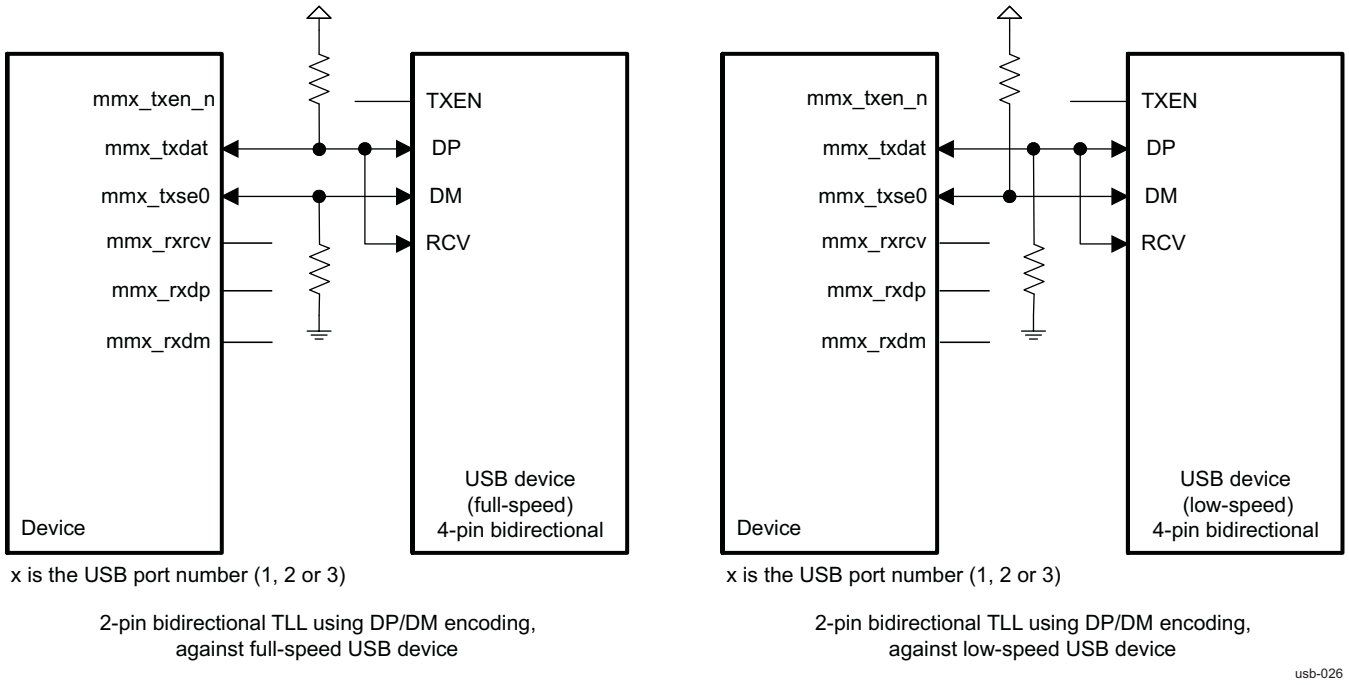


Table 20-84 shows the pullup/pulldown configuration for DP/DM encoding.

Table 20-84. Pullup/Pulldown Configuration for DP/DM Encoding

	Nonconnected Device (Any Speed)	Connected Low-Speed Device	Connected Full-Speed Device
DP	Pulldown	Pulldown	Pullup
DM	Pulldown	Pullup	Pulldown

Figure 20-93 shows a USB port using DAT/SE0 encoding.

Figure 20-93. 2-Pin Bidirectional TLL Using DAT/SE0 Encoding, With 3-Pin Bidirectional USB Device

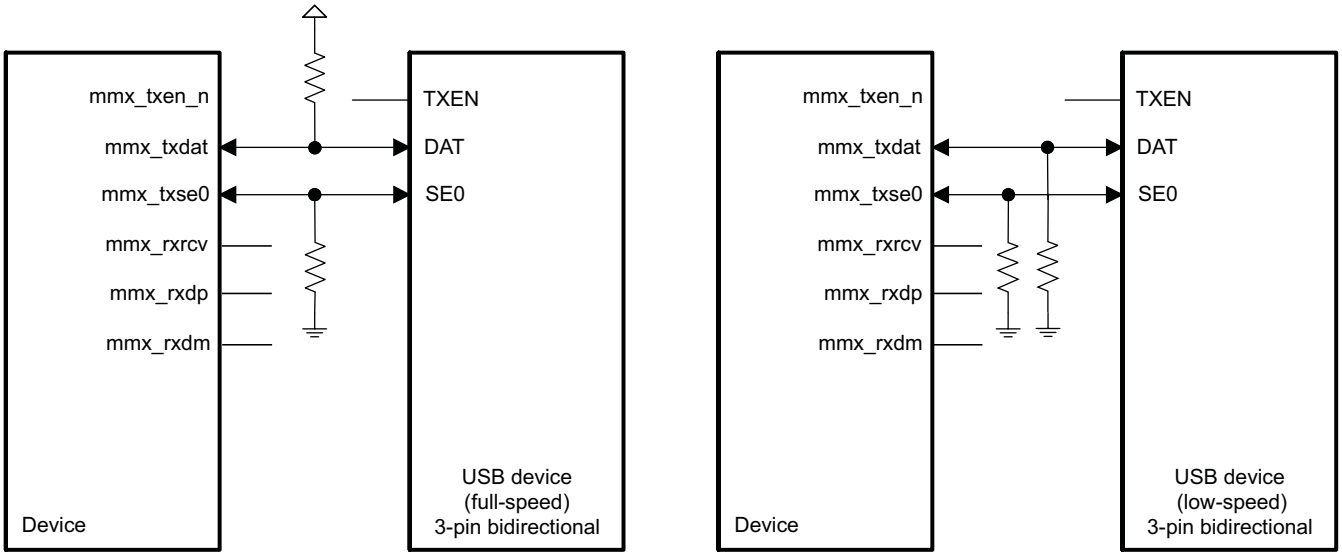


Table 20-85 shows pullup/pulldown configuration for DAT/SE0 encoding.

Table 20-85. Pullup/Pulldown Configuration for DAT/SE0 Encoding

	Nonconnected Device (Any Speed)	Connected Low-Speed Device	Connected Full-Speed Device
DAT	Pulldown	Pulldown	Pullup
SEO	Pullup	Pulldown	Pulldown

20.2.2.4.5 High-Speed USB Host Subsystem Interface Description

Table 20-86 describes the I/O of the high-speed USB host subsystem serial interfaces.

Table 20-86. I/O Description

Signal Name	I/O[1]	Description	Value at Reset
Multiple-Mode FS/LS Serial Interface: Port 1			
mm1_txse0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	0
	I/O	DM function in 4-pin bidirectional DP/DM mode	
	O	SE0 output in 6-pin unidirectional DAT/SE0 mode	
	O	DM output in 6-pin unidirectional DP/DM mode	
	I/O	SE0-TLL in 2-/3-pin bidirectional DAT/SE0 TLL mode	
	I/O	DM-TLL in 2-/4-pin bidirectional DP/DM TLL mode	
	I	SE0-TLL input in 6-pin unidirectional DAT/SE0 TLL mode	
	I	DM-TLL input in 6-pin unidirectional DP/DM TLL mode	

(1) I = Input, O = Output

Table 20-86. I/O Description (continued)

Signal Name	I/O[1]	Description	Value at Reset
mm1_txdat	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	Unknown
	I/O	DP function in 4-pin bidirectional DP/DM mode	
	O	DAT output in 6-pin unidirectional DAT/SE0 mode	
	O	DP output in 6-pin unidirectional DAT/SE0 mode	
	I/O	DAT-TLL in 2-/3-pin bidirectional DAT/SE0 TLL mode	
	I/O	DP-TLL in2-/4-pin bidirectional DP/DM TLL mode	
	I	DAT-TLL input in 6-pin unidirectional DAT/SE0 TLL mode	
	I	DP-TLL input in 6-pin unidirectional DP/DM TLL mode	
mm1_txen_n	O	Transmit enable in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM or 6-pin unidirectional modes	1
	I	Transmit enable in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 6-pin unidirectional TLL modes (not used in the 2-pin bidirectional TLL modes)	
mm1_rxrcv	I	Differential receiver signal input in the 4-pin bidirectional DP/DM or 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 mode)	Unknown
	O	Differential receiver signal output in the 4-pin bidirectional DP/DM TLL or 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 2-pin bidirectional TLL modes)	
mm1_rxdp	I	Single-ended DP receiver signal input in 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM modes)	Unknown
	O	Single-ended DP receiver signal output in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 2-pin bidirectional TLL modes)	
mm1_rxdm	I	Single-ended DM receiver signal input in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM or 2-pin bidirectional TLL modes)	Unknown
	O	Single-ended DM receiver signal output in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 2-pin bidirectional TLL modes)	
Multiple-mode FS/LS serial interface: port 2			
mm2_txse0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	0
	I/O	DM function in 4-pin bidirectional DP/DM mode	
	O	SE0 output in 6-pin unidirectional DAT/SE0 mode	
	O	DM output in 6-pin unidirectional DP/DM mode	
	I/O	SE0-TLL in 2-/3-pin bidirectional DAT/SE0 TLL mode	
	I/O	DM-TLL in 2-/4-pin bidirectional DP/DM TLL mode	
	I	SE0-TLL input in 6-pin unidirectional DAT/SE0 TLL mode	
	I	DM-TLL input in 6-pin unidirectional DP/DM TLL mode	
mm2_txdat	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	Unknown
	I/O	DP function in 4-pin bidirectional DP/DM mode	
	O	DAT output in 6-pin unidirectional DAT/SE0 mode	
	O	DP output in 6-pin unidirectional DAT/SE0 mode	
	I/O	DAT-TLL in 2-/3-pin bidirectional DAT/SE0 TLL mode	
	I/O	DP-TLL in2-/4-pin bidirectional DP/DM TLL mode	
	I	DAT-TLL input in 6-pin unidirectional DAT/SE0 TLL mode	
	I	DP-TLL input in 6-pin unidirectional DP/DM TLL mode	
mm2_txen_n	O	Transmit enable in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM or 6-pin unidirectional modes	1
	I	Transmit enable in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 6-pin unidirectional TLL modes (not used in the 2-pin bidirectional TLL modes)	
mm2_rxrcv	I	Differential receiver signal input in the 4-pin bidirectional DP/DM or 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 mode)	Unknown

Table 20-86. I/O Description (continued)

Signal Name	I/O[1]	Description	Value at Reset
	O	Differential receiver signal output in the 4-pin bidirectional DP/DM TLL or 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 2-pin bidirectional TLL modes)	
mm2_rxdp	I	Single-ended DP receiver signal input in 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM modes)	Unknown
	O	Single-ended DP receiver signal output in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 2-pin bidirectional TLL modes)	
mn2_rxdm	I	Single-ended DM receiver signal input in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM or 2-pin bidirectional TLL modes)	Unknown
	O	Single-ended DM receiver signal output in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 2-pin bidirectional TLL modes)	
Multiple-mode FS/LS serial interface: port 3			
mm3_txse0	I/O	SE0 function in 3-pin bidirectional DAT/SE0 mode	0
	I/O	DM function in 4-pin bidirectional DP/DM mode	
	O	SE0 output in 6-pin unidirectional DAT/SE0 mode	
	O	DM output in 6-pin unidirectional DP/DM mode	
	I/O	SE0-TLL in 2-/3-pin bidirectional DAT/SE0 TLL mode	
	I/O	DM-TLL in 2-/4-pin bidirectional DP/DM TLL mode	
	I	SE0-TLL input in 6-pin unidirectional DAT/SE0 TLL mode	
	I	DM-TLL input in 6-pin unidirectional DP/DM TLL mode	
mm3_txdat	I/O	DAT function in 3-pin bidirectional DAT/SE0 mode	Unknown
	I/O	DP function in 4-pin bidirectional DP/DM mode	
	O	DAT output in 6-pin unidirectional DAT/SE0 mode	
	O	DP output in 6-pin unidirectional DAT/SE0 mode	
	I/O	DAT-TLL in 2-/3-pin bidirectional DAT/SE0 TLL mode	
	I/O	DP-TLL in2-/4-pin bidirectional DP/DM TLL mode	
	I	DAT-TLL input in 6-pin unidirectional DAT/SE0 TLL mode	
	I	DP-TLL input in 6-pin unidirectional DP/DM TLL mode	
mm3_txen_n	O	Transmit enable in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM or 6-pin unidirectional modes	1
	I	Transmit enable in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 6-pin unidirectional TLL modes (not used in the 2-pin bidirectional TLL modes)	
mm3_rxrcv	I	Differential receiver signal input in the 4-pin bidirectional DP/DM or 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 mode)	Unknown
	O	Differential receiver signal output in the 4-pin bidirectional DP/DM TLL or 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 2-pin bidirectional TLL modes)	
mm3_rxdp	I	Single-ended DP receiver signal input in 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM modes)	Unknown
	O	Single-ended DP receiver signal output in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 2-pin bidirectional TLL modes)	
mm3_rxdm	I	Single-ended DM receiver signal input in 6-pin unidirectional modes (not used in the 3-pin bidirectional DAT/SE0 or 4-pin bidirectional DP/DM modes)	Unknown
	O	Single-ended DM receiver signal output in 6-pin unidirectional TLL modes (not used in the 3-pin bidirectional DAT/SE0 TLL or 4-pin bidirectional DP/DM TLL or 2-pin bidirectional TLL modes)	

20.2.3 High-Speed USB Host Subsystem Integration

This section describes the integration of the high-speed USB host subsystem.

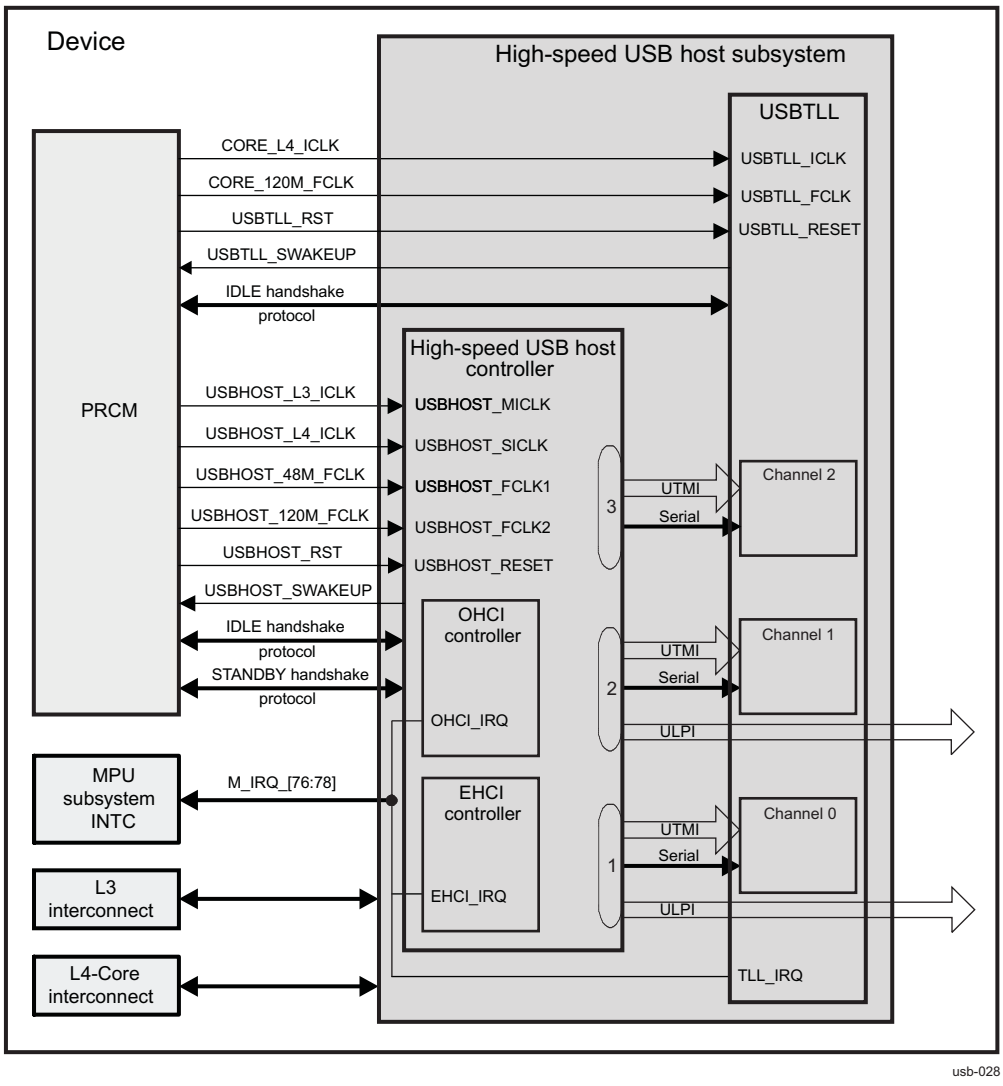
The high-speed USB host controller is connected to the L3 interconnect master (initiator) and L4-Core interconnect slave (target) interfaces. The USBTLL module is connected to the L4-Core interconnect slave (target) interface. The L3 interconnect is used to generate data traffic within the device. The L4-Core interconnect is a configuration port for register setting.

Three interrupts, M_IRQ_[78:76], are connected to the system.

The system control module (SCM) offers complementary settings for USB port connectivity modes.

Figure 20-94 highlights the high-speed USB host subsystem integration in the device.

Figure 20-94. High-Speed USB Subsystem Integration



20.2.3.1 Reset, Clocking, and Power-Management Scheme

The high-speed USB host controller belongs to the USBHOST power domain. As part of the USBHOST power domain, it is sensible to the USBHOST_RST reset signal issued by the PRCM and to USBHOST power domain-related transitions. For further details about USBHOST power domain implementation and USBHOST_RST signal, see the *Power, Reset, and Clock Management* chapter.

The USBTLL module belongs to the CORE power domain. As part of the CORE power domain, it is sensible to the asynchronous USBTLL_RST reset signal issued by the PRCM and to CORE power domain-related transitions. For further details about CORE power domain implementation and USBTLL_RST signal, see the *Power, Reset, and Clock Management* chapter.

20.2.3.1.1 High-Speed USB Host Subsystem Resets

Table 20-87 lists and describes all the high-speed USB host subsystem resets:

Table 20-87. High-Speed USB Host Subsystem Reset Description

Type	Name	Source	Polarity	Description
Software	USBHOST.UHH_SYSCONFIG[1] SOFTRESET bit	High-speed USB host controller internal software reset	Active high	Writing 1 to the SOFTRESET bit resets the module. The bit value of 1 remains until the reset is complete. When the software reset is complete, the SOFTRESET bit is automatically reset to 0 and has the same effect as the hardware reset.
Software	USBHOST.USBTLL_SYSCONFIG[1] SOFTRESET bit	USBTLL module internal software reset	Active high	Writing 1 to the SOFTRESET bit resets the module. The bit value of 1 remains until the reset is complete. When the software reset is complete, the SOFTRESET bit is automatically reset to 0 and has the same effect as the hardware reset.
Hardware	USBHOST_RESET	PRCM USBHOST_RST signal	Active low	The USBHOST_RST signal resets the module. The hardware reset signal has a global reset action on the high-speed USB host controller (see the *Power, Reset, and Clock Management* chapter).
Hardware	USBTLL_RESET	PRCM USBTLL_RST signal	Active low	The USBTLL_RST signal resets asynchronously the module. The hardware reset signal has a global reset action on the USBTLL module (see the *Power, Reset, and Clock Management* chapter).

20.2.3.1.1.1 Hardware Resets

The high-speed USB host controller is attached to the USBHOST power domain: the USBHOST_RST signal resets the module. The USBTLL module is attached to the CORE power domain: the USBTLL_RST signal resets asynchronously the module (see the *Power, Reset, and Clock Management* chapter). The hardware reset signal has a global reset action on the modules.

20.2.3.1.1.2 Software Resets

The high-speed USB host controller and the USBTLL module have their own software-reset functionality through the USBHOST.UHH_SYSCONFIG[1] SOFTRESET bit (0: normal mode; 1: module is reset) for the high-speed USB host controller, and through the USBHOST.USBTLL_SYSCONFIG[1] SOFTRESET bit (0: normal mode; 1: module is reset) for the USBTLL module.

Writing 1 to the SOFTRESET bit resets the module. The bit value of 1 remains until the reset is complete. When the software reset is complete, the SOFTRESET bit is automatically reset to 0 and has the same effect as the hardware reset.

20.2.3.1.2 High-Speed USB Host Subsystem Clocks

The high-speed USB host controller operates from four clock domains:

- USBHOST_FCLK1 is a high-speed USB host controller functional clock. It is used to clock the OHCI and EHCI controller internal logic of the high-speed USB host controller. Its source is the PRCM USBHOST_48M_FCLK output clock. USBHOST_FCLK1 is controlled by the PRCM.PRCM.CM_FCLKEN_USBHOST[0] EN_USBHOST1 bit (0: disabled; 1: enabled).
- USBHOST_FCLK2 is a high-speed USB host controller functional clock. It is used to clock EHCI controller internal logic of the high-speed USB host controller. Its source is the PRCM USBHOST_120M_FCLK output clock. USBHOST_FCLK2 is controlled by the PRCM.PRCM.CM_FCLKEN_USBHOST[1] EN_USBHOST2 bit (0: disabled; 1: enabled).
- USBHOST_MICLK is the high-speed USB host controller L3 master interface clock. It is used to synchronize the high-speed USB host controller L3 port to L3 interconnect. All accesses from the interconnect are synchronous to USBHOST_MICLK. Its source is the PRCM USBHOST_L3_ICLK output clock. USBHOST_MICLK is controlled by the PRCM.CM_ICLKEN_USBHOST[0] EN_USBHOST bit (0: disabled; 1: enabled) and the PRCM.CM_AUTOIDLE_USBHOST[0] AUTO_USBHOST bit (enables/disables automatic control of the interface clock).
- USBHOST_SICLK is a high-speed USB host controller L4 slave interface clock. It is used to synchronize the high-speed USB host controller L4 port to L4 interconnect. All accesses from the interconnect are synchronous to USBHOST_SICLK. Its source is the PRCM USBHOST_L4_ICLK output clock.

The USBTLL module operates from two clock domains:

- USBTLL_FCLK is the USBTLL module functional clock. It is used to clock the USBTLL module internal logic. Its source is the PRCM CORE_120M_FCLK output clock. UBTLL_FCLK is controlled by the PRCM.CM_FCLKEN3_CORE[2] EN_USBTLL bit (0: disabled; 1: enabled).
- USBTLL_ICLK is the USBTLL module interface clock. It is used to synchronize USBTLL module L4 port to L4 interconnect. All accesses from the interconnect are synchronous to USBTLL_ICLK. Its source is the PRCM CORE_L4_ICLK output clock. USBTLL_ICLK is controlled by the PRCM.CM_ICLKEN3_CORE[2] EN_USBTLL bit (0: disabled; 1: enabled) and the PRCM.CM_AUTOIDLE3_CORE[2] AUTO_USBTLL bit (enables/disables automatic control of the interface clock).

Table 20-88 summarizes the high-speed USB host subsystem clocks.

Table 20-88. High-Speed USB Host Subsystem Clocks

Attributes	Frequency	Name	Module	Mapping	Comments
Functional clock	48 MHZ	USBHOST_ FCLK1	High-speed USB Host controller	USBHOST_48M_FCLK	Source is PRCM module
Functional clock	120 MHZ	USBHOST_ FCLK2	High-speed USB Host controller	USBHOST_ 120M_FCLK	Source is PRCM module
L3 master interface clock	Depending on PRCM register settings	USBHOST_ MICLK	High-speed USB Host controller	USBHOST_L3_ICLK	Source is PRCM module
L4 slave interface clock	Depending on PRCM register settings	USBHOST_ SICLK	High-speed USB Host controller	USBHOST_L4_ICLK	Source is PRCM module
Functional clock	120 MHZ	USBTLL_FCLK	USBTLL	CORE_120M_FCLK	Source is PRCM module
L4 interface clock	Depending on PRCM register settings	USBTLL_ICLK	USBTLL	CORE_L4_ICLK	Source is PRCM module

20.2.3.1.2.1 L3 Master Interface Clock

The L3 master interface clock (USBHOST_L3_ICLK) is used only by the high-speed USB host controller (USBHOST_MICLK) in the subsystem, and comes from the PRCM module. This clock is controlled by the PRCM register bits PRCM.CM_ICLKEN_USBHOST[0] (0 = disabled, 1 = enabled) and PRCM.CM_AUTOIDLE_USBHOST[0] (enables/disables automatic control of the interface clock)—see Table 20-89.

Table 20-89. High-Speed USB Controller L3 Master Interface Clock

PRCM.CM_AUTOIDLE_USBHOST[0]	PRCM.CM_ICLKEN_USBHOST[0]	Interface Clock
0	0	Disabled
0	1	Enabled
1	0	Disabled
1	1	Automatic enabling/disabling

CAUTION

The L3 master interface clock shall not be less than 30 MHZ, ULPI clock divided by 2 (this can occur during DPLL3 relocking).

20.2.3.1.2.2 L4 Slave Interface Clock

The L4 slave interface clock (USBHOST_L4_ICLK) is used only by the high-speed USB host controller (USBHOST_SICLK) in the subsystem, and comes from the PRCM module.

20.2.3.1.2.3 L4 Interface Clock

The L4 interface clock (CORE_L4_ICLK) is used only by the USBTLL module (USBTLL_ICLK) in the subsystem, and comes from the PRCM module. This clock is controlled by the PRCM register bits PRCM.CM_ICLKEN3_CORE[2] (0 = disabled, 1 = enabled) and PRCM.CM_AUTOIDLE3_CORE[2] (enables/disables automatic control of the interface clock)—see Table 20-90.

Table 20-90. USBTLL Module Interface Clock

PRCM.CM_AUTOIDLE3_CORE[2]	PRCM.CM_ICLKEN3_CORE[2]	Interface Clock
0	0	Disabled
0	1	Enabled
1	0	Disabled
1	1	Automatic enabling/disabling

20.2.3.1.2.4 Functional Clocks

Two functional clocks are provided by the PRCM module to the high-speed USB host controller: USBHOST_FLCK1 running at 48 MHZ (USBHOST_48M_FCLK) and USBHOST_FLCK2 running at 120 MHZ (USBHOST_120M_FCLK).

The USBHOST_FLCK1 functional clock is controlled by the PRCM register bit PRCM.CM_FCLKEN_USBHOST[0] (0 = disabled, 1 = enabled).

The USBHOST_FLCK2 functional clock is controlled by the PRCM register bit PRCM.CM_FCLKEN_USBHOST[1] (0 = disabled, 1 = enabled).

The CORE_120M_FCLK functional clock is used only by the USBTLL module (USBTLL_FCLK) in the subsystem, and comes from the PRCM module. This clock is controlled by the PRCM register bit PRCM.CM_FCLKEN3_CORE[2] (0 = disabled, 1 = enabled).

20.2.3.1.3 Power-Management Scheme

20.2.3.1.3.1 High-Speed USB Host Controller Power-Management Scheme

20.2.3.1.3.1.1 Overview

To save dynamic power consumption, an efficient idle scheme in the device is based on the following:

- An efficient local autoclock gating for each module
- The implementation of control sideband signals between the PRCM module and each module

This enhanced idle control allows clocks to be activated/deactivated safely without complex software intervention. In both cases, the high-speed USB host controller power management is applied only to the interface clock domain.

The high-speed USB host controller has both master (initiator) and slave (target) interfaces.

- As an initiator, the high-speed USB host controller implements the standby handshake protocol to inform the PRCM module when it enters standby mode and does not generate traffic on the interconnect.
- As a target, the high-speed USB host controller implements the IDLE handshake protocol to allow the PRCM module requiring it to enter idle mode.

Table 20-91 details the high-speed USB host controller module PRCM clock control bits.

Table 20-91. High-Speed USB Host Controller PRCM Clock Control Bits

Module Clock	Associated PRCM Clock Output	Enabled Bit	Autoidle Bit
USBHOST_FCLK1	USBHOST_48M_FCLK	PRCM.CM_FCLKEN_USBHOST[0] EN_USBHOST1 bit	N/A
USBHOST_FCLK2	USBHOST_120M_FCLK	PRCM.CM_FCLKEN_USBHOST[1] EN_USBHOST2 bit	N/A
USBHOST_MICLK	USBHOST_L3_ICLK	PRCM.CM_ICLKEN_USBHOST[0] EN_USBHOST bit	PRCM.CM_AUTOIDLE_USBHOST[0] AUTO_USBHOST bit

NOTE:

- The PRCM USBHOST_48M_FCLK output is cut at PRCM level assuming all the modules that share it have been disabled in the corresponding register. Disabling the high-speed USB host controller is a necessary but not sufficient condition.
- The PRCM USBHOST_120M_FCLK output is cut at PRCM level assuming all the modules that share it have been disabled in the corresponding register. Disabling the high-speed USB host controller is a necessary but not sufficient condition.
- The PRCM USBHOST_L3_ICLK output is cut at PRCM level assuming all the modules that share it have been disabled in the corresponding register. Disabling the high-speed USB host controller is a necessary but not sufficient condition.
- The PRCM.CM_AUTOIDLE_USBHOST[0] AUTO_USBHOST bit is used to link/unlink the high-speed USB host controller from USBHOST_L3_ICLK-related clock domain transitions.
- For further details about source clocks gating and domain transitions, see the *Power, Reset, and Clock Management* chapter.

20.2.3.1.3.1.2 L3 Master Interface Power Management

The high-speed USB host controller can go to standby mode, in which case it stops generating transactions on the interconnect. The module standby leads the PRCM to disable the USB clocks to save power.

The high-speed USB host controller has a MSTANDBY/WAIT handshake mechanism with the PRCM module (see Figure 20-94).

The module is ready to enter standby mode (indicated by the MSTANDBY signal to the PRCM asserted) when there is no USB activity and the module is idle. It means the following:

- The module is committed not to start any new transaction on its master interface.
- The whole module is idle and, therefore, the power manager can start the procedure to turn off the interface clock, if needed. This procedure must be implemented using the slave power-management protocol.

The handshake mechanism lets the module go to standby mode based on the USBHOST.UHH_SYSCONFIG[13:12] MIDLEMODE field.

Table 20-92. High-Speed USB Host Controller MIDLEMODE Settings

MIDLEMODE Value	Selected Mode	Description
0x0	Force-standby	The high-speed USB host controller enters standby mode unconditionally (MSTANDBY is asserted unconditionally).
0x1	No-standby	The high-speed USB host controller never enters standby mode (MSTANDBY is never asserted).
0x2	Smart-standby	The high-speed USB host controller is ready to enter standby mode (MSTANDBY is asserted) when there is no more activity on the USB master interface of the interconnect. MSTANDBY is asserted when the module is idle and deasserted when the module is activated by either an external USB event or an appropriate register access. The module then waits for MWAIT deassertion before a DMA transfer is started.

20.2.3.1.3.1.3 L4 Slave Interface Power Management

At PRCM level, when all the conditions to shut off the high-speed USB host controller output clocks are met (see the *Power, Reset, and Clock Management* chapter for details), the PRCM module automatically launches a hardware handshake protocol to ensure the high-speed USB host controller is ready to have its clocks switched off. Namely, the PRCM asserts an IDLE request to the high-speed USB host controller. Although this handshake is completely hardware and out of any software control, the way in which the high-speed USB host controller acknowledges the PRCM IDLE request is configurable through the USBHOST.UHH_SYSCONFIG[4:3] SIDLEMODE bit field. Table 20-93 details SIDLEMODE settings and the related acknowledgment modes.

Table 20-93. High-Speed USB Host Controller SIDLEMODE Settings

SIDLEMODE Value	Selected Mode	Description
0x0	Force-idle	The high-speed USB host controller acknowledges unconditionally the IDLE request from the PRCM, regardless of its internal operations. Because such a mode does not prevent any loss of data when the clock is switched off, the mode must be used carefully.
0x1	No-idle	The high-speed USB host controller never acknowledges any IDLE request from the PRCM. This mode is secure from a module point of view as it ensures the clocks remain active; however, it is not efficient from a power-saving perspective because it does not allow the PRCM output clock to be shut off and thus the power domain to be set to a lower power state.
0x2	Smart-idle	The high-speed USB host controller acknowledges the IDLE request basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, IRQs or DMA requests are treated. This is the best approach for an efficient system power management.

When configured in smart-idle mode, the high-speed USB host controller also offers an additional granularity on its interface clock gating. The USBHOST.UHH_SYSCONFIG[9:8] CLOCKACTIVITY bit is used to control the interface clock internal gating while module is idle. Table 20-94 details the CLOCKACTIVITY settings.

Table 20-94. High-Speed USB Host Controller CLOCKACTIVITY Settings

CLOCKACTIVITY Value	Interface Clock Effect	Description
0	OFF	Interface clock is considered for generating the acknowledgment. This setting also means the interface clock is shut down upon PRCM IDLE request.
1	ON	Interface clock is not shut down upon PRCM IDLE request. The high-speed USB host controller can potentially acknowledge the IDLE request without checking the internal functionalities linked to its clock.

CAUTION

The PRCM does not have any hardware means to read CLOCKACTIVITY settings. Software ensures a consistent programming between the high-speed USB host controller CLOCKACTIVITY and PRCM interface clock control bit. Indeed, if the USBTLL module is disabled in the PRCM.CM_ICLKEN_USBHOST register while CLOCKACTIVITY is set to 1, nothing prevents the PRCM module from asserting its IDLE request, which is acknowledged regardless of the features associated to the USBTLL module interface clock. This may lead to unpredictable behaviors.

20.2.3.1.3.2 USBTLL Module Device Power-Management Scheme

From a global system power-management perspective, when one or both of the USBTLL module clocks are no longer required, the USBTLL module can be deactivated at PRCM level in the corresponding registers.

Table 20-95 details the USBTLL module PRCM clock control bits.

Table 20-95. USBTLL Module PRCM Clock Control Bits

Module Clock	Associated PRCM Clock Output	Enabled Bit	Autoidle Bit
USBTLL_FCLK	CORE_120M_FCLK	PRCM.CM_FCLKEN3_CORE[2] EN_USBTLL bit	N/A
USBTLL_ICLK	CORE_L4_ICLK	PRCM.CM_ICLKEN3_CORE[2] EN_USBTLL bit	PRCM.CM_AUTOIDLE3_CORE [2] AUTO_USBTLL bit

NOTE:

- The PRCM CORE_120M_CLK output is cut at PRCM level assuming all the modules that share it have been disabled in the corresponding register. Disabling the USBTLL module is a necessary but not sufficient condition.
- The PRCM CORE_L4_ICLK output is cut at PRCM level assuming all the modules that share it have been disabled in the corresponding register. Disabling the USBTLL module is a necessary but not sufficient condition.
- The PRCM.CM_AUTOIDLE3_CORE[2] AUTO_USBTLL bit is used to link/unlink the USBTLL module from CORE_L4_ICLK-related clock domain transitions.
- For further details about source clocks gating and domain transitions, see the *Power, Reset, and Clock Management* chapter.

At PRCM level, when all the conditions to shut off the USBTLL_FCLK or USBTLL_ICLK output clocks are met (see the *Power, Reset, and Clock Management* chapter for details), the PRCM module automatically launches a hardware handshake protocol to ensure the USBTLL module is ready to have its clocks switched off. Namely, the PRCM asserts an IDLE request to the USBTLL module. Although this handshake is completely hardware and out of any software control, the way in which the USBTLL module acknowledges the PRCM IDLE request is configurable through the USBHOST.USBTLL_SYSCONFIG[4:3] SIDLEMODE bit field. Table 20-96 details SIDLEMODE settings and the related acknowledgment modes.

Table 20-96. USBTLL Module SIDLEMODE Settings

SIDLEMODE Value	Selected Mode	Description
0x0	Force-idle	The USBTLL module acknowledges unconditionally the IDLE request from the PRCM, regardless of its internal operations. Because such a mode does not prevent any loss of data when the clock is switched off, the mode must be used carefully.
0x1	No-idle	The USBTLL module never acknowledges any IDLE request from the PRCM. This mode is secure from a module point of view as it ensures the clocks remain active; however, it is not efficient from a power-saving perspective because it does not allow the PRCM output clock to be shut off and thus the power domain to be set to a lower power state.
0x2	Smart-idle	The USBTLL module acknowledges the IDLE request basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, IRQs or DMA requests are treated. This is the best approach for an efficient system power management.

When configured in smart-idle mode, the USBTLL module also offers an additional granularity on USBTLL_ICLK gating. The USBHOST.USBTLL_SYSCONFIG[9:8] CLOCKACTIVITY bit is used to control the USBTLL_ICLK clock internal gating while the module is idle. Table 20-97 details the CLOCKACTIVITY settings.

Table 20-97. USBTLL Module CLOCKACTIVITY Settings

CLOCKACTIVITY Value	USBTLL_ICLK Effect	Description
0	OFF	USBTLL_ICLK is considered for generating the acknowledgment. This setting also means USBTLL_ICLK is shut down upon PRCM IDLE request.
1	ON	USBTLL_ICLK is not shut down upon PRCM IDLE request. The USBTLL module can potentially acknowledge the IDLE request without checking the internal functionalities linked to its clock.

CAUTION

The PRCM does not have any hardware means to read CLOCKACTIVITY settings. Software ensures a consistent programming between the USBTLL module CLOCKACTIVITY and PRCM USBTLL_ICLK control bit. Indeed, if the USBTLL module is disabled in the PRCM.CM_ICLKEN3_CORE register while CLOCKACTIVITY is set to 1, nothing prevents the PRCM module from asserting its IDLE request, which is acknowledged regardless of the features associated to the USBTLL module interface clock. This may lead to unpredictable behaviors.

20.2.3.2 Hardware Requests

20.2.3.2.1 Interrupt Requests

Table 20-98 lists the interrupt lines that are driven out from the high-speed USB host controller to the microprocessor unit (MPU) subsystem interrupt controller (INTC).

Table 20-98. High-Speed USB Host Subsystem Interrupts

Name	Mapping	Comments
HS USB OHCI Host Controller Interrupt		
OHCI_IRQ	M_IRQ_76	Destination is MPU subsystem interrupt controller
HS USB EHCI Host Controller Interrupt		
EHCI_IRQ	M_IRQ_77	Destination is MPU subsystem interrupt controller
USBTLL Module Interrupt		
TLL_IRQ	M_IRQ_78	Destination is MPU subsystem interrupt controller

20.2.3.2.2 IDLE Handshake Protocol

The PRCM handles an IDLE handshake protocol for the high-speed USB host controller and the USBTLL module. The IDLE handshake protocol allows the PRCM requiring the high-speed USB host controller to enter idle mode. The module acknowledges when it is ready.

20.2.3.2.3 MSTANDBY Handshake Protocol

The PRCM module handles an MSTANDBY handshake protocol for the high-speed USB host controller, which initiates the MSTANDBY handshake to inform the PRCM module when it enters standby mode and does not generate traffic on interconnect.

20.2.3.2.4 Wake-Up Request

Wake-up request signal USBHOST_SWAKEUP is generated by the high-speed USB host controller to the PRCM module. Wake-up request signal USBTLL_SWAKEUP is generated by the USBTLL module to the PRCM module.

20.2.4 High-Speed USB Host Subsystem Functional Description

This section describes the functionality of the high-speed USB host subsystem by describing the high-speed USB host controller and the USBTLL module.

20.2.4.1 High-Speed USB Host Controller Functionality

The full details of the standard OHCI and EHCI host controller APIs (implemented by the current module) are not repeated here. For more information, see the following specifications:

- *Open Host Controller Interface (OHCI) specification for USB Release 1.0a*
- *Enhanced Host Controller Interface (EHCI) specification for USB Release 1.0*

20.2.4.1.1 High-Speed USB Host Controller Architecture

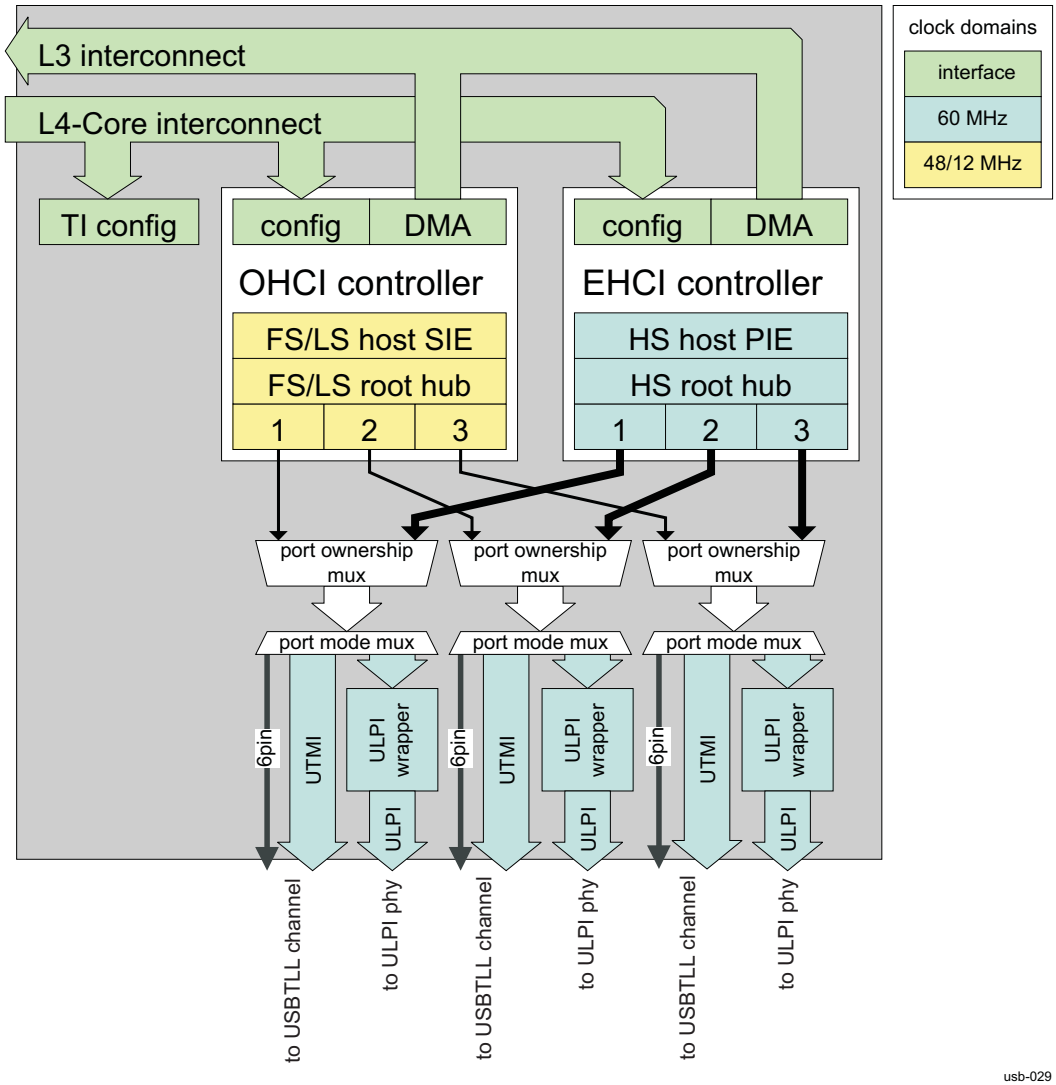
Figure 20-95 shows an overview of the high-speed USB host controller internal architecture: it contains two independent, 3-port host controllers that operate in parallel: EHCI and OHCI. Each of the three external ports is owned by exactly one of the controllers at any point in time. Each port can work in several modes:

- When the port is owned by the OHCI (full-speed) host controller, the serial 6-pin interface mode is used.
- When the port is owned by the EHCI (high-speed) host controller, either the ULPI or the UTMI modes are used.

NOTE: If one port is connected to a high-speed ULPI transceiver, all other ports must be connected to high-speed ULPI transceivers or not used.

The L4-Core interconnect is used to configure the two controllers, as well as a general, TI-specific register bank. The L3 interconnect merges and arbitrates between the transactions generated by the controller respective DMA engines.

Figure 20-95. High-Speed USB Host Controller Architecture



20.2.4.1.2 OHCI Implementation Specifications

Some features of the OHCI API are optional and/or implementation-specific. The choices made in the current implementation, the high-speed USB host controller, are described below, and are reflected in the register descriptions (see Section 20.2.6.6, *OHCI Registers*). For all standard features, see the *Open Host Controller Interface (OHCI) specification for USB Release 1.0a*.

- USBHOST.HCFMINTERVAL[30:16] FSMPS field (FullSpeedMaxPacketSize) = 0x0000: Host will stop scheduling new packets 0 bit times before the end of the frame (that is, there is no scheduling overrun protection by default). To be updated by the software driver.
- USBHOST.HCRHDESCRIPTORA[7:0] NDP field (NumberDownstreamPorts) = 0x03 = 3 ports.
- USBHOST.HCRHDESCRIPTORA[9] NPS bit (NoPowerSwitching) = 0: Ports are power-switched by default.

- USBHOST.HCRHDESCRIPTORA[8] PSM bit (PowerSwitchingMode) = 1: Per-port power switching is supported, although PPCM default setup has all ports controlled globally.
- USBHOST.HCRHDESCRIPTORA[31:24] POTPG field (PowerOnToPowerGood) = 0x0A = 10: Power rampup time is 10 x 2 ms = 20 ms.
- USBHOST.HCRHDESCRIPTORB[15:0] DR field (DeviceRemovable) = 0x0000: By default, no nonremovable devices (that is, devices attached to any of the ports) are removable.
- USBHOST.HCRHDESCRIPTORB[31:16] PPCM field (PortPowerControlMask) = 0x0000: By default, all ports are affected only by global power control.

20.2.4.1.3 UTMI Ports

The high-speed USB host controller supports N “downstream” ports, numbered from 1 through N. (In USB terminology, port 0 is necessarily an “upstream” port, and because the host is on “top” of the USB topological tree it has none). In the current implementation N = 3 (that is, available ports are 1, 2, 3).

The high-speed USB host controller is configured to be either in UTMI or in ULPI mode (see the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit).

In UTMI mode (see *USB 2.0 Transceiver Macrocell Interface specification Release 1.05*, and *UTMI+ specification Release 1.0*), all ports are in UTMI mode (that is, each port has its UTMI signal set broadcast the “outgoing“ packets — from the host to the peripherals) and gather the “incoming” ones (that is, from the addressed peripheral to the host). ULPI signal sets are undefined/don’t care on all ports.

In the device, the UTMI ports connect to the USBTLL module. The UTMI ports between the high-speed USB host controller and the USBTLL module are on-chip and remain invisible.

20.2.4.1.4 ULPI Ports

The high-speed USB host controller supports N “downstream” ports, numbered from 1 through N. (In USB terminology, port 0 is necessarily an “upstream” port, and because the host is on “top” of the USB topological tree it has none). In the current implementation N = 3 (that is, available ports are 1, 2, 3).

The high-speed USB host controller is configured to be either in UTMI or in ULPI mode (see the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit).

In ULPI mode (see *UTMI Low-Pin Interface (ULPI) specification Release 1.1*), all ports are in ULPI mode (that is, each port has its ULPI signal set broadcast the “outgoing“ packets — from the host to the peripherals) and gather the “incoming” ones (that is, from the addressed peripheral to the host). UTMI signal sets are undefined/don’t care on all ports.

When in ULPI mode, the high-speed USB host controller is in charge of generating the (nominally 60-MHZ) clock to the transceiver on the ULPI interface. This is called ULPI “input” clocking mode, because the ULPI protocol is transceiver-centric. The opposite mode, “output mode” (that is, the host receives the ULPI clock from the transceiver), is not supported and there is consequently no ULPI clock input.

In the device, the ULPI ports (only port 1 and port 2 are mapped) can only be connected directly to external transceivers. The USBTLL module is bypassed. USB traffic can be monitored directly on the USB lines.

20.2.4.1.5 Port Status

The USB port status is given through the USBHOST.UHH_HOSTCONFIG[10:8] bit field. The default value of the following bits is 1:

- USBHOST.UHH_HOSTCONFIG[8] P1_CONNECT_STATUS
- USBHOST.UHH_HOSTCONFIG[9] P2_CONNECT_STATUS
- USBHOST.UHH_HOSTCONFIG[10] P3_CONNECT_STATUS

CAUTION

These bits show the port status as connected after power on even though no USB device is connected. The USB host controller has operational status registers (for example, USBHOST.HCRHPORTSTATUS_1 for OHCI port 1) that indicate the correct port connect status. The USB driver software must read these status bits and check whether or not a port is connected. If the port is not connected, the USB driver software must reprogram the USBHOST.UHH_HOSTCONFIG bits to indicate the correct port connect status.

20.2.4.1.6 Save and Restore

The save-and-restore (SAR) mechanism can extract the hardware context of the high-speed USB host controller (after all USB activity has been suspended) before switching off (=save), save it to an external always-on memory, and reinject it later after the module has been switched on again and reset (=restore) seamlessly for the USB. Part of that context is composed of the register fields described in the current chapter. The rest of the context is composed of the "buried" flip-flops and memories (not accessible by software) like finite state-machine (FSM) states, buffer contents, and miscellaneous random logic bits.

The PRCM.PM_PWSTCTRL_USBHOSTE[4] SAVEANDRESTORE bit enables the SAR mechanism for the high-speed USB host controller (see the *Power, Reset, and Clock Management* chapter). When set, the PRCM module initiates the save and/or the restore sequences at the appropriate time. When not set, the USB host is treated as a standard module, and the save/restore sequences do not occur.

20.2.4.1.7 Burst Control

To avoid buffer underflow bursts shall be enabled by writing 0x7 in USBHOST.UHH_HOSTCONFIG[4:2] and 0x0 in USBHOST.UHH_HOSTCONFIG[5] ENA_INCR_ALIGN.

20.2.4.2 USBTLL Module Functionality

The USBTLL module implements a TLL compatible with a number of USB standard interface protocols. Once the interface protocol has been selected during an initial configuration phase, USB operation should take place seamlessly (that is, as if actual transceivers were present). To ensure maximum compatibility, as many features as possible have been included, as described in the rest of this document. The basic principle is that all the software "handles" should be available and behave in a proper way, even if there is no actual functionality underneath.

The USBTLL module is integrated with the high-speed USB host controller in the device. The transceiver interfaces (UTMI ports) between the high-speed USB host controller and the USBTLL module are on-chip and remain invisible. The other transceiver interfaces go off-chip, where they can be connected to the other controllers (for example, peripherals) on another IC.

20.2.4.2.1 Channels and Ports

Following the same convention than UTMI and ULPI, the current specification is consistently PHY-centric (that is, directions are always given with respect to the transceiver emulated here by the TLL), and not with respect to the link controller: An "input" goes from the link controller to the TLL (transceiver emulator) (that is, it is an input for the USBTLL module. Reciprocally, an "output" goes from TLL (transceiver) to the link controller (that is, it is an output for the USBTLL module).

By convention, the local link controller is the controller integrated on the same IC as the USBTLL module: This is the high-speed USB host controller in the device. The remote link controller is the other controller, located off-chip (that is, on another IC). One controller is always the USB host, the other the USB peripheral, and they communicate through the USBTLL module.

A channel is defined as a independent USB path through the USBTLL module, which always converts the UTMI+ transceiver interface protocol coming from the local link controller (the high-speed USB host controller in the device). The number of channels of the USBTLL module is three in the device.

A USB port is a set of I/O signals that carry the data and control information from/to a USB line. Several port formats exist, with different capabilities. A channel has three ports. If the channel is active, two ports are active at a time, depending on the channel configuration. The mode remains static throughout USB operation. Table 20-99 summarizes the ports features.

20.2.4.2.2 Channel Architecture

Figure 20-96 shows the architecture of a single channel (of three) of the USBTLL module, and its integration in a USB system. The ports are indicated by the circled letters (A, C, and D). See the following descriptions:

- Full arrows represent parallel, synchronous, high-speed-capable interfaces.
- Line arrows represent serial, combinatorial, full-speed/low-speed-only interfaces.
- Arrows always point toward the PHY layer (actual transceiver or TLL, in yellow), away from the link controller.
- The arrow marked ULPI* represents the entire ULPI protocol (synchronous or not) except the 3-/6-pin serial TLL modes which are reoriented toward the serial TLL block.
- The USBTLL module cannot interface with a ULPI transceiver. This functionality is provided by the high-speed USB host controller.
- Top-level multiplexing is shown only for information. It is static and does not add functionality. The figure shows ULPI and serial ports implemented on different I/O pads for clarity, but a real implementation would typically reuse the same pads, because the interfaces cannot be active simultaneously

Figure 20-96. USBTLL Channel

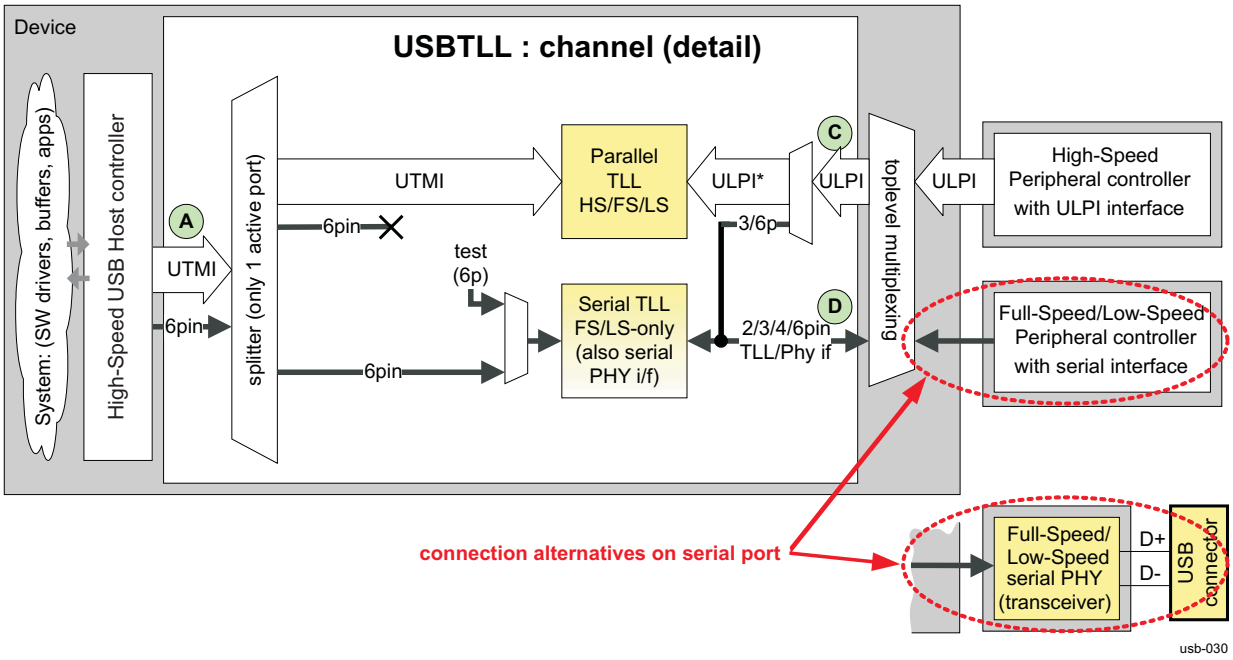


Table 20-99 summarizes the properties of each port.

Table 20-99. USBTLL Channel USB Ports

USBTLL Port	Port Description	Connect to Controller	Connect to Transceiver	Serial (Full-Speed/ Low-Speed only)	Parallel (High-Speed/ Full-Speed/Low-Speed)
A	PHY-side UTMI+	UTMI+ (L3)	No	6-pin	60-MHZ UTMI
C	PHY-side ULPI	ULPI TLL	No	6-pin	60-MHZ UTMI
D	Multimode Serial	6-/4-/3-/2-pin TLL	6-/4-/3-/2-pin	6-/4-/3-/2-pin TLL	No

20.2.4.2.2.1 Port A: PHY-side UTMI+ Port

Connects to the high-speed USB host controller in the device. The UTMI “local” port is used in all configurations (that is, the entire channel can be seen as a protocol converted from that port to one of the remote ports C or D).

- Compliant with UTMI+ version 1.0
- 8-data-bit, 60 MHZ UTMI (HS/FS/LS-capable)
- UTMI+ Level 3 extensions
- Vcontrol/Vstatus (from UTMI)
- Serial FS/LS “6-pin” mode

20.2.4.2.2.2 Port C: PHY-Side ULPI Port

Connects to a remote (off-chip) ULPI controller through I/O pads.

- Compliant with ULPI version 1.1
- SDR and DDR ULPI capable (respectively, 8-bit/4-bit data width modes)
- Supports optional 6-pin/3-pin serial modes
- Supports optional input clocking mode

20.2.4.2.2.3 Port D: Serial Multimode Port

Connects to either a serial controller (TLL modes) or a serial transceiver (transceiver interface modes).

- Supports 6-pin (TX: DAT/SE0 or TX: DP/DM) unidirectional, 4-pin bidirectional, 3-pin bidirectional, 2-pin bidirectional modes
- All modes are supported for TLL or transceiver interface configuration.
- Supports sideband signals (pullup/down control, speed/suspend enable, etc.)

20.2.4.2.3 Channel Configuration

A channel configuration is a set of software settings that specifies the connection of two of the channel ports through the USBTLL module. USB data and control injected on one side (or port) comes out on the other side (or port) after a certain amount of processing, depending on the mode. Table 20-100, lists the modes.

All configurations connect the PHY UTMI port (attached to the high-speed USB host controller) to one of the other two ports (attached to a variety of transceivers or controllers on the pads side).

Table 20-100 describes the available modes and the software settings required for each. Channel I has the following settings:

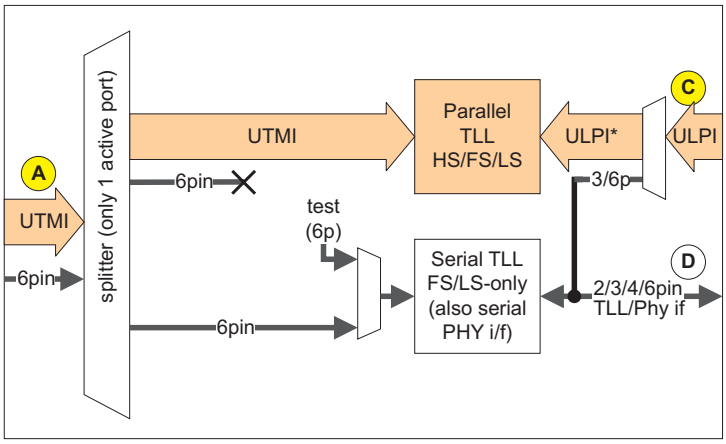
- CHANMODE: USBHOST.TLL_CHANNEL_CONF_i[2:1] CHANMODE field
- FSLSMODE: USBHOST.TLL_CHANNEL_CONF_i[27:24] FSLSMODE field
- FSLSSERIALMODE_3PIN/6PIN: Either the ULPI PHY-side USBHOST.ULPI_INTERFACE_CTRL[1] FSLSSERIALMODE_3PIN bit or the USBHOST.ULPI_INTERFACE_CTRL[0] FSLSSERIALMODE_6PIN bit (only one can be set to 1 at a time)

Table 20-100. USBTLL Channel Configuration

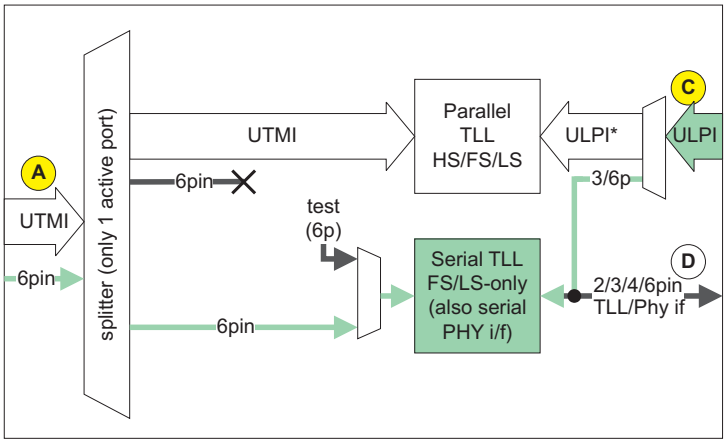
Configuration	Mode	CHAN MODE	FSLSMODE	Other Settings	Ports	Speed	Remote Port Connection
2	ULPI synchronous TLL	0	N/A	FSLSSERIALMODE_ 3PIN/6PIN = 0	A–C	HFL	ULPI link (peripheral controller)
4	Serial UTMI to serial ULPI TLL	0	N/A	FSLSSERIALMODE_ 3PIN/6PIN = 1	A–C	FL	ULPI link (peripheral controller) supporting 3-/6-pin mode
6	Serial UTMI to serial TLL	1	0x4 to 0x7; 0xA to 0xB	-	A–D	FL	Serial link (2-/3-/4-/6-pin)
6	Serial UTMI to serial PHY	1	0x0 to 0x3	-	A–D	FL	Serial transceiver (2-/3-/4-/6-pin)

Figure 20-97. Per-Configuration Datapath Through USBTLL

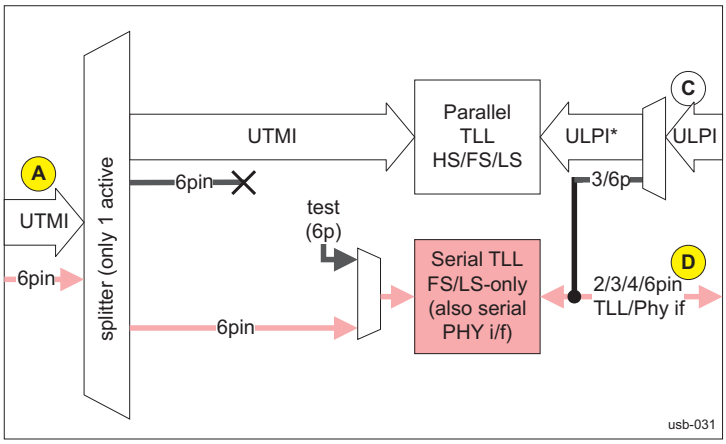
2: Sync UTMI to sync ULPI TLL config (HS/FS/LS)



4: Serial UTMI to serial ULPI TLL config (FS/LS)



6: Serial UTMI to serial ULPI, TLL or PHY config



20.2.4.2.4 VBUS Management and Emulations

In transceiver configurations, an actual USB cable is present, including an actual 5 V VBUS supply line. On the other hand, in TLL configurations, the physical USB lines are emulated and have no physical existence. This is especially true for the VBUS line, which distributes the 5-V power provided by the default host (or A-device) to the entire bus. VBUS is also used for signaling purposes, and those features must be emulated:

- A peripheral detects the presence of a host by detecting the presence of VBUS.
- USB OTG defines an elaborate voltage-sensing scheme to dynamically switch on and off VBUS (start and stop sessions). In the context of TLL, this bring no power saving compared to simple suspend.
- In particular, USB OTG uses VBUS as a wake-up source (VBUS-pulsing SRP) for the default peripheral (or B-device).

For more information on how sideband controls are integrated, see Figure 20-82 and Figure 20-83 and the related explanations.

20.2.4.2.4.1 VBUS Control and Status for Transceiver (Non-TLL) Configurations

In non-TLL modes, VBUS exists, and the problem is to propagate control and status to/from the actual VBUS manager IC (typically the transceiver itself).

Only serial transceiver configurations are concerned in the case of the high-speed USB host subsystem in the device.

20.2.4.2.4.1.1 VBUS Management in Serial Transceiver Configurations

VBUS management is not standardized in transceiver configurations. The chosen implementation is described below. See also Figure 20-82.

- VBUS control required for host and OTG operation (VBUS drive, VBUS pullup “charge”, VBUS pulldown “discharge”) is assumed to be taken care of separately from the USBTLL module (that is, by software and straight to the power IC, which can be the transceiver itself, especially in OTG cases).
- VBUS status must be sampled by the appropriate hardware (again, most of the time the transceiver itself) and reported by software to the USBTLL module, using the USBHOST.TLL_CHANNEL_CONF_i DRVVBUS and CHRGVBUS bits, as indicated in Table 20-101.

Table 20-101 lists the values to write to the USBHOST.TLL_CHANNEL_CONF_i register depending on the VBUS status observed by the transceiver on the actual VBUS line. The same register fields are also used in TLL configuration, and have been named according to that second configuration. In transceiver configurations the fields actual signification is:

- DRVVBUS: set to 1 to report a VBUS level greater than *VBUS valid.*
- CHRGVBUS: set to 1 to report a VBUS level greater than *Session valid*

Table 20-101. VBUS Level Software Reporting for Serial Transceiver Configuration

VBUS Status	USBHOST.TLL_CHANNEL_CONF_i[16] DRVVBUS Bit	USBHOST.TLL_CHANNEL_CONF_i[15] CHRGVBUS Bit
VBUS valid	1	1
Session valid (A/B)	0	1
Session not valid	0	0
Session end	0	0

20.2.4.2.4.2 VBUS Emulation for TLL Configurations

The TLL VBUS emulation sums up all actions on the VBUS line, obtains a voltage level, reported in the VBUS status bits following the protocol. The level depends on the immediate VBUS actions and has no memory of previous levels, whereas a real VBUS line behaves like an RC circuit and takes time to charge and discharge. This causes the following differences:

- The TLL level always jumps abruptly from session valid to session end (and back) with no transient time in between (where session is neither valid nor ended) as in real life.
- The Charge feature is used for VBUS-pulsing SRP, and is enabled long enough to go over the Session valid threshold, but without reaching VBUS valid. In the TLL the transition to Session valid is immediate, and VBUS valid is never reached even if the Charge is intentionally kept active.
- The Discharge feature is used in real life to accelerate the voltage drop of an undriven VBUS towards the session-end level. For TLL, this is therefore useless (although the UTMI input/ULPI register bit do exist, for compatibility), and always a “don’t care”

20.2.4.2.4.2.1 VBUS Emulation in ULPI TLL Modes

Table 20-102 summarizes the VBUS emulation in ULPI TLL modes. VBUS controls are writable, static PHY-side registers on the ULPI side, and input signals on the ULPI ports (port A). VBUS status bits are read-only, volatile PHY-side registers on the ULPI, and output signals on the ULPI ports (port A).

Table 20-102. Emulation of VBUS Levels for UTMI-to-ULPI TLL Mode

VBUS Controls (Actions)			VBUS Level	VBUS Status		
USBHOST.ULPI_OTG_CTRL[5] DRVVBUS Bit	USBHOST.ULPI_OTG_CTRL[4] CHRGVBUS Bit	USBHOST.ULPI_OTG_CTRL[3] DISCHRGVBUS Bit		USBHOST.ULPI_USB_INT_STATUS[1] VBUSVALID Bit	USBHOST.ULPI_USB_INT_STATUS[2] SESSVALID Bit	USBHOST.ULPI_USB_INT_STATUS[3] SESSEND Bit
1	X	X	VBUS valid	1	1	0
0	1	X	VBUS valid	0	1	0
0	0	X	Session end	0	0	1

20.2.4.2.4.2.2 VBUS Emulation in Serial TLL Modes

In serial TLL modes, VBUS status and control is implemented with ad-hoc sideband signals. See Figure 20-83.

VBUS control can be done in software, by writing to the following fields of the USBHOST.TLL_CHANNEL_CONF_i register:

- DRVVBUS: set to 1 to drive VBUS to 5 V (for A-device or host)
- CHRGVBUS: set to 1 to pullup VBUS (for SRP)
- There is no pulldown (discharge) control, because the emulated VBUS has no latency and VBUS level goes to the session end level as soon as it is neither driven nor pulled up.

Alternatively, VBUS drive can also be hardware-controlled through a dedicated input. (DRVVBUS register bit and input signal are actually ORed internally.)

VBUS status is available on dedicated output signals. If those outputs are not available at top level, a software alternative is to use the voltage status reported on the local controller (the high-speed USB host controller in the device) interface (through the standard UTMI+ sideband signals) and to pass it to the remote controller (a peripheral controller), by means of an ad hoc software-controller interface other than the USB itself. This is based on the fact that the level of VBUS is the same on both extremities of the bus (that is, it does not matter which side does the measurement).

20.2.4.2.5 Multimode Serial Port

The multimode serial port requires six bidirectional I/O pads to support all eight defined modes (selected in the USBHOST.TLL_CHANNEL_CONF_i[27:24] FSLSMODE field when field CHANMODE = 0x1 = UTMI-to-serial). Those modes are full-speed/low-speed only (that is, high-speed is not supported over a serial interface).

The pads are named TXEN, TXDAT, TXSE0, RXRCV, and RXDM after their functionality in standard 6-pin mode (mode 0). Each pad has an input, output, and output enable signal associated to it on the USBTLL entity.

Table 20-103 shows the functionality of each pad in each mode. USBTLL outputs are shown in yellow, inputs in blue, and bidirectional pads in green.

Table 20-103. Serial Mode Description, Signal Functionality

Usual Name	6-Pin Mode	6-Pin Mode (Alt)	3-Pin Mode	4-Pin Mode	6-Pin TLL Mode	6-Pin TLL (Alt) Mode	3-Pin TLL Mode	4-Pin TLL Mode	2-Pin TLL Mode	2-Pin TLL (Alt) Mode
USBHOST.TLL_CHANNEL_CONF_i[27:24] FSLSMODE field	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0xA	0xB
TX encoding	DAT/SE0	DP/DM	DAT/SE0	DP/DM	DAT/SE0	DP/DM	DAT/SE0	DP/DM	DAT/SE0	DP/DM
RX encoding	DP/DM/RCV	DP/DM/RCV	DAT/SE0	DP/DM/RCV	DP/DM/RCV	DP/DM/RCV	DAT/SE0	DP/DM/RCV	DAT/SE0	DP/DM
Pin usage	Unidirect	Unidirect	Bidirect	Bidirect	Unidirect	Unidirect	Bidirect	Bidirect	Bidirect	Bidirect
Pin count	6	6	3	4	6 or 5[1]	6 or 5[1]	3	4 or 3[2]	2	2
I/O Pad Function Per Mode										
TXEN	TX Enable	TX Enable	TX Enable	TX Enable	TX Enable	TX Enable	TX Enable	TX Enable	N/C	N/C
TXDAT	TX Diff Data	TX SE Plus Data	TX/RX Diff Data	TX/RX SE Plus Data	TX Diff Data	TX SE Plus Data	TX/RX Diff Data	TX/RX Diff Data	TX/RX Diff Data	TX/RX SE Plus Data
TXSE0	TX force SE0	TX SE Minus Data	TX/RX force SE0	TX/RX SE Minus Data	TX force SE0	TX SE Minus Data	TX/RX force SE0	TX/RX force SE0	TX/RX force SE0	TX/RX SE Minus Data
RXRCV	RX Diff Data	RX Diff Data	N/C	RX Diff Data	RX Diff Data	RX Diff Data	N/C	RX Diff Data	N/C	N/C
RXDM	RX SE Plus Data	RX SE Plus Data	N/C	N/C	RX SE Plus Data	RX SE Plus Data	N/C	N/C	N/C	N/C
RXDM	RX SE Minus Data	RX SE Minus Data	N/C	N/C	RX SE Minus Data	RX SE Minus Data	N/C	N/C	N/C	N/C

(1) RXRCV and RXDM carry the same info: RXDM can drive both inputs of the remote controller and RXRCV kept unused

(2) Same remark on TXDAT (for outputs) and RXRCV: TXDAT only is enough

20.2.4.2.6 Attach/Connect Emulation for Serial TLL Modes

This section applies to all serial TLL modes:

- In UTMI-to-serial mode (USBHOST.TLL_CHANNEL_CONF_i[2:1] CHANMODE field = 0x1) for all TLL values of USBHOST.TLL_CHANNEL_i[27:24] FSLSMODE field (0x4 to 0x7; 0xA to 0xB)
- In UTMI-to-ULPI TLL mode (USBHOST.TLL_CHANNEL_CONF_i[2:1] CHANMODE field = 0x0) when the ULPI bus is switched to 6-pin serial or 3-pin serial modes

In those modes, the USB bus lines are emulated by USBTLL internal logic, and are never available on the outside. The pullup/pulldown actions described in the USB specification cannot be applied directly, and the USB cable cannot be physically attached.

Because serial modes do not specify a standard format for those sideband settings, a custom software-controlled one was implemented:

- USBHOST.TLL_CHANNEL_CONF_i[4] TLLATTACH bit emulates the physical attachment of the two controllers through a TLL cable.
  - When this bit is cleared, the local controller RX path only shows the local controller (the high-speed USB host controller) actions on the bus: TX driving, pullups, pulldowns (see below). The same thing applies for the remote controller RX path (except that test override is not available).
  - As soon as the bit is set, the actions of both sides are applied to the same bus and are resolved, similar to a real bus. The RX path for both sides shows the same bus state.
- USBHOST.TLL_CHANNEL_CONF_i[5] TLLCONNECT bit emulates the USB electrical connect (that is, the pullup by the USB peripheral of one of the two USB lines [by a 1.5KOhm resistor]), which causes the linestate to transition from SE0 to J, which is detected by the USB host. The register bit is ORed with a USBTLL module input signal — the connect control can be software (L4-Core interconnect write access) or hardware (input level). The speed of the connection is determined by the TLLFULLSPEED bit below.
- USBHOST.TLL_CHANNEL_CONF_i[6] TLLFULLSPEED bit determines the speed (full or low) of the USB connect to be emulated. The connect enable (controlled as defined above) results in the pulling-up of either D+ (1 = full speed) or D- (0 = low speed): see Table 20-104.
- The 15kOhm pulldowns are implicit: because they are supposed to be turned on at least on the host side of the bus, they do not require an additional control.

NOTE: Sideband control and status actions like pullups are included in parallel (that is, nonserial) standards (UTMI, ULPI), and do not require any custom additions.

Table 20-104. Pullup Enable Emulation in Serial TLL Modes

USBHOST.TLL_CHANNEL_CONF_i Fields		Input Signal	Resulting TLL Pullup Emulation	
TLLFULLSPEED	TLLCONNECT	USB State	D+ Pullup	D- Pullup
1	0	Full-speed unconnected	Off	Off
1	1	Full-speed connected	On	Off
0	0	Low-speed unconnected	Off	Off
0	1	Low-speed connected	Off	On

20.2.4.2.7 Save and Restore

The SAR mechanism can extract the hardware context of the USBTLL module (after all USB activity has been suspended) before switching off (=save), save it to an external always-on memory, and reinject it later after the module has been switched on again and reset (=restore) seamlessly for the USB. Part of that context is composed of the register fields described in the current chapter. The rest of the context is composed of the buried flip-flops and memories (not accessible by software) like FSM states, buffer contents, and miscellaneous random logic bits.

The PRCM.PM_PWSTCTRL_CORE[4] SAVEANDRESTORE bit enables the SAR mechanism for the USBTLL module (see the *Power, Reset, and Clock Management* chapter). When set, the PRCM module initiates the save and/or the restore sequences at the appropriate time. When not set, the USB host is treated as a standard module, and the save/restore sequences do not occur.

Table 20-105 lists the USBTLL registers impacted by the SAR context.

NOTE: Because all addresses give access to the same physical register (that is, to the same piece of context), the ULPI registers with multiple accesses (write, set, clear) are listed only once in the table.

Table 20-105. USBTLL Registers Impacted by the SAR Context

Register Name	Comments on SAR Policy
USBTLL_SYSCONFIG	Except the SOFTRESET bit (write-only)
USBTLL_IRQENABLE	-
TLL_SHARED_CONF	Except the FCLK_REQ bit
TLL_CHANNEL_CONF_i	Except the FSLSLINESTATE field
ULPI_FUNCTION_CTRL_i	-
ULPI_INTERFACE_CTRL_i	-
ULPI_OTG_CTRL_i	-
ULPI_USB_INT_EN_RISE_i	-
ULPI_USB_INT_EN_FALL_i	-
ULPI_USB_INT_STATUS_i	-
ULPI_VENDOR_INT_EN_i	-
ULPI_VENDOR_INT_STATUS_i	-

20.2.5 High-Speed USB Host Subsystem Basic Programming Model

20.2.5.1 Selecting and Configuring USB Connectivity

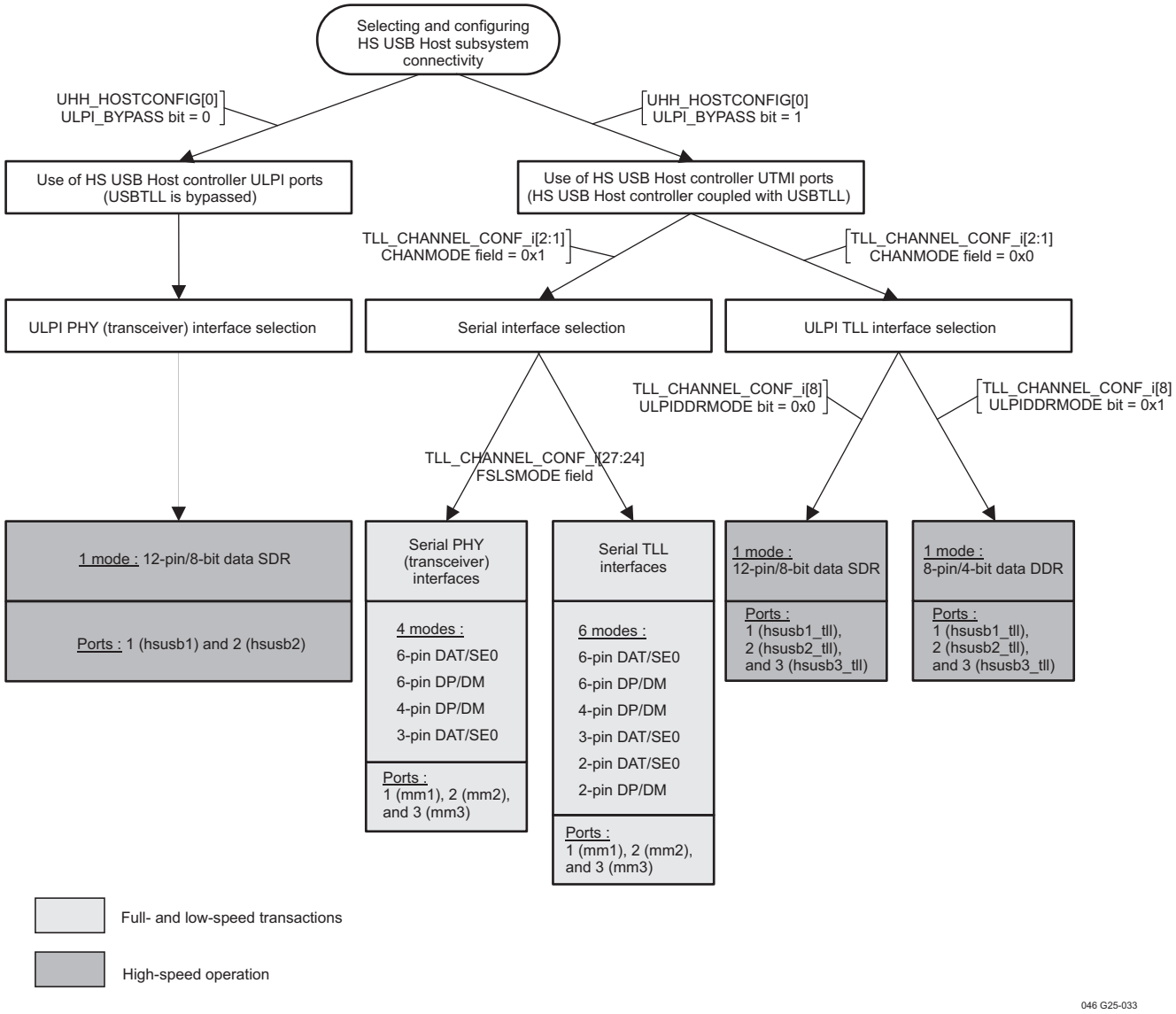
Perform the following steps to select the desired USB connectivity and configure the device accordingly.

The high-speed USB host subsystem provides three kinds of interfaces for connection:

- ULPI PHY interfaces for high-speed data transactions (up to 480 Mbps)
- Serial interfaces for full- and low-speed data transactions (up to 12 Mbps)
- ULPI TLL interfaces for high-speed data transactions (up to 480 Mbps)

Figure 20-98 shows how to select and configure the high-speed USB host subsystem connectivity.

Figure 20-98. Selecting and Configuring High-Speed USB Host Subsystem Connectivity



NOTE: When the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit is 0 (ULPI transceiver interface selection), only the ULPI ports 1 and 2 of the high-speed USB host controller can be used. Only the 12-pin/8-bit data SDR version of the ULPI interface mode is supported.

When the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit is 1 (ULPI TLL interface selection and serial interface selection), there is no restriction and the three ports can be configured in any ULPI TLL or serial mode.

20.2.5.1.1 ULPI Interface Selection

The high-speed USB host subsystem supports the following modes with the ULPI interfaces:

- External USB transceiver
  - ULPI interfaces: 12-pin/8-bit data version supporting "input" clocking mode
- TLL
  - ULPI TLL interfaces: 12-pin/8-bit data version or 8-pin/4-bit data version

20.2.5.1.1.1 Transceiver Interfaces

The USBTLL module is bypassed and the high-speed USB host controller ports 1 and 2 are connected directly to external transceivers.

The high-speed USB host subsystem supports only the 12-pin/8-bit data SDR version of the ULPI interface mode. The high-speed USB host controller uses its ULPI ports (UTMI ports cannot be used), the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit must be cleared to 0. In ULPI mode, all ports are in ULPI mode.

20.2.5.1.1.2 TLL

The high-speed USB host controller is coupled with the USBTLL module to compose the ULPI TLL interface modes.

The high-speed USB host controller uses its UTMI ports (bypassing the ULPI ports), the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit must be set to 1. In UTMI mode, all ports of the high-speed USB host controller are in UTMI mode.

At the USBTLL module level, a channel configuration for ULPI TLL interfaces must be set (see Section 20.2.4.2.3), *Channel Configurations*.

Two configurations are supported for ULPI TLL interfaces in the device:

- Configuration 2: ULPI synchronous TLL mode
- Configuration 4: Serial UTMI to serial ULPI TLL mode

In both configurations, the USBHOST.TLL_CHANNEL_CONF_i[2:1] CHANMODE field must be cleared to 0x0 (UTMI-to-ULPI TLL mode).

The selection of the ULPI TLL interface version, 12-pin/8-bit data version (SDR mode) or 8-pin/4-bit data version (DDR mode) is done through the USBHOST.TLL_CHANNEL_CONF_i[8] ULPIDDRMODE bit (0: SDR mode; 1: DDR mode).

20.2.5.1.2 Serial Interface Selection

The high-speed USB host subsystem supports the following modes with the serial interfaces:

- External USB transceiver configurations
  - Serial 6-pin PHY (transceiver) interfaces: 6-pin unidirectional (TX: DAT/SE0 or TX: DP/DM), 4-pin bidirectional and 3-pin bidirectional modes
- TLL configurations
  - Serial 6-pin TLL interfaces: 6-pin unidirectional (TX: DAT/SE0 or TX: DP/DM), 4-pin bidirectional, 3-pin bidirectional, and 2-pin bidirectional modes

The high-speed USB host controller is coupled with the USBTLL module to compose the serial interface modes.

The high-speed USB host controller uses its UTMI ports (bypassing the ULPI ports), and the USBHOST.UHH_HOSTCONFIG[0] ULPI_BYPASS bit must be set to 1. In UTMI mode, all ports of the high-speed USB host controller are in UTMI mode.

At the USBTLL module level, a channel configuration for serial interfaces must be set (see Section 20.2.4.2.3, *Channel Configurations*).

One configuration is supported for serial interfaces in the device, the Configuration 6 including two modes:

- Serial UTMI to serial TLL
- Serial UTMI to serial PHY

The multimode-mode serial interface mode selection is done through the USBHOST.TLL_CHANNEL_CONF_i[27:24] FSLSMODE field only when the main channel mode is serial (USBHOST.TLL_CHANNEL_CONF_i[2:1] CHANMODE field = 0x1 = UTMI-to-serial mode).

Table 20-106. USB Connectivity Mode Description

Usual Name	6-Pin Mode	6-Pin Mode (Alt)	3-Pin Mode	4-Pin Mode	6-Pin TLL Mode	6-Pin TLL Mode (Alt)	3-Pin TLL Mode	4-Pin TLL Mode	2-Pin TLL Mode	2-Pin TLL (Alt) Mode
USBHOST.TLL_CHANNEL_CONF_i[27:24] FSLSMODE field	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0xA	0xB
TX encoding	DAT/SE0	DP/DM	DAT/SE0	DP/DM	DAT/SE0	DP/DM	DAT/SE0	DP/DM	DAT/SE0	DP/DM
RX encoding	DP/DM/RCV	DP/DM/RCV	DAT/SE0	DP/DM/RCV	DP/DM/RCV	DP/DM/RCV	DAT/SE0	DP/DM/RCV	DAT/SE0	DP/DM
Pin usage	Unidirect	Unidirect	Bidirect	Bidirect	Unidirect	Unidirect	Bidirect	Bidirect	Bidirect	Bidirect
Pin count	6	6	3	4	6 or 5[1]	6 or 5[1]	3	4 or 3[2]	2	2

(1) RxRCV and RxDP carry the same info: RxDP can drive both inputs of the remote controller and RxRCV kept unused
(2) Same remark on TXDAT (for outputs) and RxRCV: TXDAT only is enough

20.2.5.2 USBTLL Registers

The USBTLL module contains two types of software-programmable registers.

20.2.5.2.1 TLL Control and Status Registers

Those 32-bit registers configure the various channels. Those registers are accessed by the MPU through the L4-Core interconnect. They are used mostly before the actual USB activity starts. Those registers are:

- OCP-standard registers for revision number, IRQ, clocking management, etc.
- TLL-specific registers

20.2.5.2.2 ULPI PHY-Side Registers

Each TLL channel emulates a ULPI transceiver and accordingly contains this set of 8-bit PHY-side registers, per ULPI specification. Those registers are:

- All ULPI-mandatory standard registers and fields
- A selection of ULPI-optional standard registers and fields, when relevant to the TLL context
- Vendor-specific registers, mapped at the addresses specified for that purpose in ULPI specification

Those registers are accessed by the external (that is, off-chip) link controller over the ULPI port of each channel, in the 0x100-byte ULPI address space, using the ULPI register access protocol.

Those registers are accessible by the L4-Core interconnect: The ULPI register sets of all channels are mapped side by side in the upper part of the L4-Core interconnect address space, where they can be accessed through byte accesses. In case of conflict between the two access modes, the access over ULPI will have priority, but both accesses will eventually complete correctly. For normal USB activity, all register accesses are expected to go over ULPI, and register changes caused by L4-Core interconnect accesses could compromise proper USB operation. The L4-Core interconnect access port is intended for:

- Miscellaneous test and debug
- Nonintrusive observation of ongoing USB operations (test)
- Context restore: During USB suspend periods, the USBTLL module can be switched off to save power. Upon resume, the ULPI register contents have been lost and can be restored over L4-Core

interconnect before USB operations restarts, provided they have been saved elsewhere beforehand.

CAUTION

The INSNREG04[5] bit must be set to 1 for proper behavior.

20.2.6 High-Speed USB Host Subsystem Registers

20.2.6.1 USBTLL ULPI PHY-Side Register Space

Each ULPI port emulates a separate ULPI transceiver and as such gives access to a single set of ULPI PHY-side registers, mapped in a separate ULPI register space as specified in the ULPI specification. The ULPI protocol defines two register access methods: immediate and extended.

- The immediate space maps ULPI PHY-side registers in a 0x40- (64-) byte space (address is 6 bits wide). All ULPI PHY-side registers implemented in the USBTLL implementation are in the immediate space.
- The extended register space maps all ULPI PHY-side registers in a 0x100- (256-) byte space (address is 8 bits wide). The immediate space is remapped at the bottom of the extended space (that is, the extended access method can be used to access any ULPI register).

An access is recognized as extended by first pointing to a reserved dummy address 0x2F (EXTENDED_SET_ACCESS in Table 20-107). Immediate-mode accesses to this address over the ULPI interface are forbidden by the protocol and the USBTLL behavior is then undefined. Extended accesses to this address have no effect.

Some physical registers are accessible at more than one address, where write accesses perform different actions on the register value: (over-)write, set, clear. A read to any of the addresses returns the register value. The names of the set and clear registers are the write name postfixed with respectively _SET and _CLR. The register fields are described only once, at the write address (see Section 20.2.6.7, *EHCI Register Descriptions*).

NOTE: Some ULPI registers are cleared upon read.

Table 20-107. ULPI Register Mapping Summary (For a Single ULPI Port)

Register Name	Type	Read Action	Write Action	Address Offset
VENDOR_ID_LO	R	-	-	0x0000 0000
VENDOR_ID_HI	R	-	-	0x0000 0001
PRODUCT_ID_LO	R	-	-	0x0000 0002
PRODUCT_ID_HI	R	-	-	0x0000 0003
FUNCTION_CTRL	RW	-	Overwrite	0x0000 0004
FUNCTION_CTRL_SET	RW	-	Set if 1	0x0000 0005
FUNCTION_CTRL_CLR	RW	-	Clear if 1	0x0000 0006
INTERFACE_CTRL	RW	-	Overwrite	0x0000 0007
INTERFACE_CTRL_SET	RW	-	Set if 1	0x0000 0008
INTERFACE_CTRL_CLR	RW	-	Clear if 1	0x0000 0009
OTG_CTRL	RW	-	Overwrite	0x0000 000A
OTG_CTRL_SET	RW	-	Set if 1	0x0000 000B
OTG_CTRL_CLR	RW	-	Clear if 1	0x0000 000C
USB_INT_EN_RISE	RW	-	Overwrite	0x0000 000D
USB_INT_EN_RISE_SET	RW	-	Set if 1	0x0000 000E
USB_INT_EN_RISE_CLR	RW	-	Clear if 1	0x0000 000F
USB_INT_EN_FALL	RW	-	Overwrite	0x0000 0010
USB_INT_EN_FALL_SET	RW	-	Set if 1	0x0000 0011

Table 20-107. ULPI Register Mapping Summary (For a Single ULPI Port) (continued)

Register Name	Type	Read Action	Write Action	Address Offset
USB_INT_EN_FALL_CLR	RW	-	Clear if 1	0x0000 0012
USB_INT_STATUS	R	-	-	0x0000 0013
USB_INT_LATCH	R	Clear	-	0x0000 0014
DEBUG	R	-	-	0x0000 0015
SCRATCH_REGISTER	RW	-	Overwrite	0x0000 0016
SCRATCH_REGISTER_SET	RW	-	Set if 1	0x0000 0017
SCRATCH_REGISTER_CLR	RW	-	Clear if 1	0x0000 0018
EXTENDED_SET_ACCESS	Reserved	-	N/A	0x0000 002F
UTMI_VCONTROL_EN	RW	-	Overwrite	0x0000 0030
UTMI_VCONTROL_EN_SET	RW	-	Set if 1	0x0000 0031
UTMI_VCONTROL_EN_CLR	RW	-	Clear if 1	0x0000 0032
UTMI_VCONTROL_STATUS	RW	-	Overwrite	0x0000 0033
UTMI_VCONTROL_LATCH	R	Clear	-	0x0000 0034
UTMI_VSTATUS	RW	-	Overwrite	0x0000 0035
UTMI_VSTATUS_SET	RW	-	Set if 1	0x0000 0036
UTMI_VSTATUS_CLR	RW	-	Clear if 1	0x0000 0037
USB_INT_LATCH_NOCLR	R	-	-	0x0000 0038
VENDOR_INT_EN	RW	-	Overwrite	0x0000 003B
VENDOR_INT_EN_SET	RW	-	Set if 1	0x0000 003C
VENDOR_INT_EN_CLR	RW	-	Clear if 1	0x0000 003D
VENDOR_INT_STATUS	R	-	-	0x0000 003E
VENDOR_INT_LATCH	R	Clear	-	0x0000 003F

20.2.6.2 L4-Core Interconnect Register Space

Table 20-108 lists the base address and address space for the high-speed USB host subsystem.

Table 20-108. Instance Summary

Module Name	Base Address (hex)	Size
USBTLL	0x4806 2000	4096 bytes
UHH_CONFIG	0x4806 4000	1024 bytes
OHCI	0x4806 4400	1024 bytes
EHCI	0x4806 4800	1024 bytes

20.2.6.3 High-Speed USB Host Subsystem Register Mapping Summary

The USBTLL single L4-Core interconnect gives access to both the TLL control and status registers, and the ULPI PHY-side registers.

Table 20-109 lists all the USBTLL registers mapped by the L4-Core interconnect, for the maximum 8-channel configuration.

Table 20-110, Table 20-111, and Table 20-112 list the high-speed USB host controller registers.

CAUTION

On ULPI PHY-side register access over L4-Core interconnect:

ULPI registers are byte-sized, and can only be accessed in this size. Attempts to access them over L4-Core interconnect using any other data size (16- or 32-bit) will complete without error (or any other warning), but will result in undefined behaviors.

The following cases can cause problems:

- If the ULPI register contents are defined as static (nonvolatile) by the software, a cache update may result in a burst of 32-bit access, with unwanted consequences.
- Some registers have adjacent overwrite, set, and clear addresses. An oversized write access could clear and set the same bit, which can have several results.
- Some registers are cleared on read: oversized read accesses to adjacent memory locations could cause unwanted clears.

Table 20-109. USBTLL Register Mapping Summary (L4-Core Interconnect Register Space)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
USBTLL_REVISION	R	32	0x0000 0000	0x4806 2000	Section 20.2.6.4.1
USBTLL_SYSCONFIG	RW	32	0x0000 0010	0x4806 2010	Section 20.2.6.4.2
USBTLL_SYSSTATUS	R	32	0x0000 0014	0x4806 2014	Section 20.2.6.4.3
USBTLL_IRQSTATUS	RW	32	0x0000 0018	0x4806 2018	Section 20.2.6.4.4
USBTLL_IRQENABLE	RW	32	0x0000 001C	0x4806 201C	Section 20.2.6.4.5
TLL_SHARED_CONF	RW	32	0x0000 0030	0x4806 2030	Section 20.2.6.4.6
TLL_CHANNEL_CONF_i (1)	RW	32	0x0000 0040 + (0x04 x I)	0x4806 2040 + (0x04 x I)	Section 20.2.6.4.7
ULPI_VENDOR_ID_LO_i (1)	R	8	0x0000 0800 + (0x100 x I)	0x4806 2800 + (0x100 x I)	Section 20.2.6.4.8
ULPI_VENDOR_ID_HI_i (1)	R	8	0x0000 0001 + (0x100 x I)	0x4806 2801 + (0x100 x I)	Section 20.2.6.4.9
ULPI_PRODUCT_ID_LO_i (1)	R	8	0x0000 0002 + (0x100 x I)	0x4806 2802 + (0x100 x I)	Section 20.2.6.4.10
ULPI_PRODUCT_ID_HI_i (1)	R	8	0x0000 0003 + (0x100 x I)	0x4806 2803 + (0x100 x I)	Section 20.2.6.4.11
ULPI_FUNCTION_CTRL_i (1)	RW	8	0x0000 0004 + (0x100 x I)	0x4806 2804 + (0x100 x I)	Section 20.2.6.4.12
ULPI_FUNCTION_CTRL_SET_i (1)	RW	8	0x0000 0005 + (0x100 x I)	0x4806 2805 + (0x100 x I)	Section 20.2.6.4.13
ULPI_FUNCTION_CTRL_CLR_i (1)	RW	8	0x0000 0006 + (0x100 x I)	0x4806 2806 + (0x100 x I)	Section 20.2.6.4.14
ULPI_INTERFACE_CTRL_i (1)	RW	8	0x0000 0007 + (0x100 x I)	0x4806 2807 + (0x100 x I)	Section 20.2.6.4.15
ULPI_INTERFACE_CTRL_SET_i (1)	RW	8	0x0000 0008 + (0x100 x I)	0x4806 2808 + (0x100 x I)	Section 20.2.6.4.16
ULPI_INTERFACE_CTRL_CLR_i (1)	RW	8	0x0000 0009 + (0x100 x I)	0x4806 2809 + (0x100 x I)	Section 20.2.6.4.17
ULPI_OTG_CTRL_i (1)	RW	8	0x0000 000A + (0x100 x I)	0x4806 280A + (0x100 x I)	Section 20.2.6.4.18
ULPI_OTG_CTRL_SET_i (1)	RW	8	0x0000 000B + (0x100 x I)	0x4806 280B + (0x100 x I)	Section 20.2.6.4.19

(1) i = 0 to 2

Table 20-109. USBTLL Register Mapping Summary (L4-Core Interconnect Register Space) (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
ULPI_OTG_CTRL_CLR_i [1]	RW	8	0x0000 000C + (0x100 x I)	0x4806 280C + (0x100 x I)	Section 20.2.6.4.20
ULPI_USB_INT_EN_RISE_i [1]	RW	8	0x0000 000D + (0x100 x I)	0x4806 280D + (0x100 x I)	Section 20.2.6.4.21
ULPI_USB_INT_EN_RISE_SET_i [1]	RW	8	0x0000 000E + (0x100 x I)	0x4806 280E + (0x100 x I)	Section 20.2.6.4.22
ULPI_USB_INT_EN_RISE_CLR_i [1]	RW	8	0x0000 000F + (0x100 x I)	0x4806 280F + (0x100 x I)	Section 20.2.6.4.23
ULPI_USB_INT_EN_FALL_i [1]	RW	8	0x0000 0010 + (0x100 x I)	0x4806 2810 + (0x100 x I)	Section 20.2.6.4.24
ULPI_USB_INT_EN_FALL_SET_i [1]	RW	8	0x0000 0011 + (0x100 x I)	0x4806 2811 + (0x100 x I)	Section 20.2.6.4.25
ULPI_USB_INT_EN_FALL_CLR_i [1]	RW	8	0x0000 0012 + (0x100 x I)	0x4806 2812 + (0x100 x I)	Section 20.2.6.4.26
ULPI_USB_INT_STATUS_i [1]	R	8	0x0000 0013 + (0x100 x I)	0x4806 2813 + (0x100 x I)	Section 20.2.6.4.27
ULPI_USB_INT_LATCH_i [2]	R	8	0x0000 0014 + (0x100 x I)	0x4806 2814 + (0x100 x I)	Section 20.2.6.4.28
ULPI_DEBUG_i [2]	R	8	0x0000 0015 + (0x100 x I)	0x4806 2815 + (0x100 x I)	Section 20.2.6.4.29
ULPI_SCRATCH_REGISTER_i [2]	RW	8	0x0000 0016 + (0x100 x I)	0x4806 2816 + (0x100 x I)	Section 20.2.6.4.30
ULPI_SCRATCH_REGISTER_SET_i [2]	RW	8	0x0000 0017 + (0x100 x I)	0x4806 2817 + (0x100 x I)	Section 20.2.6.4.31
ULPI_SCRATCH_REGISTER_CLR_i [2]	RW	8	0x0000 0018 + (0x100 x I)	0x4806 2818 + (0x100 x I)	Section 20.2.6.4.32
ULPI_EXTENDED_SET_ACCESS_i [2]	Rsvd	8	0x0000 002F + (0x100 x I)	0x4806 282F + (0x100 x I)	Section 20.2.6.4.33
ULPI_UTMI_VCONTROL_EN_i [2]	RW	8	0x0000 0030 + (0x100 x I)	0x4806 2830 + (0x100 x I)	Section 20.2.6.4.34
ULPI_UTMI_VCONTROL_EN_SET_i [2]	RW	8	0x0000 0031 + (0x100 x I)	0x4806 2831 + (0x100 x I)	Section 20.2.6.4.35
ULPI_UTMI_VCONTROL_EN_CLR_i [2]	RW	8	0x0000 0032 + (0x100 x I)	0x4806 2832 + (0x100 x I)	Section 20.2.6.4.36
ULPI_UTMI_VCONTROL_STATUS_i [2]	RW	8	0x0000 0033 + (0x100 x I)	0x4806 2833 + (0x100 x I)	Section 20.2.6.4.37
ULPI_UTMI_VCONTROL_LATCH_i [2]	R	8	0x0000 0034 + (0x100 x I)	0x4806 2834 + (0x100 x I)	Section 20.2.6.4.38
ULPI_UTMI_VSTATUS_i [2]	RW	8	0x0000 0035 + (0x100 x I)	0x4806 2835 + (0x100 x I)	Section 20.2.6.4.39
ULPI_UTMI_VSTATUS_SET_i [2]	RW	8	0x0000 0036 + (0x100 x I)	0x4806 2836 + (0x100 x I)	Section 20.2.6.4.40
ULPI_UTMI_VSTATUS_CLR_i [2]	RW	8	0x0000 0037 + (0x100 x I)	0x4806 2837 + (0x100 x I)	Section 20.2.6.4.41
ULPI_USB_INT_LATCH_NOCLR_i [2]	R	8	0x0000 0038 + (0x100 x I)	0x4806 2838 + (0x100 x I)	Section 20.2.6.4.42
ULPI_VENDOR_INT_EN_i [2]	RW	8	0x0000 003B + (0x100 x I)	0x4806 283B + (0x100 x I)	Section 20.2.6.4.43
ULPI_VENDOR_INT_EN_SET_i [2]	RW	8	0x0000 003C + (0x100 x I)	0x4806 283C + (0x100 x I)	Section 20.2.6.4.44
ULPI_VENDOR_INT_EN_CLR_i [2]	RW	8	0x0000 003D + (0x100 x I)	0x4806 283D + (0x100 x I)	Section 20.2.6.4.45

[2] i = 0 to 2

Table 20-109. USBTLL Register Mapping Summary (L4-Core Interconnect Register Space) (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
ULPI_VENDOR_INT_STATUS_i [2]	R	8	0x0000 003E + (0x100 x I)	0x4806 283E + (0x100 x I)	Section 20.2.6.4.46
ULPI_VENDOR_INT_LATCH_i [2]	R	8	0x0000 003F + (0x100 x I)	0x4806 283F + (0x100 x I)	Section 20.2.6.4.47

Table 20-110. UHH_CONFIG Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
UHH_REVISION	R	32	0x0000 0000	0x4806 4000	Section 20.2.6.5.1
UHH_SYSCONFIG	RW	32	0x0000 0010	0x4806 4010	Section 20.2.6.5.2
UHH_SYSSTATUS	R	32	0x0000 0014	0x4806 4014	Section 20.2.6.5.3
UHH_HOSTCONFIG	RW	32	0x0000 0040	0x4806 4040	Section 20.2.6.5.4
UHH_DEBUG_CSR	RW	32	0x0000 0044	0x4806 4044	Section 20.2.6.5.5

Table 20-111. OHCI Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
HCREVISION	R	32	0x0000 0000	0x4806 4400	Section 20.2.6.6.1
HCCONTROL	RW	32	0x0000 0004	0x4806 4404	Section 20.2.6.6.2
HCCOMMANDSTATUS	RW	32	0x0000 0008	0x4806 4408	Section 20.2.6.6.3
HCINTERRUPTSTATUS	RW	32	0x0000 000C	0x4806 440C	Section 20.2.6.6.4
HCINTERRUPTENABLE	RW	32	0x0000 0010	0x4806 4410	Section 20.2.6.6.5
HCINTERRUPTDISABLE	RW	32	0x0000 0014	0x4806 4414	Section 20.2.6.6.6
HCHCCA	RW	32	0x0000 0018	0x4806 4418	Section 20.2.6.6.7
HCPERIODCURRENTED	R	32	0x0000 001C	0x4806 441C	Section 20.2.6.6.8
HCCONTROLHEADED	RW	32	0x0000 0020	0x4806 4420	Section 20.2.6.6.9
HCCONTROLCURRENTED	RW	32	0x0000 0024	0x4806 4424	Section 20.2.6.6.10
HCBULKHEADED	RW	32	0x0000 0028	0x4806 4428	Section 20.2.6.6.11
HCBULKCURRENTED	RW	32	0x0000 002C	0x4806 442C	Section 20.2.6.6.12
HCDONEHEAD	R	32	0x0000 0030	0x4806 4430	Section 20.2.6.6.13
HCFMINTERVAL	RW	32	0x0000 0034	0x4806 4434	Section 20.2.6.6.14
HCFMREMAINING	R	32	0x0000 0038	0x4806 4438	Section 20.2.6.6.15
HCFMNUMBER	R	32	0x0000 003C	0x4806 443C	Section 20.2.6.6.16
HCPERIODICSTART	RW	32	0x0000 0040	0x4806 4440	Section 20.2.6.6.17
HCLSTHRESHOLD	RW	32	0x0000 0044	0x4806 4444	Section 20.2.6.6.18
HCRHDESCRIPTORA	RW	32	0x0000 0048	0x4806 4448	Section 20.2.6.6.19
HCRHDESCRIPTORB	RW	32	0x0000 004C	0x4806 444C	Section 20.2.6.6.20
HCRHSTATUS	RW	32	0x0000 0050	0x4806 4450	Section 20.2.6.6.21
HCRHPORTSTATUS_1	RW	32	0x0000 0054	0x4806 4454	Section 20.2.6.6.22
HCRHPORTSTATUS_2	RW	32	0x0000 0058	0x4806 4458	Section 20.2.6.6.23
HCRHPORTSTATUS_3	RW	32	0x0000 005C	0x4806 445C	Section 20.2.6.6.24

OHCI register descriptions conform to the OHCI USB standard: *Open Host controller Interface Specification for USB*, Release 1.0a.

For more information about these registers, or for new specification releases, search OHCI on www.usb.org.

Table 20-112. EHCI Register Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address	Section
HCCAPBASE	R	32	0x0000 0000	0x4806 4800	Section 20.2.6.7.1
HCSPARAMS	R	32	0x0000 0004	0x4806 4804	Section 20.2.6.7.2
HCCPARAMS	R	32	0x0000 0008	0x4806 4808	Section 20.2.6.7.3
USBCMD	RW	32	0x0000 0010	0x4806 4810	Section 20.2.6.7.4
USBSTS	RW	32	0x0000 0014	0x4806 4814	Section 20.2.6.7.5
USBINTR	RW	32	0x0000 0018	0x4806 4818	Section 20.2.6.7.6
FRINDEX	RW	32	0x0000 001C	0x4806 481C	Section 20.2.6.7.7
CTRLDSSEGMENT	R	32	0x0000 0020	0x4806 4820	Section 20.2.6.7.8
PERIODICLISTBASE	RW	32	0x0000 0024	0x4806 4824	Section 20.2.6.7.9
ASYNCLISTADDR	RW	32	0x0000 0028	0x4806 4828	Section 20.2.6.7.10
CONFIGFLAG	RW	32	0x0000 0050	0x4806 4850	Section 20.2.6.7.11
PORTSC_i[(1)]	RW	32	0x0000 0054 + (0x04 x I)	0x4806 4854 + (0x04 x I)	Section 20.2.6.7.12
INSNREG00	RW	32	0x0000 0090	0x4806 4890	Section 20.2.6.7.13
INSNREG01	RW	32	0x0000 0094	0x4806 4894	Section 20.2.6.7.14
INSNREG02	RW	32	0x0000 0098	0x4806 4898	Section 20.2.6.7.15
INSNREG03	RW	32	0x0000 009C	0x4806 489C	Section 20.2.6.7.16
INSNREG04	RW	32	0x0000 00A0	0x4806 48A0	Section 20.2.6.7.17
INSNREG05_UTMI	RW	32	0x0000 00A4	0x4806 48A4	Section 20.2.6.7.18
INSNREG05_ULPI	RW	32	0x0000 00A4	0x4806 48A4	Section 20.2.6.7.19

(1) i = 0 to 2

EHCI register descriptions conform to the EHCI USB standard: *Enhanced Host Controller Interface (EHCI) Specification for USB*, Release 1.0.

For more information about these registers or for new specification releases, search EHCI on www.usb.org.

20.2.6.4 USBTLL Register Descriptions

20.2.6.4.1 USBTLL_REVISION

Table 20-113. USBTLL_REVISION

Address Offset	0x0000 0000		
Physical Address	0x4806 2000	**Instance**	USBTLL
Description	OCP standard revision number, BCD encoded		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								MAJOR				MINOR			

Bits	Field Name	Description	Type	Reset
31:8	RESERVED		R	0x000000
7:4	MAJOR	Major revision number	R	0x0
3:0	MINOR	Minor revision number	R	0x1

20.2.6.4.2 USBTLL_SYSCONFIG

Table 20-114. USBTLL_SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x4806 2010	**Instance**	USBTLL
Description	OCP standard system configuration register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							CLOCKACTIVITY	RESERVED			SIDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:9	RESERVED	reserved	R	0x000000
8	CLOCKACTIVITY	Enable autogating of OCP-derived internal clocks while module is idle. 0x0: OCP-derived internal clocks OFF during idle 0x1: OCP-derived internal clocks ON during idle	RW	0x0
7:5	RESERVED	reserved	R	0x0
4:3	SIDLEMODE	Slave interface power management control. Idle Req/ack control 0x0: Force-Idle mode. Sidleack asserted after Idlereq assertion 0x1: No-idle mode. Sidleack never asserted. 0x2: Smart-idle mode. Sidleack asserted after Idlereq assertion when no more activity on the USB.	RW	0x0
2	ENAWAKEUP	Asynchronous wakeup generation control (Swakeup) 0x0: Wakeup generation disabled 0x1: Wakeup generation enabled	RW	0x0
1	SOFTRESET	Module software reset 0x0: no effect 0x1: Starts softreset sequence.	W	0x0
0	AUTOIDLE	Internal autogating control 0x0: Clock always running 0x1: When no activity on OCP, clock is cut off.	RW	0x1

20.2.6.4.3 USBTLL_SYSSTATUS

Table 20-115. USBTLL_SYSSTATUS

Address Offset	0x0000 0014		
Physical Address	0x4806 2014	**Instance**	USBTLL
Description	OCP standard system status register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	reserved	R	0x00000000
0	RESETDONE	Indicates when the module has entirely come out of reset 0x0: Reset is ongoing 0x1: Reset is done	R	0x0

20.2.6.4.4 USBTLL_IRQSTATUS

Table 20-116. USBTLL_IRQSTATUS

Address Offset	0x0000 0018		
Physical Address	0x4806 2018	**Instance**	USBTLL
Description	OCP standard IRQ status vector. Write 1 to clear a bit.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													ACCESS_ERROR	FCLK_END	FCLK_START

Bits	Field Name	Description	Type	Reset
31:3	RESERVED		R	0x00000000
2	ACCESS_ERROR	Access error to ULPI register over OCP: USB clock must run for that type of access to succeed. 0x0: No event pending 0x1: Event pending	RW	0x0
1	FCLK_END	Functional clock is no longer requested for USB clocking 0x0: No event pending 0x1: Event pending	RW	0x0
0	FCLK_START	Functional clock is requested for USB clocking 0x0: No event pending 0x1: Event pending	RW	0x0

20.2.6.4.5 USBTLL_IRQENABLE

Table 20-117. USBTLL_IRQENABLE

Address Offset	0x0000 001C		
Physical Address	0x4806 201C	**Instance**	USBTLL
Description	OCP standard IRQ enable vector		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													ACCESS_ERROR_EN	FCLK_END_EN	FCLK_START_EN

Bits	Field Name	Description	Type	Reset
31:3	RESERVED		R	0x00000000
2	ACCESS_ERROR_EN	Enable IRQ generation upon access error to ULPI register over OCP 0x0: IRQ event is masked 0x1: IRQ event is enabled	RW	0x0
1	FCLK_END_EN	0x0: IRQ event is masked 0x1: IRQ event is enabled	RW	0x0
0	FCLK_START_EN	0x0: IRQ event is masked 0x1: IRQ event is enabled	RW	0x0

20.2.6.4.6 TLL_SHARED_CONF

Table 20-118. TLL_SHARED_CONF

Address Offset	0x0000 0030		
Physical Address	0x4806 2030	**Instance**	USBTLL
Description	Common control register for all TLL channels		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								USB_90D_DDR_EN	USB_180D_SDR_EN	USB_DIVRATIO				FCLK_REQ	FCLK_IS_ON

Bits	Field Name	Description	Type	Reset
31:7	RESERVED		R	0x0000000
6	USB_90D_DDR_EN	Software enable/disable of the 90-degree phase shift scheme on output DDR data, when implemented. Read-only, always-0 when HDL generic ULPI_90DEG4DDR = 0. 0x0: ULPI DDR output DATA aligned with CLK 0x1: ULPI DDR output DATA delayed by 90 degree wrt CLK	RW	0x1
5	USB_180D_SDR_EN	Software enable/disable of the 180-degree phase shift scheme on output SDR data, when implemented. Read-only, always-0 when HDL generic ULPI_180DEG4SDR = 0 0x0: ULPI SDR output DATA aligned with CLK 0x1: ULPI SDR output DATA delayed by 180 degree wrt CLK	RW	0x1
4:2	USB_DIVRATIO	(Log2 of) division ratio from functional clock to USB (UTMI/ULPI) clock 0x0: div ratio is 2**0 = 1 : bypass 0x1: div ratio is 2**1 = 2 0x2: div ratio is 2**2 = 4 0x3: div ratio is 2**3 = 8 0x4: div ratio is 2**4 = 16 0x5: div ratio is 2**5 = 32 0x6: div ratio is 2**6 = 64 0x7: div ratio is 2**7 = 128	RW	0x0
1	FCLK_REQ	Functional clock request, ORed from all channels depending on their respective USB bus state. Combined with the Fclk_is_on status to generate fclk_start/end IRQs. 0x0: Func clock input is not requested by TLL 0x1: Func clock input is requested by TLL	R	0x0
0	FCLK_IS_ON	Status of the functional clock input, provided by the system to the TLL module. The TLL module will only use that clock if the current status indicated that it is ready. Combined with the Fclk_request to generate fclk_start/end IRQs. 0x0: Functional clock input is not guaranteed ON (can actually be ON, OFF, or unstable) 0x1: Functional clock input is guaranteed ON and stable	RW	0x0

20.2.6.4.7 TLL_CHANNEL_CONF_i

Table 20-119. TLL_CHANNEL_CONF_i

Address Offset	0x0000 0040 + (0x04 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2040 + (0x04 x I)	**Instance**	USBTLL
Description	Control and Status register for channel I.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		FSLSLINESTATE		FSLSMODE				RESERVED			TESTTXSE0	TESTTXDAT	TESTTXEN	TESTEN	DRVVBUS	CHRGVBUS	RESERVED			ULPINOBITSTUFF	ULPIAUTOIDLE	UTMIAUTOIDLE	ULPIDDRMODE	ULPIOUTCLKMODE	TLLFULLSPEED	TLLCONNECT	TLLATTACH	UTMIISADEV	CHANMODE		CHANEN

Bits	Field Name	Description	Type	Reset
31:30	RESERVED		R	0x0
29:28	FSLSLINESTATE	Line state for Full/Low speed serial modes Bit1 = D-/ Bit0 = D+ 0x0: Single-ended 0 0x1: Full-Speed J = differential 1 0x2: Full-Speed K = differential 0 0x3: Single-ended 1 (illegal in USB)	R	0x0
27:24	FSLSMODE	Multiple-mode serial interface's mode select. Only when main channel mode is serial. No effect in other main modes. 0x0: "6pin" unidirectional PHY I/f mode. TX encoding is Dat/Se0 (default) 0x1: "6-pin" unidirectional PHY I/f mode. TX encoding is Dp/Dm 0x2: "3-pin" bidirectional PHY I/f mode. 0x3: "4-pin" bidirectional PHY I/f mode. 0x4: "6pin" unidirectional TLL mode. TX encoding is Dat/Se0 0x5: "6pin" unidirectional TLL mode. TX encoding is Dp/Dm 0x6: "3-pin" bidirectional TLL mode. 0x7: "4-pin" bidirectional TLL mode. 0xA: "2-pin" bidirectional TLL mode. Encoding is Dat/Se0 0xB: "2-pin" bidirectional TLL mode. Encoding is Dp/Dm	RW	0x0
23:21	RESERVED		R	0x0
20	TESTTXSE0	Force-Se0 transmit override value for serial mode test Don't care if TestEn = 0 (functional mode) or = TestTxen = 1 (tx = hiz) 0x0: drive differential value on TX according to TestTXDat 0x1: drive SE0 on TX	RW	0x0
19	TESTTXDAT	Differential data transmit override value for serial mode test Don't care if TestEn = 0 (functional mode) or = TestTxen = 1 (tx = hiz) or TestSe0 = 1 (tx = se0) 0x0: Drive full-speed K = differential 0 0x1: Drive full-speed J = differential 1	RW	0x0
18	TESTTXEN	Differential data transmit override value for serial mode test Don't care if TestEn = 0 (functional mode) 0x0: Drive TX according to TestTXDat/Se0 0x1: Drive TX Hiz (no drive: pullups determine line state)	RW	0x0

Bits	Field Name	Description	Type	Reset
17	TESTEN	Enable manual test override for serial mode TX path (from local controller's UTMI port) 0x0: No override. TX is from local link controller 0x1: Override enabled	RW	0x0
16	DRVVBUS	VBUS-drive for ChanMode = serial * In TLL config, write 1 to emulate serial-side VBUS drive * In PHY config, write 1 to report "VBUS valid" status (of actual VBUS) to UTMI controller 0x0: VBUS not driven 0x1: VBUS driven to 5V	RW	0x0
15	CHRGVBUS	VBUS-drive for ChanMode = serial * In TLL config, write 1 to emulate serial-side VBUS charge / pullup (OTG) * In PHY config, write 1 to reports "session valid" status (of actual VBUS) to UTMI controller 0x0: VBUS not charged, session not valid 0x1: VBUS charged, session valid	RW	0x0
14:12	RESERVED		R	0x0
11	ULPINOBITSTUFF	Disable bitstuff emulation in ULPI TLL for ULPI ChanMode 0x0: Bitstuff enabled, following USB standard 0x1: No bitstuff or associated delays (non-standard)	RW	0x0
10	ULPIAUTOIDLE	For ChanMode = ULPI TLL only. Allow the ULPI output clock to be stopped when ULPI goes into asynchronous mode (low-power, 3-pin serial, 6-pin serial). No effect in ULPI input clock mode. 0x0: ULPI output clock always-on 0x1: ULPI output clock stops during asynchronous ULPI modes	RW	0x1
9	UTMIAUTOIDLE	For ChanMode = ULPI TLL only. Allow the UTMI clock (output) to be stopped when UTMII goes to suspended mode (suspendm = 0) 0x0: UTMI clock output always on 0x1: UTMI clock output gated upon suspend	RW	0x1
8	ULPIDDRMODE	Select single/double data rate (SDR/DDR) mode for ULPI TLL Reset value depends on hardware generics ULPI_SDR/DDR_MODE. 0x0: SDR mode (8 data bit / 12 pin) 0x1: DDR mode (4 data bit / 8 pin)	RW	0x0
7	ULPIOUTCLKMODE	ULPI clocking mode select for ULPI TLL ChanMode 0x0: ULPI clock provided by LINK (that is, off-chip). ULPI clock is input 0x1: ULPI clock provided by PHY side (that is, TLL, from functional clock). ULPI clock is output	RW	0x1
6	TLLFULLSPEED	Sets PHY speed emulation in TLL (full/slow), which determines the line to pull up upon connect. The two connect source controls are: input m(N)_tllpuen, register field TllConnect. 0x0: Connect is Low-speed: D- pullup 0x1: Connect is Full-Speed: D+ pullup	RW	0x1
5	TLLCONNECT	Emulation of Full/Low-Speed connect (that is, D+ resp D- pullup) for serial TLL modes. Speed is determined by field TllSpeed. 0x0: Unconnected 0x1: Connected	RW	0x0
4	TLLATTACH	Emulates cable attach/detach for all serial TLL modes: * ChanMode = serial, in TLL mode (FsLsMode) * ChanMode = ULPI, in serial mode (6pin/3pin TLL) 0x0: cable detach emulated on serial TLL 0x1: cable attach emulated on serial TLL	RW	0x1

Bits	Field Name	Description	Type	Reset
3	UTMIISADEV	Select the cable end "seen" by UTMI side of TLL, that is, the emulated USB cable's orientation. Note that host must always be on A side, Peripheral on B side. Reset value depends on generic DEFUTMIISHOST. 0x0: UTMI side is peripheral, ULPI side is host 0x1: UTMI side is host, ULPI side is peripheral	RW	0x1
2:1	CHANMODE	Main channel mode selection 0x0: UTMI-to-ULPI TLL mode (HS capable): to ULPI controller 0x1: UTMI-to-serial (FS/LS) mode: to serial controller (TLL) or serial PHY 0x2: Transparent UTMI mode: to UTMI PHY 0x3: No mode selected	RW	0x0
0	CHANEN	Active-high channel enable. A disabled channel is unclocked and kept under reset. 0x0: Channel #N disabled 0x1: Channel #N enabled	RW	0x0

20.2.6.4.8 ULPI_VENDOR_ID_LO_i

Table 20-120. ULPI_VENDOR_ID_LO_i

Address Offset	0x0000 0000 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2800 + (0x100 x I)	**Instance**	ULPI
Description	Lower byte of USB-IF-supplied vendor ID Value is set for all channels by HDL generic ULPI_VENDORID Default is Texas-Instruments Vendor ID = 0x0451		
Type	R		

7	6	5	4	3	2	1	0
VENDOR_ID_LO							

Bits	Field Name	Description	Type	Reset
7:0	VENDOR_ID_LO		R	0x51

20.2.6.4.9 ULPI_VENDOR_ID_HI_i

Table 20-121. ULPI_VENDOR_ID_HI_i

Address Offset	0x0000 0001 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2801 + (0x100 x I)	**Instance**	ULPI
Description	Upper byte of USB-IF-supplied 16-bit vendor ID Value is set for all channels by HDL generic ULPI_VENDORID Default is Texas-Instruments Vendor ID = 0x0451		
Type	R		

7	6	5	4	3	2	1	0
VENDOR_ID_HI							

Bits	Field Name	Description	Type	Reset
7:0	VENDOR_ID_HI		R	0x04

20.2.6.4.10 ULPI_PRODUCT_ID_LO_i

Table 20-122. ULPI_PRODUCT_ID_LO_i

Address Offset	0x0000 0002 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2802 + (0x100 x I)	**Instance**	ULPI
Description	Lower byte of vendor-chosen 16-bit product ID Value is set for all channels by HDL generic ULPI_PRODUCTID		
Type	R		

7	6	5	4	3	2	1	0
PRODUCT_ID_LO							

Bits	Field Name	Description	Type	Reset
7:0	PRODUCT_ID_LO		R	0x00

20.2.6.4.11 ULPI_PRODUCT_ID_HI_i

Table 20-123. ULPI_PRODUCT_ID_HI_i

Address Offset	0x0000 0003 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2803 + (0x100 x I)	**Instance**	ULPI
Description	Upper byte of vendor-chosen 16-bit product ID Value is set for all channels by HDL generic ULPI_PRODUCTID		
Type	R		

7	6	5	4	3	2	1	0
PRODUCT_ID_HI							

Bits	Field Name	Description	Type	Reset
7:0	PRODUCT_ID_HI		R	0x00

20.2.6.4.12 ULPI_FUNCTION_CTRL_i

Table 20-124. ULPI_FUNCTION_CTRL_i

Address Offset	0x0000 0004 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2804 + (0x100 x I)	**Instance**	ULPI
Description	Controls UTMI function settings of the PHY. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
RESERVED	SUSPENDM	RESET	OPMODE		TERMSELECT	XCVRSELECT	

Bits	Field Name	Description	Type	Reset
7	RESERVED		R	0x0
6	SUSPENDM	Active low PHY suspend: puts the ULPI bus in Low Power Mode. Automatically set back to '1' upon Low Power Mode exit. 0x0: PHY is in low-power mode 0x1: PHY is not in low-power mode	RW	0x1
5	RESET	Active high UTMI transceiver reset. Auto-cleared. Does not reset the ULPI interface or ULPI register set. 0x0: No ongoing reset/ no action 0x1: Ongoing reset / apply reset	RW	0x0
4:3	OPMODE	Select the required bit encoding style during transmit 0x0: Normal operation 0x1: Non-driving 0x2: Disable bit-stuff and NRZI encoding 0x3: Reserved	RW	0x0
2	TERMSELECT	Controls the internal 1.5Kohms pull-up resistor and 45ohms HS terminations. Control over bus resistors changes depending on XcvrSelect, OpMode, DpPulldown and DmPulldown. 0x0: HS termination enabled (other conditions) 0x1: FS termination enabled (other conditions)	RW	0x0
1:0	XCVRSELECT	Select the required transceiver speed. 0x0: Enable HS transceiver 0x1: Enable FS transceiver 0x2: Enable LS transceiver 0x3: Enable FS transceiver for LS packets (automatic FS preamble pre-pending)	RW	0x1

20.2.6.4.13 ULPI_FUNCTION_CTRL_SET_i

Table 20-125. ULPI_FUNCTION_CTRL_SET_i

Address Offset	0x0000 0005 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2805 + (0x100 x I)	**Instance**	ULPI
Description	Controls UTMI function settings of the PHY. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read/Write address of the same register.		
Type	RW		

20.2.6.4.14 ULPI_FUNCTION_CTRL_CLR_i

Table 20-126. ULPI_FUNCTION_CTRL_CLR_i

Address Offset	0x0000 0006 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2806 + (0x100 x I)	**Instance**	ULPI
Description	Controls UTMI function settings of the PHY. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.15 ULPI_INTERFACE_CTRL_i

Table 20-127. ULPI_INTERFACE_CTRL_i

Address Offset	0x0000 0007 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2807 + (0x100 x I)	**Instance**	ULPI
Description	Enables alternative interfaces and PHY features. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
INTERFACE_PROTECT_DISABLE	RESERVED		AUTORESUME	CLOCKSUSPENDM	RESERVED	FSLSSERIALMODE_3PIN	FSLSSERIALMODE_6PIN

Bits	Field Name	Description	Type	Reset
7	INTERFACE_PROTECT_DISABLE	Controls circuitry built into the PHY for protecting the ULPI interface when the link tri-states stp and data. 0x0: Enables the interface protect circuit 0x1: Disables the interface protect circuit	RW	0x0
6:5	RESERVED		R	0x0
4	AUTORESUME	Enables the PHY to automatically drive resume signaling. On by default. 0x0: AutoResume disabled 0x1: AutoResume enabled	RW	0x1
3	CLOCKSUSPENDM	Active low clock suspend for serial modes (6pin/3-pin). 0x0: ULPI clock will stop during serial modes. 0x1: ULPI clock will run during serial modes.	RW	0x0
2	RESERVED		R	0x0
1	FSLSSERIALMODE_3PIN	Sets the ULPI interface to 3-pin (FS/LS only) Serial Mode. Auto-cleared when serial mode is exited. 0x0: ULPI is not in 3-pin mode 0x1: ULPI in 3-pin serial mode	RW	0x0
0	FSLSSERIALMODE_6PIN	Sets the ULPI interface to 6-pin (FS/LS only) Serial Mode. Auto-cleared when serial mode is exited. 0x0: ULPI is not in 6-pin mode 0x1: ULPI in 6-pin serial mode	RW	0x0

20.2.6.4.16 ULPI_INTERFACE_CTRL_SET_i

Table 20-128. ULPI_INTERFACE_CTRL_SET_i

Address Offset	0x0000 0008 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2808 + (0x100 x I)	**Instance**	ULPI
Description	Enables alternative interfaces and PHY features. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read/Write address of the same register.		
Type	RW		

20.2.6.4.17 ULPI_INTERFACE_CTRL_CLR_i

Table 20-129. ULPI_INTERFACE_CTRL_CLR_i

Address Offset	0x0000 0009 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2809 + (0x100 x I)	**Instance**	ULPI
Description	Enables alternative interfaces and PHY features. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.18 ULPI_OTG_CTRL_i

Table 20-130. ULPI_OTG_CTRL_i

Address Offset	0x0000 000A + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 280A + (0x100 x I)	**Instance**	ULPI
Description	Controls UTMI+ OTG functions of the PHY. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
RESERVED		DRVVBUS	CHRGVBUS	DISCHRGVBUS	DMPULLDOWN	DPPULLDOWN	IDPULLUP

Bits	Field Name	Description	Type	Reset
7:6	RESERVED		R	0x0
5	DRVVBUS	Drive 5V on VBUS 0x0: no action 0x1: drive VBUS	RW	0x0
4	CHRGVBUS	Charge VBUS through a resistor for VBUS-pulsing SRP. 0x0: No action 0x1: Set the bit to 1	RW	0x0
3	DISCHRGVBUS	Discharge VBUS through a resistor, until the session-end VBUS state is reached. 0x0: no action 0x1: discharge VBUS	RW	0x0
2	DMPULLDOWN	Enables the 15k Ohm pull-down resistor on D- 0x0: Pull-down resistor not connected to D- 0x1: Pull-down resistor connected to D-	RW	0x1
1	DPPULLDOWN	Enables the 15k Ohm pull-down resistor on D+ 0x0: Pull-down resistor not connected to D+ 0x1: Pull-down resistor connected to D+	RW	0x1
0	IDPULLUP	Pull-up to the (OTG) ID line to allow its sampling 0x0: Disable sampling of ID line. 0x1: Enable sampling of ID line.	RW	0x0

20.2.6.4.19 ULPI_OTG_CTRL_SET_i

Table 20-131. ULPI_OTG_CTRL_SET_i

Address Offset	0x0000 000B + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 280B + (0x100 x I)	**Instance**	ULPI
Description	Controls UTMI+ OTG functions of the PHY. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read/Write address of the same register.		
Type	RW		

20.2.6.4.20 ULPI_OTG_CTRL_CLR_i

Table 20-132. ULPI_OTG_CTRL_CLR_i

Address Offset	0x0000 000C + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 280C + (0x100 x I)	**Instance**	ULPI
Description	Controls UTMI+ OTG functions of the PHY. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.21 ULPI_USB_INT_EN_RISE_i

Table 20-133. ULPI_USB_INT_EN_RISE_i

Address Offset	0x0000 000D + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 280D + (0x100 x I)	**Instance**	ULPI
Description	Enables an interrupt event notification when the corresponding status bit changes from low to high. By default, all transitions are enabled. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
RESERVED			IDGND_RISE	SESSEND_RISE	SESSVALID_RISE	VBUSVALID_RISE	HOSTDISCONNECT_RISE

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0
4	IDGND_RISE	Generate an interrupt event notification when IdGnd changes from low to high. Event is automatically masked if IdPullup bit is clear to 0 and for 50ms after IdPullup is set to 1..	RW	0x1
3	SESSEND_RISE	Generate an interrupt event notification when SessEnd changes from low to high.	RW	0x1
2	SESSVALID_RISE	Generate an interrupt event notification when SessValid changes from low to high. SessValid is the same as UTMI+ AValid.	RW	0x1
1	VBUSVALID_RISE	Generate an interrupt event notification when VbusValid changes from low to high.	RW	0x1
0	HOSTDISCONNECT_RISE	Generate an interrupt event notification when Hostdisconnect changes from low to high. Applicable only in host mode (DpPulldown and DmPulldown both set to 1b).	RW	0x1

20.2.6.4.22 ULPI_USB_INT_EN_RISE_SET_i

Table 20-134. ULPI_USB_INT_EN_RISE_SET_i

Address Offset	0x0000 000E + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 280E + (0x100 x I)	**Instance**	ULPI
Description	Enables an interrupt event notification when the corresponding status bit changes from low to high. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read/Write address of the same register.		
Type	RW		

20.2.6.4.23 ULPI_USB_INT_EN_RISE_CLR_i

Table 20-135. ULPI_USB_INT_EN_RISE_CLR_i

Address Offset	0x0000 000F + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 280F + (0x100 x I)	**Instance**	ULPI
Description	Enables an interrupt event notification when the corresponding status bit changes from low to high. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.24 ULPI_USB_INT_EN_FALL_i

Table 20-136. ULPI_USB_INT_EN_FALL_i

Address Offset	0x0000 0010 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2810 + (0x100 x I)	**Instance**	ULPI
Description	Enables an interrupt event notification when the corresponding status bit changes from high to low. By default, all transitions are enabled. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
RESERVED			IDGND_FALL	SESSEND_FALL	SESSVALID_FALL	VBUSVALID_FALL	HOSTDISCONNECT_FALL

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0
4	IDGND_FALL	Generate an interrupt event notification when IdGnd changes from high to low. Event is automatically masked if IdPullup bit is clear to 0 and for 50ms after IdPullup is set to 1.	RW	0x1
3	SESSEND_FALL	Generate an interrupt event notification when SessEnd changes from high to low.	RW	0x1
2	SESSVALID_FALL	Generate an interrupt event notification when SessValid changes from high to low. SessValid is the same as UTMI+ AValid.	RW	0x1
1	VBUSVALID_FALL	Generate an interrupt event notification when VbusValid changes from high to low.	RW	0x1
0	HOSTDISCONNECT_FALL	Generate an interrupt event notification when Hostdisconnect changes from high to low. Applicable only in host mode (DpPulldown and DmPulldown both set to 1b).	RW	0x1

20.2.6.4.25 ULPI_USB_INT_EN_FALL_SET_i

Table 20-137. ULPI_USB_INT_EN_FALL_SET_i

Address Offset	0x0000 0011 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2811 + (0x100 x I)	**Instance**	ULPI
Description	Enables an interrupt event notification when the corresponding status bit changes from high to low. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read/Write address of the same register.		
Type	RW		

20.2.6.4.26 ULPI_USB_INT_EN_FALL_CLR_i

Table 20-138. ULPI_USB_INT_EN_FALL_CLR_i

Address Offset	0x0000 0012 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2812 + (0x100 x I)	**Instance**	ULPI
Description	Enables an interrupt event notification when the corresponding status bit changes from high to low. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.27 ULPI_USB_INT_STATUS_i

Table 20-139. ULPI_USB_INT_STATUS_i

Address Offset	0x0000 0013 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2813 + (0x100 x I)	**Instance**	ULPI
Description	Indicates the current value of the interrupt source signal.		
Type	R		

7	6	5	4	3	2	1	0
RESERVED			IDGND	SESSEND	SESSVALID	VBUSVALID	HOSTDISCONNECT

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0
4	IDGND	Value of UTMI+ IdDig output. Undefined unless IdPullup = 1 0x0: ID pin is grounded = OTG A = default Host 0x1: ID pin is floating = OTG B = default Peripheral	R	0x0
3	SESSEND	Current value of UTMI+ SessEnd output. 0x0: VBUS is above Session-End threshold 0x1: VBUS is below Session-End threshold	R	0x0
2	SESSVALID	Current value of UTMI+ SessValid output. SessValid is the same as UTMI+ AValid. 0x0: VBUS is below Session-Valid threshold 0x1: VBUS is above Session-Valid threshold	R	0x0
1	VBUSVALID	Current value of UTMI+ VbusValid output. 0x0: VBUS is below Vbus-Valid threshold 0x1: VBUS is above Vbus-Valid threshold	R	0x0
0	HOSTDISCONNECT	Current value of UTMI+ Hostdisconnect output. Applicable only in host mode. Automatically reset to 0 when Low Power Mode is entered. 0x0: Peripheral not disconnected or non-host mode 0x1: Peripheral disconnected	R	0x0

20.2.6.4.28 ULPI_USB_INT_LATCH_i

Table 20-140. ULPI_USB_INT_LATCH_i

Address Offset	0x0000 0014 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2814 + (0x100 x I)	**Instance**	ULPI
Description	Set by unmasked changes on the corresponding status bits to generate the ULPI interrupt. Cleared upon read, and when Low Power Mode or Serial Mode are entered.		
Type	R		

7	6	5	4	3	2	1	0
RESERVED			IDGND_LATCH	SESSEND_LATCH	SESSVALID_LATCH	VBUSVALID_LATCH	HOSTDISCONNECT_LATCH

Bits	Field Name	Description	Type	Reset
7:5	RESERVED		R	0x0
4	IDGND_LATCH	Set to 1 by the PHY when an unmasked event occurs on IdGnd. Cleared when this register is read.	R	0x0
3	SESSEND_LATCH	Set to 1 by the PHY when an unmasked event occurs on SessEnd. Cleared when this register is read.	R	0x0
2	SESSVALID_LATCH	Set to 1 by the PHY when an unmasked event occurs on SessValid. Cleared when this register is read. SessValid is the same as UTMI+ AValid.	R	0x0
1	VBUSVALID_LATCH	Set to 1 by the PHY when an unmasked event occurs on VbusValid. Cleared when this register is read.	R	0x0
0	HOSTDISCONNECT_LATCH	Set to 1 by the PHY when an unmasked event occurs on Hostdisconnect. Cleared when this register is read. Applicable only in host mode.	R	0x0

20.2.6.4.29 ULPI_DEBUG_i

Table 20-141. ULPI_DEBUG_i

Address Offset	0x0000 0015 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2815 + (0x100 x I)	**Instance**	ULPI
Description	Indicates the current value of various signals useful for debugging.		
Type	R		

7	6	5	4	3	2	1	0
RESERVED						LINESTATE	

Bits	Field Name	Description	Type	Reset
7:2	RESERVED		R	0x00
1:0	LINESTATE	Current state of the USB line: D+ (bit 0) and D- (bit 1). 0x0: SE0 (LS/FS), Squelch (HS/Chirp) 0x1: LS: 'K' State, FS: 'J' State, HS: !Squelch, Chirp: !Squelch & HS_Differential_Receiver_Output 0x2: LS: 'J' State, FS: 'K' State, HS: Invalid, Chirp: !Squelch & !HS_Differential_Receiver_Output 0x3: SE1 (LS/FS), Invalid (HS/Chirp)	R	0x0

20.2.6.4.30 ULPI_SCRATCH_REGISTER_i

Table 20-142. ULPI_SCRATCH_REGISTER_i

Address Offset	0x0000 0016 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2816 + (0x100 x I)	**Instance**	ULPI
Description	Register byte for register access testing purposes. Value has no functional effect on PHY. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
SCRATCH							

Bits	Field Name	Description	Type	Reset
7:0	SCRATCH	Scratch data.	RW	0x00

20.2.6.4.31 ULPI_SCRATCH_REGISTER_SET_i

Table 20-143. ULPI_SCRATCH_REGISTER_SET_i

Address Offset	0x0000 0017 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2817 + (0x100 x I)	**Instance**	ULPI
Description	Register byte for register access testing purposes. Value has no functional effect on PHY. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read/Write address of the same register.		
Type	RW		

20.2.6.4.32 ULPI_SCRATCH_REGISTER_CLR_i

Table 20-144. ULPI_SCRATCH_REGISTER_CLR_i

Address Offset	0x0000 0018 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2818 + (0x100 x I)	**Instance**	ULPI
Description	Register byte for register access testing purposes. Value has no functional effect on PHY. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.33 ULPI_EXTENDED_SET_ACCESS_i

Table 20-145. ULPI_EXTENDED_SET_ACCESS_i

Address Offset	0x0000 002F + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 282F + (0x100 x I)	**Instance**	ULPI
Description	This address is used to access the extended register set, that is, addresses above 0x40		
Type	UNDEFINED_TYPE_STRING		

20.2.6.4.34 ULPI_UTMI_VCONTROL_EN_i

Table 20-146. ULPI_UTMI_VCONTROL_EN_i

Address Offset	0x0000 0030 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2830 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 Enables an interrupt notification when the corresponding vcontrol_status bit changes. Read / Write address. Lowest VCS_CTRL_WIDTH (HDL generic) bits are implemented, others are always-0, read-only. (UTMI standard is 4-bit).		
Type	RW		

7	6	5	4	3	2	1	0
VC7_EN	VC6_EN	VC5_EN	VC4_EN	VC3_EN	VC2_EN	VC1_EN	VC0_EN

Bits	Field Name	Description	Type	Reset
7	VC7_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
6	VC6_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
5	VC5_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
4	VC4_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
3	VC3_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
2	VC2_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
1	VC1_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0
0	VC0_EN	enable alt_int assertion upon vcontrol_status bit change	RW	0x0

20.2.6.4.35 ULPI_UTMI_VCONTROL_EN_SET_i

Table 20-147. ULPI_UTMI_VCONTROL_EN_SET_i

Address Offset	0x0000 0031 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2831 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 Enables an interrupt notification when the corresponding vcontrol_status bit changes. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.36 ULPI_UTMI_VCONTROL_EN_CLR_i

Table 20-148. ULPI_UTMI_VCONTROL_EN_CLR_i

Address Offset	0x0000 0032 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2832 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 Enables an interrupt notification when the corresponding vcontrol_status bit changes. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.37 ULPI_UTMI_VCONTROL_STATUS_i

Table 20-149. ULPI_UTMI_VCONTROL_STATUS_i

Address Offset	0x0000 0033 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2833 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 UTMI-standard Vcontrol vector byte is sent by the UTMI controller (other side of TLL) to its PHY (emulated here by the TLL). Alternatively, data can be also written directly into the register. Can contain any user-defined data. Vcontrol bit changes can be used to assert the ULPI ALT interrupt. Lowest VCS_CTRL_WIDTH (HDL generic) bits are implemented, others are always-0, read-only. (UTMI standard is 4-bit).		
Type	RW		

7	6	5	4	3	2	1	0
VC							

Bits	Field Name	Description	Type	Reset
7:0	VC	User-defined UTMI Control data byte	RW	0x00

20.2.6.4.38 ULPI_UTMI_VCONTROL_LATCH_i

Table 20-150. ULPI_UTMI_VCONTROL_LATCH_i

Address Offset	0x0000 0034 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2834 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 Set by unmasked changes on the corresponding vcontrol_status bits to generate the ULPI ALT interrupt. Cleared upon read, and when Low Power Mode or Serial Mode are entered. Lowest VCS_CTRL_WIDTH (HDL generic) bits are implemented, others are always-0, read-only. (UTMI standard is 4-bit).		
Type	R		

7	6	5	4	3	2	1	0
VC7_CHANGE	VC6_CHANGE	VC5_CHANGE	VC4_CHANGE	VC3_CHANGE	VC2_CHANGE	VC1_CHANGE	VC0_CHANGE

Bits	Field Name	Description	Type	Reset
7	VC7_CHANGE	unmasked change on vcontrol_status bit	R	0x0
6	VC6_CHANGE	unmasked change on vcontrol_status bit	R	0x0
5	VC5_CHANGE	unmasked change on vcontrol_status bit	R	0x0
4	VC4_CHANGE	unmasked change on vcontrol_status bit	R	0x0
3	VC3_CHANGE	unmasked change on vcontrol_status bit	R	0x0
2	VC2_CHANGE	unmasked change on vcontrol_status bit	R	0x0
1	VC1_CHANGE	unmasked change on vcontrol_status bit	R	0x0
0	VC0_CHANGE	unmasked change on vcontrol_status bit	R	0x0

20.2.6.4.39 ULPI_UTMI_VSTATUS_i

Table 20-151. ULPI_UTMI_VSTATUS_i

Address Offset	0x0000 0035 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2835 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 UTMI-standard Vstatus vector byte is sent by the PHY (emulated here by the TLL) to the UTMI controller (other side of TLL): information written into this register will go directly to the UTMI controller, and can contain any user-defined data. Read / Write address. Lowest VCS_STAT_WIDTH (HDL generic) bits are implemented, others are always-0, read-only. (UTMI standard is 8-bit).		
Type	RW		

7	6	5	4	3	2	1	0
VS							

Bits	Field Name	Description	Type	Reset
7:0	VS	User-defined UTMI Status data byte	RW	0x00

20.2.6.4.40 ULPI_UTMI_VSTATUS_SET_i

Table 20-152. ULPI_UTMI_VSTATUS_SET_i

Address Offset	0x0000 0036 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2836 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 UTMI-standard Vstatus vector byte is sent by the PHY (emulated here by the TLL) to the UTMI controller (other side of TLL): information written into this register will go directly to the UTMI controller, and can contain any user-defined data. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.41 ULPI_UTMI_VSTATUS_CLR_i

Table 20-153. ULPI_UTMI_VSTATUS_CLR_i

Address Offset	0x0000 0037 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2837 + (0x100 x I)	**Instance**	ULPI
Description	Part of non-standard UTMI-to-ULPI mailbox system, implemented if HDL generic VCS_MAILBOX bit is 1 UTMI-standard Vstatus vector byte is sent by the PHY (emulated here by the TLL) to the UTMI controller (other side of TLL): information written into this register will go directly to the UTMI controller, and can contain any user-defined data. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.42 ULPI_USB_INT_LATCH_NOCLR_i

Table 20-154. ULPI_USB_INT_LATCH_NOCLR_i

Address Offset	0x0000 0038 + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 2838 + (0x100 x I)	**Instance**	ULPI
Description	Set by unmasked changes on the corresponding status bits to generate the ULPI interrupt. Debug, non-standard address to the standard register: Register is not cleared on read. See fields description at the "clear-on-read" address of the same register.		
Type	UNDEFINED_TYPE_STRING		

20.2.6.4.43 ULPI_VENDOR_INT_EN_i

Table 20-155. ULPI_VENDOR_INT_EN_i

Address Offset	0x0000 003B + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 283B + (0x100 x I)	**Instance**	ULPI
Description	Vendor-specific interrupt enables (mask) for miscellaneous ULPI alt_int events. Read / Write address.		
Type	RW		

7	6	5	4	3	2	1	0
RESERVED							P2P_EN

Bits	Field Name	Description	Type	Reset
7:1	RESERVED		R	0x00
0	P2P_EN	Enable PHY-to-PHY ULPI wakeup upon inactive UTMI suspendm. 0x0: PHY-to-PHY wakeup enabled 0x1: PHY-to-PHY wakeup enabled	RW	0x0

20.2.6.4.44 ULPI_VENDOR_INT_EN_SET_i

Table 20-156. ULPI_VENDOR_INT_EN_SET_i

Address Offset	0x0000 003C + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 283C + (0x100 x I)	**Instance**	ULPI
Description	Vendor-specific interrupt enable bit (mask) for miscellaneous ULPI alt_int events. Read / Set address (write 1 to a bit to set it to 1, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.45 ULPI_VENDOR_INT_EN_CLR_i

Table 20-157. ULPI_VENDOR_INT_EN_CLR_i

Address Offset	0x0000 003D + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 283D + (0x100 x I)	**Instance**	ULPI
Description	Vendor-specific interrupt enables (mask) for miscellaneous ULPI alt_int events. Read / Clear address (write 1 to a bit to clear it to 0, writing 0 has no effect on bit value). See fields description at the Read / Write address of the same register.		
Type	RW		

20.2.6.4.46 ULPI_VENDOR_INT_STATUS_i

Table 20-158. ULPI_VENDOR_INT_STATUS_i

Address Offset	0x0000 003E + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 283E + (0x100 x I)	**Instance**	ULPI
Description	Vendor-specific interrupt sources for miscellaneous ULPI alt_int events.		
Type	R		

7	6	5	4	3	2	1	0
RESERVED							UTMI_SUSPENDM

Bits	Field Name	Description	Type	Reset
7:1	RESERVED		R	0x00
0	UTMI_SUSPENDM	UTMI suspendm status (active-low), source of TLL PHY-to-PHY wakeup interrupt. 0x0: UTMI interface is suspended 0x1: UTMI interface is active (not suspended)	R	0x1

20.2.6.4.47 ULPI_VENDOR_INT_LATCH_i

Table 20-159. ULPI_VENDOR_INT_LATCH_i

Address Offset	0x0000 003F + (0x100 x I)	**Index**	I = 0 to 2
Physical Address	0x4806 283F + (0x100 x I)	**Instance**	ULPI
Description	Vendor-specific interrupt latches for miscellaneous ULPI alt_int events. Cleared upon read, and when Low Power Mode or Serial Mode are entered.		
Type	R		

7	6	5	4	3	2	1	0
RESERVED							P2P_LATCH

Bits	Field Name	Description	Type	Reset
7:1	RESERVED		R	0x00
0	P2P_LATCH	PHY-to-PHY ULPI wakeup event latch. Set when ULPI is in low-power mode (suspendm = 0) and UTMI is active (suspendm = 1). 0x0: No PHY-to-PHY wakeup event latched 0x1: PHY-to-PHY wakeup event was latched, ALT interrupt active	R	0x0

20.2.6.5 UHH_CONFIG Register Descriptions

20.2.6.5.1 UHH_REVISION

Table 20-160. UHH_REVISION

Address Offset	0x0000 0000		
Physical Address	0x4806 4000	**Instance**	UHH_config
Description	Standard revision number, BCD encoded Revision = <maj>.<min>		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								MAJ_REV				MIN_REV			

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	reserved	R	0x000000
7:4	MAJ_REV	Major revision number 0..9	R	0x1
3:0	MIN_REV	Minor revision number 0..9	R	0x0

20.2.6.5.2 UHH_SYSCONFIG

Table 20-161. UHH_SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x4806 4010	**Instance**	UHH_config
Description	OCP standard system configuration register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																		MIDLEMODE		RESERVED				CLOCKACTIVITY	RESERVED				SIDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	reserved	R	0x00000
13:12	MIDLEMODE	Master interface power management control. Standby/wait control 0x0: Force-standby mode. Mstandby asserted unconditionally 0x1: No-standby mode. Mstandby never asserted. 0x2: Smart-standby mode. Mstandby asserted when initiator activity stops	RW	0x0
11:9	RESERVED	reserved	R	0x0
8	CLOCKACTIVITY	Control of clock internal gating while module is idle. One bit per clock, actual register width depends on the number of functional clocks controlled. Lower bit: Interface clock Upper bits (if any): Functional clocks 1: Clock is kept on during idle 0: Clock is switched off during idle	RW	0x0
7:5	RESERVED	reserved	R	0x0
4:3	SIDLEMODE	Slave interface power management control. Idle Req/ack control 0x0: Force-Idle mode. Sidleack asserted after Idlereq assertion 0x1: No-idle mode. Sidleack never asserted. 0x2: Smart-idle mode. Sidleack asserted upon Idlereq assertion, after target activity is over	RW	0x0
2	ENAWAKEUP	Asynchronous wakeup generation control (Swakeup) 0x0: Wakeup generation disabled 0x1: Wakeup generation enabled	RW	0x0
1	SOFTRESET	Module software reset 0x0: no effect 0x1: Starts softreset sequence.	W	0x0
0	AUTOIDLE	Internal autogating control 0x0: Clock always running 0x1: When no activity on OCP, clock is cut off.	RW	0x1

20.2.6.5.3 UHH_SYSSTATUS

Table 20-162. UHH_SYSSTATUS

Address Offset	0x0000 0014		
Physical Address	0x4806 4014	**Instance**	UHH_config
Description	OCP standard system status register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													EHCI_RESETDONE	OHCI_RESETDONE	RESETDONE

Bits	Field Name	Description	Type	Reset
31:3	RESERVED	reserved	R	0x00000000
2	EHCI_RESETDONE	Indicated when the EHCI HS host is out of reset	R	0x0
1	OHCI_RESETDONE	Indicates when the OHCI FS/LS host is out of reset	R	0x0
0	RESETDONE	Indicates when the USB Host has come out of reset 0x0: Reset is ongoing 0x1: Reset is done	R	0x0

20.2.6.5.4 UHH_HOSTCONFIG

Table 20-163. UHH_HOSTCONFIG

Address Offset	0x0000 0040		
Physical Address	0x4806 4040	**Instance**	UHH_config
Description	Static configuration of the OTG controller host		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					P3_CONNECT_STATUS	P2_CONNECT_STATUS	P1_CONNECT_STATUS	RESERVED		ENA_INCR_ALIGN	ENA_INCR16	ENA_INCR8	ENA_INCR4	RESERVED	P1_ULPI_BYPASS

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Reserved	R	0x00000000
10	P3_CONNECT_STATUS	Connection status for port 3. 0x0: USB port 3 is disconnected 0x1: USB port 3 is connected and in use (also the default state)	RW	0x1
9	P2_CONNECT_STATUS	Connection status for port 2. 0x0: USB port 2 is disconnected 0x1: USB port 2 is connected and in use (also the default state)	RW	0x1
8	P1_CONNECT_STATUS	Connection status for port 1. 0x0: USB port 1 is disconnected 0x1: USB port 1 is connected and in use (also the default state)	RW	0x1
7:6	RESERVED	Reserved	RW	0x0
5	ENA_INCR_ALIGN	Force alignment of bursts to the respective burst-size boundaries 0x0: Disable burst type 0x1: Enable burst type	RW	0x0
4	ENA_INCR16	Control the use of INCR16-type bursts (in AHB sense) 0x0: Disable burst type 0x1: Enable burst type	RW	0x0
3	ENA_INCR8	Control the use of INCR8-type bursts (in AHB sense) 0x0: Disable burst type 0x1: Enable burst type	RW	0x0
2	ENA_INCR4	Control the use of INCR4-type bursts (in AHB sense) 0x0: Disable burst type 0x1: Enable burst type	RW	0x0
1	RESERVED	Reserved	RW	0x0
0	P1_ULPI_BYPASS	Host controller (root hub) port 1 control. 0x0: ULPI port 1 is active (and UTMI port 1 is inactive) 0x1: UTMI port 1 is active (and ULPI port 1 is inactive)	RW	0

20.2.6.5.5 UHH_DEBUG_CSR

Table 20-164. UHH_DEBUG_CSR

Address Offset	0x0000 0044		
Physical Address	0x4806 4044	**Instance**	UHH_config
Description	Debug control and status for the EHCI, OHCI hosts.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												OHCI_CCS_3	OHCI_CCS_2	OHCI_CCS_1	OHCI_GLOBALSUSPEND	RESERVED								OCHI_CNTSEL	EHCI_SIMULATION_MODE	EHCI_FLADJ					

Bits	Field Name	Description	Type	Reset
31:20	RESERVED		R	0x000
19	OHCI_CCS_3	Current Connect Status of port 3 0x0: no peripheral connected 0x1: peripheral connected	R	0x0
18	OHCI_CCS_2	Current Connect Status of port 2 0x0: no peripheral connected 0x1: peripheral connected	R	0x0
17	OHCI_CCS_1	Current Connect Status of port 1 0x0: no peripheral connected 0x1: peripheral connected	R	0x0
16	OHCI_GLOBALSUSPEND	OHCI global suspend status, asserted 5ms after the suspend order. 0x0: host is not suspended 0x1: host is suspended	R	0x0
15:8	RESERVED		R	0x00
7	OCHI_CNTSEL	Selection of a shorter "1 ms" counter in OHCI host, to speed up long USB phases like reset, resume, etc (Used only for simulation.) 0x0: functional mode, 1ms = 12,000 x 12 MHZ cycles 0x1: simulation mode, 1ms = 7 x 12 MHZ cycles = 583 ns	RW	0x0
6	EHCI_SIMULATION_MODE	Sets the PHY to non-driving mode. (Used only for simulation.) 0x0: functional mode 0x1: PHY set to non-driving	RW	0x0
5:0	EHCI_FLADJ	EHCI host frame length adjust. Modify only when EHCI bitfield USBSTS.HCHalted = 1 Field value + 59,488 = 60,000 by default = number of 60 MHZ UTMI/ULPI clock cycles per 1 ms USB frame = number of 480 MHZ HS bits per 125 us HS USB microframe	RW	0x20

20.2.6.6 OHCI Register Descriptions

20.2.6.6.1 HCREVISION

Table 20-165. HCREVISION

Address Offset	0x0000 0000		
Physical Address	0x4806 4400	**Instance**	OHCI
Description	OHCI revision number		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								REV							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Reserved	R	0x000000
7:0	REV	OHCI specification revision the OHCI revision number upon which the USB host controller is based. Examples: 0x10 for 1.0 0x21 for 2.1	R	0x10

20.2.6.6.2 HCCONTROL

Table 20-166. HCCONTROL

Address Offset	0x0000 0004		
Physical Address	0x4806 4404	**Instance**	OHCI
Description	HC Operating Mode Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																					RWE	RWC	IR	HCFS		BLE	CLE	IE	PLE	CBSR	

Bits	Field Name	Description	Type	Reset
31:11	RESERVED	Reserved	R	0x000000
10	RWE	Remote wake-up enable This bit is used to enable or disable the remote wakeup feature upon detection of upstream resume signaling.	RW	0x0
9	RWC	Remote wake-up connected. This bit indicates whether the host controller supports remote wakeup signaling.	RW	0x0
8	IR	Interrupt routing. This bit determines the routing of interrupts generated by events registered in USBHOST.HCINTERRUPTSTATUS. 0x0: All interrupts are routed to the normal host bus interrupt mechanism. 0x1: interrupts are routed to the system management Interrupt.	RW	0x0
7:6	HCFS	Host Controller Functional State 0x0: HCFS: USB Reset 0x1: HCFS: USB Resume 0x2: HCFS: USB Operational 0x3: HCFS: USB Suspend	RW	0x0
5	BLE	Bulk list processing enable. 0x0: Bulk ED list is not processed after the next SOF. 0x1: Enables processing of bulk ED list in the next frame.	RW	0x0
4	CLE	Control list processing enable. 0x0: Control ED list is not processed after the next SOF. 0x1: Enables processing of control ED list in the next frame.	RW	0x0
3	IE	Isochronous ED processing enabled by Host Controller Driver. 0x0: Isochronous EDs are not processed 0x1: Enables processing of isochronous EDs.	RW	0x0
2	PLE	Periodic list enable 0x0: Periodic ED lists are not processed after the next frame. 0x1: Enables processing of periodic ED lists in the next frame.	RW	0x0
1:0	CBSR	Control/bulk service ratio. Specifies the ratio between control and bulk EDs processed in a frame. 0x0: One control ED per bulk ED 0x1: Two control ED per bulk ED 0x2: Three control ED per bulk ED 0x3: Four control ED per bulk ED	RW	0x0

20.2.6.6.3 HCCOMMANDSTATUS

Table 20-167. HCCOMMANDSTATUS

Address Offset	0x0000 0008		
Physical Address	0x4806 4408	**Instance**	OHCI
Description	HC Command and Status		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														SOC		RESERVED												OCR	BLF	CLF	HCR

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved	R	0x0000
17:16	SOC	Scheduling overrun count. This is used to monitor any persistent scheduling problems. These bits are incremented on each scheduling overrun error. It is initialized to 0x0 and wraps around at 0x3.	R	0x0
15:4	RESERVED	Reserved	R	0x000
3	OCR	Ownership change request. This bit is set to request a change of control of the host controller.	RW	0x0
2	BLF	Bulk list filled. This bit is used to indicate whether there are any TDs on the bulk list. It is set whenever it adds a TD to an ED in the bulk list.	RW	0x0
1	CLF	Control list filled. This bit is used to indicate whether there are any TDs on the control list. It is set whenever it adds a TD to an ED in the control list.	RW	0x0
0	HCR	Host controller reset. (software reset) Set this bit to initiate a USB host controller reset. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this register returns a 0. 0x0: No effect 0x1: USB host controller is reset.	RW	0x0

20.2.6.6.4 HCINTERRUPTSTATUS

Table 20-168. HCINTERRUPTSTATUS

Address Offset	0x0000 000C		
Physical Address	0x4806 440C	**Instance**	OHCI
Description	HC Interrupt Status		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	OC	RESERVED																							RHSC	FNO	UE	RD	SF	WDH	SO

Bits	Field Name	Description	Type	Reset
31	RESERVED	Reserved	R	0x0
30	OC	Ownership change. This bit is set when the USBHOST.HCCOMMANDSTATUS[3] OCR bit is set. Read 0x1: An ownership change has occurred. Write 0x0: No effect Write 0x1: Clears this bit	R	0x0
29:7	RESERVED	Reserved	R	0x000000
6	RHSC	Root hub status change. When 0x1: a root hub status change has occurred. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
5	FNO	Frame number overflow. When 0x1: a frame number overflow has occurred. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
4	UE	Unrecoverable error. When 0x1: an unrecoverable error has occurred. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
3	RD	Resume detected. When 0x1: a downstream device has issued a resume request. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
2	SF	Start of frame. When 0x1: a SOF has been issued. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
1	WDH	Write done head. When 0x1: the USB host controller has updated the HCDONEHEAD register. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
0	SO	Scheduling overrun. When 0x1: a scheduling overrun has occurred. Write 0x0: no effect. Write 0x1: clears this bit	RW	0x0

20.2.6.6.5 HCINTERRUPTENABLE

Table 20-169. HCINTERRUPTENABLE

Address Offset	0x0000 0010		
Physical Address	0x4806 4410	**Instance**	OHCI
Description	HC Interrupt Enable		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIE	OC	RESERVED																							RHSC	FNO	UE	RD	SF	WDH	SO	

Bits	Field Name	Description	Type	Reset
31	MIE	Master interrupt enable. When 0x1: allows other enabled OHCI interrupt sources to propagate to the device interrupt controller. When 0x0: OHCI interrupt sources are ignored. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0
30	OC	Ownership change. Write 0x0: No effect. Write 0x1: Enable interrupt generation due to ownership change.	RW	0x0
29:7	RESERVED	Reserved	R	0x000000
6	RHSC	Root hub status change. When 0x1 and MIE is 0x1: allows root hub status change interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: root hub status change interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0
5	FNO	Frame number overflow. When 0x1 and MIE is 0x1: allows FNO interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: FNO interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0
4	UE	Unrecoverable error. When 0x1 and MIE is 0x1: allows UE interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: UE interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0
3	RD	Resume detected. When 0x1 and MIE is 0x1: allows RD interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: RD interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0
2	SF	Start of frame. When 0x1 and MIE is 0x1: allows SF interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: SF interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0
1	WDH	Write done head. When 0x1 and MIE is 0x1: allows WDH interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: WDH interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0

Bits	Field Name	Description	Type	Reset
0	SO	Scheduling overrun. When 0x1 and MIE is 0x1: allows SO interrupts to propagate to the device interrupt controller. When 0x0 or MIE is 0x0: SO interrupts do not propagate. Write 0x0 no effect. Write 0x1: sets this bit.	RW	0x0

20.2.6.6.6 HCINTERRUPTDISABLE

Table 20-170. HCINTERRUPTDISABLE

Address Offset	0x0000 0014		
Physical Address	0x4806 4414	**Instance**	OHCI
Description	HC Interrupt Disable		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIE	OC	RESERVED																						RHSC	FNO	UE	RD	SF	WDH	SO	

Bits	Field Name	Description	Type	Reset
31	MIE	Master interrupt enable. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE MIE bit.	RW	0x0
30	OC	Ownership change. Write 0x0: No effect. Write 0x1: Disable interrupt generation due to ownership change.	RW	0x0
29:7	RESERVED	Reserved	R	0x000000
6	RHSC	Root hub status change. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE RHSC bit.	RW	0x0
5	FNO	Frame number overflow. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE FNO bit.	RW	0x0
4	UE	Unrecoverable error. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE UE bit.	RW	0x0
3	RD	Resume detected. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE RD bit.	RW	0x0
2	SF	Start of frame. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE SF bit.	RW	0x0
1	WDH	Write done head. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE WDH bit.	RW	0x0
0	SO	Scheduling overrun. Always reads 0x0. Write 0x0: no effect. Write 0x1: clears the HCINTERRUPTENABLE SO bit.	RW	0x0

20.2.6.6.7 HCHCCA

Table 20-171. HCHCCA

Address Offset	0x0000 0018		
Physical Address	0x4806 4418	**Instance**	OHCI
Description	HC HCCA Address Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HCCA																								RESERVED							

Bits	Field Name	Description	Type	Reset
31:8	HCCA	Physical address of the beginning of the HCCA.	RW	0x000000
7:0	RESERVED	Reserved	R	0x00

20.2.6.6.8 HCPERIODCURRENTED

Table 20-172. HCPERIODCURRENTED

Address Offset	0x0000 001C		
Physical Address	0x4806 441C	**Instance**	OHCI
Description	HC Current Periodic Register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCED																												RESERVED			

Bits	Field Name	Description	Type	Reset
31:4	PCED	Physical address of current ED on the periodic ED list.	R	0x0000000
3:0	RESERVED	Reserved	R	0x0

20.2.6.6.9 HCCONTROLHEADED

Table 20-173. HCCONTROLHEADED

Address Offset	0x0000 0020		
Physical Address	0x4806 4420	**Instance**	OHCI
Description	HC Head Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHED																												RESERVED			

Bits	Field Name	Description	Type	Reset
31:4	CHED	Physical address of head ED on the control ED list.	RW	0x0000000
3:0	RESERVED	Reserved	R	0x0

20.2.6.6.10 HCCONTROLCURRENTED

Table 20-174. HCCONTROLCURRENTED

Address Offset	0x0000 0024		
Physical Address	0x4806 4424	**Instance**	OHCI
Description	HC Current Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCED																												RESERVED			

Bits	Field Name	Description	Type	Reset
31:4	CCED	Physical address of current ED on the control ED list.	RW	0x0000000
3:0	RESERVED	Reserved	R	0x0

20.2.6.6.11 HCBULKHEADED

Table 20-175. HCBULKHEADED

Address Offset	0x0000 0028		
Physical Address	0x4806 4428	**Instance**	OHCI
Description	HC Head Bulk Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BHED																												RESERVED			

Bits	Field Name	Description	Type	Reset
31:4	BHED	Physical address of head ED on the bulk ED list.	RW	0x0000000
3:0	RESERVED	Reserved	R	0x0

20.2.6.6.12 HCBULKCURRENTED

Table 20-176. HCBULKCURRENTED

Address Offset	0x0000 002C		
Physical Address	0x4806 442C	**Instance**	OHCI
Description	HC Current Bulk Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCED																												RESERVED			

Bits	Field Name	Description	Type	Reset
31:4	BCED	Physical address of current ED on the bulk ED list.	RW	0x0000000
3:0	RESERVED	Reserved	R	0x0

20.2.6.6.13 HCDONEHEAD

Table 20-177. HCDONEHEAD

Address Offset	0x0000 0030		
Physical Address	0x4806 4430	**Instance**	OHCI
Description	HC Head Done Register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DH																												RESERVED			

Bits	Field Name	Description	Type	Reset
31:4	DH	Physical address of last TD that was added to the Done queue.	R	0x0000000
3:0	RESERVED	Reserved	R	0x0

20.2.6.6.14 HCFMINTERVAL

Table 20-178. HCFMINTERVAL

Address Offset	0x0000 0034		
Physical Address	0x4806 4434	**Instance**	OHCI
Description	HC Frame Interval Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIT	FSMPS															RESERVED	FI														

Bits	Field Name	Description	Type	Reset
31	FIT	Frame interval toggle. This bit is toggled whenever it loads a new value to FI.	RW	0x0
30:16	FSMPS	Largest data packet size for full-speed packets, bit times. This field specifies a value which is loaded into the largest data packet counter at the beginning of each frame.	RW	0x0000
15:14	RESERVED	Reserved	R	0x0
13:0	FI	Frame interval. Number of 12-MHZ clocks in the USB frame. The nominal value is set to 11,999, to give a 1-ms frame.	RW	0x2EDF

20.2.6.6.15 HCFMREMAINING

Table 20-179. HCFMREMAINING

Address Offset	0x0000 0038		
Physical Address	0x4806 4438	**Instance**	OHCI
Description	HC Frame Remaining Register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRT	RESERVED																							FR							

Bits	Field Name	Description	Type	Reset
31	FRT	Frame remaining toggle. This bit is used for the synchronization between USBHOST.HCFMINTERVAL[13:0] FI and FR. This bit is loaded from the USBHOST.HCFMINTERVAL[31] FIT field whenever FR reaches 0.	R	0x0
30:14	RESERVED	Reserved	R	0x00000
13:0	FR	Frame remaining. This counter is decremented at each bit time. When it reaches 0, it is reset by loading the USBHOST.HCFMINTERVAL[13:0] FI value at the next bit time boundary.	R	0x0000

20.2.6.6.16 HCFMNUMBER

Table 20-180. HCFMNUMBER

Address Offset	0x0000 003C		
Physical Address	0x4806 443C	**Instance**	OHCI
Description	HC Frame Number Register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																FN															

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved	R	0x0000
15:0	FN	Frame Number. This is incremented when USBHOST.HCFMREMAINING is reloaded. It is rolled over to 0x0000 after 0xFFFF.	R	0x0000

20.2.6.6.17 HCPERIODICSTART

Table 20-181. HCPERIODICSTART

Address Offset	0x0000 0040		
Physical Address	0x4806 4440	**Instance**	OHCI
Description	HC Periodic Start Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								PS							

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Reserved	R	0x00000
13:0	PS	Periodic start. The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.	RW	0x0000

20.2.6.6.18 HCLSTHRESHOLD

Table 20-182. HCLSTHRESHOLD

Address Offset	0x0000 0044		
Physical Address	0x4806 4444	**Instance**	OHCI
Description	HC Low-Speed Threshold Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				LST											

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Reserved	R	0x00000
11:0	LST	Low-speed threshold.	RW	0x628

20.2.6.6.19 HCRHDESCRIPTORA

Table 20-183. HCRHDESCRIPTORA

Address Offset	0x0000 0048		
Physical Address	0x4806 4448	**Instance**	OHCI
Description	HC Root Hub A Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POTPG								RESERVED													DT	NPS	PSM	NDP							

Bits	Field Name	Description	Type	Reset
31:24	POTPG	Power-on to power-good time. Defines the minimum amount of time (2 ms * POTPG) between the USB host controller turning on power to a downstream port and when the USB host can access the downstream device.	RW	0x0A
23:11	RESERVED	Reserved	R	0x000
10	DT	Device type. Always reads 0x0: indicates that the USB host controller implemented is not a compound device.	R	0x0
9	NPS	No power switching 0x0: VBUS power switching is supported, either per-port or all-port switched per the power 0x1: VBUS power switching is not supported, power is available to all downstream ports.	RW	0x0
8	PSM	Power switching mode. 0x0: Indicates that all ports are powered at the same time 0x1: Individual port power switching is supported	RW	0x1
7:0	NDP	Number of downstream ports. These bits specify the number of downstream ports supported by the root hub. It is implementation-specific. The minimum number of ports is 1. The maximum number of ports supported by OHCI is 15.	R	0x03

20.2.6.6.20 HCRHDESCRIPTORB

Table 20-184. HCRHDESCRIPTORB

Address Offset	0x0000 004C		
Physical Address	0x4806 444C	**Instance**	OHCI
Description	HC Root Hub B Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPCM																DR															

Bits	Field Name	Description	Type	Reset
31:16	PPCM	Port power control mask. Each bit defines whether a corresponding downstream port has port power controlled by the global power control. When set the port's power state is only affected by per-port power control. When cleared the port is controlled by the global power switch. If the device is configured to global switch mode this field is not valid. bit 0: reserved, bit 1: Ganged-power mask on port #1, ..., bit 15: Ganged-power mask on port #15	RW	0x0000
15:0	DR	Device removable. Each bit defines whether a corresponding downstream port has a removable device. When cleared the attached device is removable. When set the attached device is not removable. Bit 0: reserved, bit 1 : Device attached to port #1, &, bit 15: Device attached to port #15	RW	0x0000

20.2.6.6.21 HCRHSTATUS

Table 20-185. HCRHSTATUS

Address Offset	0x0000 0050		
Physical Address	0x4806 4450	**Instance**	OHCI
Description	HC Root Hub Status Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRWE	RESERVED														LPSC	DRWE	RESERVED														LPS

Bits	Field Name	Description	Type	Reset
31	CRWE	Clear remote wake-up enable. Write 0x0: no effect. Write 0x1: clears the device remote wake-up enable bit.	W	0x0
30:17	RESERVED	Reserved	R	0x0000
16	LPSC	Local power status change. Always reads 0x0: The Root Hub does not support the local power status feature Write 0x0: no effect. Write 0x1: Sets port power status bits for all ports, if power switching mode is 0. Sets port power status bits for ports with their corresponding port power control mask bits cleared if power switching mode is 1.	RW	0x0
15	DRWE	Device remote wake-up enable. Enables a connect status change event as a resume event, causing a USB suspend to USB resume state transition and sets the resume detected interrupt status bit. Read 0x1: connect status change is a remote wake-up event. Read 0x0: connect status change is not a remote wake-up event. Write 0x0: no effect. Write 0x1: sets the device remote wake-up enable bit.	RW	0x0
14:1	RESERVED	Reserved	R	0x0000
0	LPS	Local power status. Always reads 0x0. Write 0x0: no effect. Write 0x1: When in global power mode (power switching mode = 0), turns off power to all ports. If in per-port power mode (power switching mode = 1), turns of power to those ports whose corresponding port power control mask bit is 0.	RW	0x0

20.2.6.6.22 HCRHPORTSTATUS_1

Table 20-186. HCRHPORTSTATUS_1

Address Offset	0x0000 0054		
Physical Address	0x4806 4454	**Instance**	OHCI
Description	HC Port 1 Status and Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PRSC	RESERVED	PSSC	PESC	CSC	RESERVED							LSDA_CPP	PPS_SPP	RESERVED			PRS_SPR	RESERVED	PSS_SPS	PES_SPE	CCS_CPE

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	PRSC	Port 1 reset status change. This bit is set when the Port 1 port reset status bit has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
19	RESERVED	Reserved	R	0x000
18	PSSC	Port 1 suspend status change. This bit is set when the Port1 port suspend status has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
17	PESC	Port 1 enable status change. This bit is set when the Port1 port enable status has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
16	CSC	Port 1 connect status change. This bit is set when the Port1 port current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, this bit is set. Write 0x0: no effect. Write 0x1: clears this bit. Note: If the DR bit HCRHDESCRIPTORB[1] is set, this bit is set only after a root hub reset to inform the system that the device is attached.	RW	0x0
15:10	RESERVED	Reserved	R	0x00
9	LSDA_CPP	Port 1 low-speed device attached/clear port power. This bit is valid only when port 1 current connect status is 1. Read 0x0: a full-speed device is attached to port 1. Read 0x1: a low-speed device is attached to port 1. Write 0x0: no effect. Write 0x1: clears the port 1 port power status.	RW	0x0
8	PPS_SPP	Port 1 port power status/set port power. Read 0x0: port 1 power is enabled. Read 0x1: port 1 power is not enabled. Write 0x0: no effect. Write 0x1: sets the port 1 port power status bit.	RW	0x0
7:5	RESERVED	Reserved	R	0x0
4	PRS_SPR	Port 1 port reset status/set port reset. Read 0x0: USB reset is not being sent to port 1. Read 0x1: port 1 is signaling the USB reset. Write 0x0: no effect. Write 0x1: sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1.	RW	0x0
3	RESERVED	Reserved	R	0x000

Bits	Field Name	Description	Type	Reset
2	PSS_SPS	Port 1 port suspend status/set port suspend. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. Write 0x0: no effect. Read 0x0: port 1 is not in the USB suspend state. Read 0x1: port 1 is in the USB suspend state or is in the resume sequence. Write 0x1: If port 1 current connect status is 1, sets the port 1 port suspend status bit and places port 1 in USB suspend state. If current connect status is 0, sets instead connect status change to inform the USB host controller driver of an attempt to suspend a disconnected port.	RW	0x0
1	PES_SPE	Port 1 port enable status/set port enable. This bit is automatically set at completion of port 1 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. Read 0x0: port 1 is not enabled. Read 0x1: port 1 is enabled. Write 0x0: no effect. Write 0x1: When port 1 current connect status is 1 sets the port 1 port enable status bit. When port 1 current status is 0 has no effect.	RW	0x0
0	CCS_CPE	Port 1 current connection status/clear port enable. Read 0x0: no USB device is attached to port 1. Read 0x1: port 1 currently has a USB device attached. Write 0x0: no effect. Write 0x1: clears the port 1 port enable bit. Note: This bit is set to 1 if the DR bit HCRHDESCRIPTORB[1] is set to indicate a non-removable device on port 1.	RW	0x0

20.2.6.6.23 HCRHPORTSTATUS_2

Table 20-187. HCRHPORTSTATUS_2

Address Offset	0x0000 0058		
Physical Address	0x4806 4458	**Instance**	OHCI
Description	HC Port 2 Status and Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PRSC	RESERVED	PSSC	PESC	CSC	RESERVED							LSDA_CPP	PPS_SPP	RESERVED			PRS_SPR	RESERVED	PSS_SPS	PES_SPE	CCS_CPE

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	PRSC	Port 2 reset status change. This bit is set when the Port 2 port reset status bit has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
19	RESERVED	Reserved	R	0x000
18	PSSC	Port 2 suspend status changed. This bit is set when the Port 2 port suspend status has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
17	PESC	Port 2 enable status change. This bit is set when the Port 2 port enable status has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
16	CSC	Port 2 connect status change. This bit is set when the Port 2 port current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, this bit is set. Write 0x0: no effect. Write 0x1: clears this bit. Note: If the DR bit HCRHDESCRIPTORB[1] is set, this bit is set only after a root hub reset to inform the system that the device is attached.	RW	0x0
15:10	RESERVED	Reserved	R	0x00
9	LSDA_CPP	Port 2 low-speed device attached/clear port power. This bit is valid only when port 2 current connect status is 1. Read 0x0: a full-speed device is attached to port 2. Read 0x1: a low-speed device is attached to port 2. Write 0x0: no effect. Write 0x1: clears the port 2 port power status.	RW	0x0
8	PPS_SPP	Port 2 port power status/set port power. Read 0x0: port 2 power is enabled. Read 0x1: port 2 power is not enabled. Write 0x0: no effect. Write 0x1: sets the port 2 port power status bit.	RW	0x0
7:5	RESERVED	Reserved	R	0x0
4	PRS_SPR	Port 2 port reset status/set port reset. Read 0x0: USB reset is not being sent to port 2. Read 0x1: port 2 is signaling the USB reset. Write 0x0: no effect. Write 0x1: sets the port 2 port reset status bit and causes the USB host controller to begin signaling USB reset to port 2.	RW	0x0
3	RESERVED	Reserved	R	0x000

Bits	Field Name	Description	Type	Reset
2	PSS_SPS	Port 2 port suspend status/set port suspend. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. Write 0x0: no effect. Read 0x0: port 2 is not in the USB suspend state. Read 0x1: port 2 is in the USB suspend state or is in the resume sequence. Write 0x1: If port 2 current connect status is 1, sets the port 2 port suspend status bit and places port 2 in USB suspend state. If current connect status is 0, sets instead connect status change to inform the USB host controller driver of an attempt to suspend a disconnected port.	RW	0x0
1	PES_SPE	Port 2 port enable status/set port enable. This bit is automatically set at completion of port 2 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. Read 0x0: port 2 is not enabled. Read 0x1: port 2 is enabled. Write 0x0: no effect. Write 0x1: When port 2 current connect status is 1 sets the port 2 port enable status bit. When port 2 current status is 0 has no effect.	RW	0x0
0	CCS_CPE	Port 2 current connection status/clear port enable. Read 0x0: no USB device is attached to port 2. Read 0x1: port 2 currently has a USB device attached. Write 0x0: no effect. Write 0x1: clears the port 2 port enable bit. Note: This bit is set to 1 if the DR bit HCRHDESCRIPTORB[1] is set to indicate a non-removable device on port 2.	RW	0x0

20.2.6.6.24 HCRHPORTSTATUS_3

Table 20-188. HCRHPORTSTATUS_3

Address Offset	0x0000 005C		
Physical Address	0x4806 445C	**Instance**	OHCI
Description	HC Port 3 Status and Control Register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											PRSC	RESERVED	PSSC	PESC	CSC	RESERVED							LSDA_CPP	PPS_SPP	RESERVED			PRS_SPR	RESERVED	PSS_SPS	PES_SPE	CCS_CPE

Bits	Field Name	Description	Type	Reset
31:21	RESERVED	Reserved	R	0x000
20	PRSC	Port 3 reset status change. This bit is set when the Port 3 port reset status bit has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
19	RESERVED	Reserved	R	0x000
18	PSSC	Port 3 suspend status change. This bit is set when the Port 3 port suspend status has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
17	PESC	Port 3 enable status change. This bit is set when the Port 3 port enable status has changed. Write 0x0: no effect. Write 0x1: clears this bit.	RW	0x0
16	CSC	Port 3 connect status change. This bit is set when the Port 3 port current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, this bit is set. Write 0x0: no effect. Write 0x1: clears this bit. Note: If the DR bit HCRHDESCRIPTORB[1] is set, this bit is set only after a root hub reset to inform the system that the device is attached.	RW	0x0
15:10	RESERVED	Reserved	R	0x00
9	LSDA_CPP	Port 3 low-speed device attached/clear port power. This bit is valid only when port 3 current connect status is 1. Read 0x0: a full-speed device is attached to port 3. Read 0x1: a low-speed device is attached to port 3. Write 0x0: no effect. Write 0x1: clears the port 3 port power status.	RW	0x0
8	PPS_SPP	Port 3 power status/set port power. Read 0x0: port 3 power is enabled. Read 0x1: port 3 power is not enabled. Write 0x0: no effect. Write 0x1: sets the port 3 port power status bit.	RW	0x0
7:5	RESERVED	Reserved	R	0x0
4	PRS_SPR	Port 3 reset status/set port reset. Read 0x0: USB reset is not being sent to port 3. Read 0x1: port 3 is signaling the USB reset. Write 0x0: no effect. Write 0x1: sets the port 3 port reset status bit and causes the USB host controller to begin signaling USB reset to port 3.	RW	0x0
3	RESERVED	Reserved	R	0x000

Bits	Field Name	Description	Type	Reset
2	PSS_SPS	Port 3 port suspend status/set port suspend. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. Write 0x0: no effect. Read 0x0: port 3 is not in the USB suspend state. Read 0x1: port 3 is in the USB suspend state or is in the resume sequence. Write 0x1: If port 3 current connect status is 1, sets the port 3 port suspend status bit and places port 3 in USB suspend state. If current connect status is 0, sets instead connect status change to inform the USB host controller driver of an attempt to suspend a disconnected port.	RW	0x0
1	PES_SPE	Port 3 enable status/set port enable. This bit is automatically set at completion of port 3 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed. Read 0x0: port 3 is not enabled. Read 0x1: port 3 is enabled. Write 0x0: no effect. Write 0x1: When port 3 current connect status is 1 sets the port 3 port enable status bit. When port 3 current status is 0 has no effect.	RW	0x0
0	CCS_CPE	Port 3 current connection status/clear port enable. Read 0x0: no USB device is attached to port 3. Read 0x1: port 3 currently has a USB device attached. Write 0x0: no effect. Write 0x1: clears the port 3 port enable bit. Note: This bit is set to 1 if the DR bit HCRHDESCRIPTORB[1] is set to indicate a non-removable device on port 3.	RW	0x0

20.2.6.7 EHCI Register Descriptions

20.2.6.7.1 HCCAPBASE

Table 20-189. HCCAPBASE

Address Offset	0x0000 0000		
Physical Address	0x4806 4800	**Instance**	EHCI
Description	Host Controller Capability register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HCIVERSION																RESERVED								CAPLENGTH							

Bits	Field Name	Description	Type	Reset
31:16	HCIVERSION	Interface version number It contains a BCD encoding of the EHCI revision number supported by this host controller. [7:4] Major revision [3:0] Minor revision	R	0x0100
15:8	RESERVED	Reserved.	R	0x00
7:0	CAPLENGTH	Capability register length.	R	0x10

20.2.6.7.2 HCSPARAMS

Table 20-190. HCSPARAMS

Address Offset	0x0000 0004		
Physical Address	0x4806 4804	**Instance**	EHCI
Description	Host Controller Structural Parameters		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												RESERVED			P_INDICATOR	N_CC				N_PCC				PRR	RESERVED		PPC	N_PORTS			

Bits	Field Name	Description	Type	Reset
31:20	RESERVED	Reserved	R	0x000
19:17	RESERVED	Reserved	R	0x0
16	P_INDICATOR	Port indicator support indication This bit indicates whether the ports support port indicator control. 0x1: The port status and control registers include a read/write field for controlling the state of the port indicator.	R	0
15:12	N_CC	Number of Companion Controllers This field indicates the number of companion controllers associated with this USB 2.0 host controller. 0x0: There are no companion host controllers. Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports. Others: there are companion USB 1.1 host controller(s). Port-ownership hand-off is supported. High-, full-, and low-speed devices are supported on the host controller root ports.	R	0x1
11:8	N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per companion host controller. It is used to indicate the port routing configuration to system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2, then N_PCC can have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. The number in this field must be consistent with N_PORTS and N_CC.	R	0x3
7	PRR	Port Routing Rules The first N_PCC ports are routed to the lowest-numbered function companion host controller, the next N_PCC ports are routed to the next lowest-function companion controller, and so on.	R	0
6:5	RESERVED	Reserved	R	0x0
4	PPC	Port Power control This field indicates whether the host controller implementation includes port power control. 0x0: The ports do not have port power switches. 0x1: The ports have port power switches.	R	1
3:0	N_PORTS	Number of downstream ports This field specifies the number of physical downstream ports implemented on this host controller.	R	0x3

20.2.6.7.3 HCCPARAMS

Table 20-191. HCCPARAMS

Address Offset	0x0000 0008		
Physical Address	0x4806 4808	**Instance**	EHCI
Description	Host Controller Capability Parameters		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																EECP								IST				RESERVED	ASPC	PFLF	BIT64AC

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved	R	0x0000
15:8	EECP	EHCI Extended Capabilities Pointer This field indicates the existence of a capabilities list. 0x0: No extended capabilities are implemented. Others: The offset in PCI configuration space of the first EHCI extended capability.	R	0x00
7:4	IST	Isochronous Scheduling Threshold This field indicateswhere software can reliably update the isochronous schedule in relation to the current position of the executing host controller. The host controller can hold 1 microframe of isochronous data structures before flushing the state.	R	0x1
3	RESERVED	Reserved	R	0
2	ASPC	Asynchronous Schedule Park Capability 0x1: The host controller supports the park feature for high-speed queue heads in the asynchronous schedule. The feature can be disabled or enabled and set to a specific level by using the USBHOST.USBCMD[11]ASPME and the USBHOST.USBCMD[9:8] ASPMC fields.	R	1
1	PFLF	Programmable Frame List Flag 0x0: System software must use a frame list length of 1024 elements with this host controller. 0x1: System software can specify and use a smaller frame list and configure the host controller via the USBHOST.USBCMD[3:2] FLS field. The frame list must always be aligned on a 4K-page boundary.	R	1
0	BIT64AC	64-bit addressing capability This field documents the addressing range capability of this implementation. 0x0: Data structures using 32-bit address memory pointers 0x1: Data structures using 64-bit address memory pointers	R	0

20.2.6.7.4 USBCMD

Table 20-192. USBCMD

Address Offset	0x0000 0010		
Physical Address	0x4806 4810	**Instance**	EHCI
Description	USB Command		
Type	RW		

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12	11	10	9 8	7	6	5	4	3 2	1	0
RESERVED	ITC	RESERVED	ASPME	RESERVED	ASPMC	LHCR	IAAD	ASE	PSE	FLS	HCR	RS

Bits	Field Name	Description	Type	Reset
31:24	RESERVED	Reserved	R	0x00
23:16	ITC	Interrupt Threshold Control This field is used by the system software to select the maximum rate at which the host controller issues interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. 0x00: Reserved 0x01: 1 microframe 0x02: 2 microframes 0x04: 4 microframes 0x08: 8 microframes (default, equates to 1 ms) 0x10: 16 microframes (2 ms) 0x20: 32 microframes (4 ms) 0x40: 64 microframes (8 ms) Others: Undefined	RW	0x08
15:12	RESERVED	Reserved	R	0x0
11	ASPME	Asynchronous Schedule Park Mode Enable 0x0: Park mode is disabled. 0x1: Park mode is enabled.	RW	1
10	RESERVED	Reserved	R	0
9:8	ASPMC	Asynchronous Schedule Park Mode Count It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the asynchronous schedule before continuing traversal of the asynchronous schedule. Valid values are 0x1 to 0x3. **The software must not write a 0 to this bit when Park Mode Enable is a 1 as this may result in undefined behavior.**	RW	0x3
7	LHCR	Light Host Controller Reset It allows the driver to reset the EHCI controller without affecting the state of the ports or the relationship to the companion host controllers. Read 0x0: Light host controller reset is complete and it is safe for host software to reinitialize the host controller. Read 0x1: Light host controller reset is still ongoing.	RW	0

Bits	Field Name	Description	Type	Reset
6	IAAD	Interrupt on Asynchronous Advance Doorbell This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Write 0x1: Ring the doorbell. Software should not write a 1 to this bit when the asynchronous schedule is disabled. Doing so may yield undefined results.	RW	0
5	ASE	Asynchronous Schedule Enable This bit controls whether the host controller skips processing the asynchronous schedule. 0x0: Do not process the asynchronous schedule. 0x1: Use the USBHOST.ASYNCLISTADDR register to access the asynchronous schedule.	RW	0
4	PSE	Periodic Schedule Enable This bit controls whether the host controller skips processing the periodic schedule. 0x0: Do not process the periodic schedule. 0x1: Use the USBHOST.PERIODICLISTBASE register to access the periodic schedule.	RW	0
3:2	FLS	Frame List Size This field specifies the size of the frame list. The size of the frame list controls which bits in the frame index register should be used for the frame list vurrent index. 0x0: 1024 elements (4096 bytes) 0x1: 512 elements (2048 bytes) 0x2: 256 elements (1024 bytes), for resource-constrained environments 0x3: Reserved	RW	0
1	HCR	Host Controller Reset This control bit is used by software to reset the host controller. Write 0x1: Reset the host controller, the PCI configuration registers are not affected by this reset and all operational registers are set to their initial values. This bit is set to 0 by the host controller when the reset process is complete.	W	0
0	RS	Run/stop 0x1: Run, the host controller proceeds with execution of the schedule. The host controller continues execution as long as this bit is set to 1. 0x0: Stop, the host controller completes the current and any actively pipelined transactions on the USB and then halts.	RW	0

20.2.6.7.5 USBSTS

Table 20-193. USBSTS

Address Offset	0x0000 0014		
Physical Address	0x4806 4814	**Instance**	EHCI
Description	USB status		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																ASS	PSS	REC	HCH	RESERVED				IAA	HSE	FLR	PCD	USBEI	USBI		

Bits	Field Name	Description	Type	Reset
31:16	RESERVED	Reserved	R	0x0000
15	ASS	Asynchronous Schedule Status The bit reports the current real status of the asynchronous schedule. 0x0: The status of the asynchronous schedule is disabled. 0x1: The status of the asynchronous schedule is enabled.	R	0
14	PSS	Periodic Schedule Status The bit reports the current real status of the periodic schedule. 0x0: The status of the periodic schedule is disabled. 0x1: The status of the periodic schedule is enabled.	R	0
13	REC	Reclamation It is used to detect an empty asynchronous schedule.	R	0
12	HCH	Host Controller Halted This bit is a 0 whenever the USBHOST.USBCMD[0] RS bit is a 1. The host controller sets this bit to 1 after it has stopped executing as a result of the RS bit being set to 0, either by software or by the host controller hardware.	R	1
11:6	RESERVED	Reserved	R	0x00
5	IAA	Interrupt on Async Advance System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a 1 in the USBHOST.USBCMD[6] IAAD bit. This status bit indicates the assertion of that interrupt source.	RW	0
4	HSE	Host System Error The host controller sets this bit to 1 when a serious error occurs during a host system access involving the host controller module.	RW	0
3	FLR	Frame List Rollover The host controller sets this bit to 1 when the USBHOST.FRINDEX rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size.	RW	0
2	PCD	Port Change Detect The host controller sets this bit to 1 when any port for which the USBHOST.PORTSC_i[13] PO bit is set to 0 has a change bit transition from a 0 to a 1 or a USBHOST.PORTSC_i[6] FPR bit transition from a 0 to a 1. This bit is also set as a result of the USBHOST.PORTSC_i[1] CSC bit being set to 1 after system software has relinquished ownership of a connected port by writing a 1 to a USBHOST.PORTSC_i[13] PO bit.	RW	0
1	USBEI	USB Error Interrupt The host controller sets this bit to 1 when completion of a USB transaction results in an error condition.	RW	0
0	USBI	USB Interrupt	RW	0

Bits	Field Name	Description	Type	Reset
		The host controller sets this bit to 1 on completion of a USB transaction, which results in the retirement of a transfer descriptor that had its IOC bit set. The host controller also sets this bit to 1 when a short packet is detected (actual number of bytes received was less than the expected number of bytes).		

20.2.6.7.6 USBINTR

Table 20-194. USBINTR

Address Offset	0x0000 0018		
Physical Address	0x4806 4818	**Instance**	EHCI
Description	USB interrupt enable		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								IAAE	HSEE	FLRE	PCIE	USBEIE	USBIE		

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved	R	0x0000000
5	IAAE	Interrupt on Async Advance Enable 0x1: When the USBHOST.USBSTS[5] IAA bit is 1, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHOST.USBSTS[5] IAA bit.	RW	0
4	HSEE	Host System Error Enable 0x1: When the USBHOST.USBSTS[4] HSE bit is 1, the host controller issues an interrupt. The interrupt is acknowledged by software clearing the USBHOST.USBSTS[4] HSE bit.	RW	0
3	FLRE	Frame List Rollover Enable 0x1: When the USBHOST.USBSTS[3] FLR bit is 1, the host controller issues an interrupt. The interrupt is acknowledged by software clearing the USBHOST.USBSTS[3] FLR bit.	RW	0
2	PCIE	Port Change Interrupt Enable 0x1: When the USBHOST.USBSTS[2] PCD bit is 1, the host controller issues an interrupt. The interrupt is acknowledged by software clearing the USBHOST.USBSTS[3] FLR bit.	RW	0
1	USBEIE	USB Error Interrupt Enable 0x1: When the USBHOST.USBSTS[1] USBEI bit is 1, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHOST.USBSTS[1] USBEI bit.	RW	0
0	USBIE	USB Interrupt Enable 0x1: When the USBHOST.USBSTS[0] USBI bit is 1, the host controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHOST.USBSTS[0] USBI bit.	RW	0

20.2.6.7.7 FRINDEX

Table 20-195. FRINDEX

Address Offset	0x0000 001C		
Physical Address	0x4806 481C	**Instance**	EHCI
Description	USB frame index		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								FI							

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Reserved	R	0x00000
13:0	FI	Frame index The value in this register is incremented at the end of each time frame.	RW	0x0000

20.2.6.7.8 CTRLDSSEGMENT

Table 20-196. CTRLDSSEGMENT

Address Offset	0x0000 0020		
Physical Address	0x4806 4820	**Instance**	EHCI
Description	4G segment selector		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDSS																															

Bits	Field Name	Description	Type	Reset
31:0	CDSS	This 32-bit register corresponds to the most significant address bits [63:32] for all EHCI data structures.	R	0x00000000

20.2.6.7.9 PERIODICLISTBASE

Table 20-197. PERIODICLISTBASE

Address Offset	0x0000 0024		
Physical Address	0x4806 4824	**Instance**	EHCI
Description	Frame list base address		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAL																				RESERVED											

Bits	Field Name	Description	Type	Reset
31:12	BAL	Base Address (Low) These bits correspond to memory address signals.	RW	0x00000
11:0	RESERVED	Reserved	R	0x000

20.2.6.7.10 ASYNCLISTADDR

Table 20-198. ASYNCLISTADDR

Address Offset	0x0000 0028		
Physical Address	0x4806 4828	**Instance**	EHCI
Description	Next asynchronous list address		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPL																											RESERVED			

Bits	Field Name	Description	Type	Reset
31:5	LPL	Link Pointer Low It contains the address of the next asynchronous queue head to be executed.	RW	0x0000000
4:0	RESERVED	Reserved.	R	0x00

20.2.6.7.11 CONFIGFLAG

Table 20-199. CONFIGFLAG

Address Offset	0x0000 0050		
Physical Address	0x4806 4850	**Instance**	EHCI
Description	Configured flag register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															CF

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved.	R	0x00000000
0	CF	Configure Flag This bit controls the default port-routing control logic. 0x0: Port routing control logic default-routes each port to an implementation dependent classic host controller. 0x1: Port routing control logic default-routes all ports to this host controller.	RW	0x0

20.2.6.7.12 PORTSC_i

Table 20-200. PORTSC_i

Address Offset	0x0000 0054 + (0x04 * i)	**Index**	i = 0 to 2
Physical Address	0x4806 4854 + (0x04 * i)	**Instance**	EHCI
Description	Port Status/Control		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										WDE	WCE	PTC				PIC		PO	PP	LS		RESERVED	PR	SUS	FPR	RESERVED		PEDC	PED	CSC	CCS

Bits	Field Name	Description	Type	Reset
31:22	RESERVED	Reserved	R	0x000
21	WDE	Wake on Disconnect Enable This field is 0 if the PP bit is 0. Write 0x1: Enables the port to be sensitive to device disconnects as wake-up events.	RW	0
20	WCE	Wake on Connect Enable This field is 0 if the PP bit is 0. Write 0x1: Enables the port to be sensitive to device connects as wake-up events.	RW	0
19:16	PTC	Port Test Control The port is operating in specific test modes as indicated by the specific value. The encoding of the test mode bits are: 0x0: Test mode not enabled 0x1: Test J_STATE 0x2: Test K_STATE 0x3: Test SE0_NAK 0x4: Test Packet 0x5: Test FORCE_ENABLE Others: Reserved	RW	0x0
15:14	PIC	Port Indicator Control (not implemented)	R	0x0
13	PO	Port Owner This bit unconditionally goes to a 0x0 when the USBHOST.CONFIGFLAG[0] CF bit makes a 0 to 1 transition. This bit unconditionally goes to 0 whenever the USBHOST.CONFIGFLAG[0] CF bit is 0. 0x1: A companion host controller owns and controls the port.	RW	1
12	PP	Port Power The function of this bit depends on the value of the USBHOST.HCSPARAMS[4] PPC bit. The behavior is as follows: **PPC** / **PP** / **Operation** 0x0 / 0x1 / Host controller does not have port power. control switches. Each port is hard-wired to power. 0x1 / N/A / Host controller has port power control switches. This bit represents the current setting of the switch (0 = Off, 1 = On). When an overcurrent condition is detected on a powered port and the USBHOST.HCSPARAMS[4] PPC bit is a 1, the PP bit in each affected port may be transitioned by the host controller from 1 to 0.	RW	0

Bits	Field Name	Description	Type	Reset
11:10	LS	Line Status These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. This field is valid only when the port enable bit is 0 and the current connect status bit is set to 1. The encoding of the bits is: **Bits[11:10] / USB State / Interpretation** 0x0 / SE0 / Not low-speed device, perform EHCI reset. 0x2 / J-state / Not low-speed device, perform EHCI reset. 0x1 / K-state / Low-speed device, release ownership of port. 0x3 / Undefined / Not low-speed device, perform EHCI reset.	R	0x0
9	RESERVED	Reserved	R	0
8	PR	Port Reset This field is 0 if the PP bit is 0. 0x0: Port is not in reset. 0x1: Port is in reset. Write 0x0: Terminate the bus reset sequence. Write 0x1 when at 0x0: The bus reset sequence is started.	RW	0
7	SUS	Suspend This field is 0 if the PP bit is 0. 0x0 when PED = 0x1: Port enabled 0x1 when PED = 0x1: Port in suspend state When PED = 0x0: Port disabled	RW	0
6	FPR	Force Port Resume This field is 0 if the PP bit is 0. 0x0: No resume (K-state) detected/driven on port 0x1: Resume detected/driven on port	RW	0
5:4	RESERVED	Reserved	R	0x000
3	PEDC	Port Enabled/Disabled Change This field is 0 if the PP bit is 0. Read 0x0: No change. Read 0x1: Port enabled/disabled status has changed. Write 0x1: Clears this bit to 0.	RW	0
2	PED	Port Enabled/Disabled Software cannot enable a port by writing a 1 to this field. The host controller only sets this to 1 when the reset sequence determines that the attached device is a high-speed device. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. This field is 0 if the PP bit is 0. 0x0: Disable 0x1: Enable	RW	0
1	CSC	Connect Status Change Indicates a change has occurred in the port CCS bit. This field is 0 if the PP bit is 0. Read 0x0: No change Read 0x1: Change in current connect status Write 0x1: Clears this bit to 0	RW	0
0	CCS	Current Connect Status This value reflects the current state of the port, and may not correspond directly to the event that caused the CSC bit to be set.	R	0

Bits	Field Name	Description	Type	Reset
		This field is 0 if the PP bit is 0. 0x0: No device is present. 0x1: Device is present on port.		

20.2.6.7.13 INSNREG00

Table 20-201. INSNREG00

Address Offset	0x0000 0090		
Physical Address	0x4806 4890	**Instance**	EHCI
Description	Implementation-specific register #0		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																UFRAME_CNT															EN

Bits	Field Name	Description	Type	Reset
31:14	RESERVED	Reserved.	R	0x00000
13:1	UFRAME_CNT	1-microframe length value, to reduce simulation time SIMULATIONS ONLY, NOT AN ACTUAL REGISTER	RW	0x0000
0	EN	Enable of this register	RW	0x0

20.2.6.7.14 INSNREG01

Table 20-202. INSNREG01

Address Offset	0x0000 0094		
Physical Address	0x4806 4894	**Instance**	EHCI
Description	Implementation-specific register #1		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT_THRESHOLD																IN_THRESHOLD															

Bits	Field Name	Description	Type	Reset
31:16	OUT_THRESHOLD	Programmable output packet buffer threshold, in 32-bit words	RW	0x0020
15:0	IN_THRESHOLD	Programmable input packet buffer threshold, in 32-bit words	RW	0x0020

20.2.6.7.15 INSNREG02

Table 20-203. INSNREG02

Address Offset	0x0000 0098		
Physical Address	0x4806 4898	**Instance**	EHCI
Description	Implementation-specific register #2		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				BUF_DEPTH											

Bits	Field Name	Description	Type	Reset
31:12	RESERVED	Reserved.	R	0x00000
11:0	BUF_DEPTH	Programmable packet buffer depth, in 32-bit words	RW	0x080

20.2.6.7.16 INSNREG03

Table 20-204. INSNREG03

Address Offset	0x0000 009C		
Physical Address	0x4806 489C	**Instance**	EHCI
Description	Implementation-specific register #3		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															BRK_MEM_TRSF

Bits	Field Name	Description	Type	Reset
31:1	RESERVED	Reserved.	R	0x00000000
0	BRK_MEM_TRSF	Break Memory Transfer, in conjunction with INSNREG01 0x0: Disabled 0x1: Enabled	RW	0x1

20.2.6.7.17 INSNREG04

Table 20-205. INSNREG04

Address Offset	0x0000 00A0		
Physical Address	0x4806 48A0	**Instance**	EHCI
Description	Implementation-specific register #4		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											NAK_FIX_DIS	RESERVED	SHORT_PORT_ENUM	HCCPARAMS_WRE	HCSPARAMS_WRE

Bits	Field Name	Description	Type	Reset
31:6	RESERVED	Reserved.	R	0x0000000
5	RESERVED	Must be set to 1 for proper behavior.	RW	0x0
4	NAK_FIX_DIS	Disable NAK fix (don't touch)	RW	0x0
3	RESERVED	Reserved.	R	0x0
2	SHORT_PORT_ENUM	scale down Port enumeration time (debug)	RW	0x0
1	HCCPARAMS_WRE	Make read-only HCCPARAMS register writable (debug)	RW	0x0
0	HCSPARAMS_WRE	Make read-only HCSPARAMS register writable (debug)	RW	0x0

20.2.6.7.18 INSNREG05_UTMI

Table 20-206. INSNREG05_UTMI

Address Offset	0x0000 00A4		
Physical Address	0x4806 48A4	**Instance**	EHCI
Description	Implementation-specific register #5 Register functionality for UTMI mode		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														VBUSY		VPORT				VCONTROLLOADM		VCONTROL				VSTATUS					

Bits	Field Name	Description	Type	Reset
31:18	RESERVED	Reserved.	R	0x0000
17	VBUSY	0x0: vendor interface is done / inactive 0x1: vendor interface is busy	R	0x0
16:13	VPORT	Vendor interface port selection 0x1: Port 1 vendor interface selected 0x2: Port 2 vendor interface selected 0x3: Port 3 vendor interface selected	RW	0x0
12	VCONTROLLOADM	UTMI VcontrolLoadM output (active-low) 0x0: Load Vcontrol value into PHY 0x1: No Action	RW	0x0
11:8	VCONTROL	UTMI Vcontrol output, to be loaded into the PHY	RW	0x0
7:0	VSTATUS	UTMI Vstatus input image, from PHY	R	0x00

20.2.6.7.19 INSNREG05_ULPI

Table 20-207. INSNREG05_ULPI

Address Offset	0x0000 00A4		
Physical Address	0x4806 48A4	**Instance**	EHCI
Description	Implementation-specific register #5 Register functionality for ULPI mode		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTROL	RESERVED			PORTSEL				OPSEL	REGADD							EXTREGADD								WRDATA							

Bits	Field Name	Description	Type	Reset
31	CONTROL	Control/status of the ULPI register access 0x0: ULPI access done 0x1: Start ULPI access	RW	0x0
30:28	RESERVED	Reserved.	R	0x0
27:24	PORTSEL	0x1: Port 1 selected for register access 0x2: Port 2 selected for register access 0x3: Port 3 selected for register access	RW	0x0
23:22	OPSEL	0x2: Register access is Write 0x3: Register access is Read	RW	0x0
21:16	REGADD	ULPI direct register address, for any value different than 0x2F. 0x2F: Triggers an extended address	RW	0x00
15:8	EXTREGADD	Address for extended register accesses. Don't care for direct accesses.	RW	0x00
7:0	WRDATA	Read/Write data of register access	RW	0x00

General-Purpose Interface

This chapter describes the general-purpose interface.

Topic **Page**

21.1 General-Purpose Interface Overview

The general-purpose interface combines six general-purpose input/output (GPIO) banks.

Each GPIO module provides 32 dedicated general-purpose pins with input and output capabilities; thus, the general-purpose interface supports up to 192 (6 x 32) pins.

These pins can be configured for the following applications:

- Data input (capture)/output (drive)
- Keyboard interface with a debounce cell
- Interrupt generation in active mode upon the detection of external events. Detected events are processed by two parallel independent interrupt-generation submodules to support biprocessor operations.
- Wake-up request generation in idle mode upon the detection of external events

These modules do not include pad control (pull up/down control, open-drain feature). For more information, see Chapter 6, *System Control Module*.

21.1.1 Global Features

The GPIO modules include the following global features:

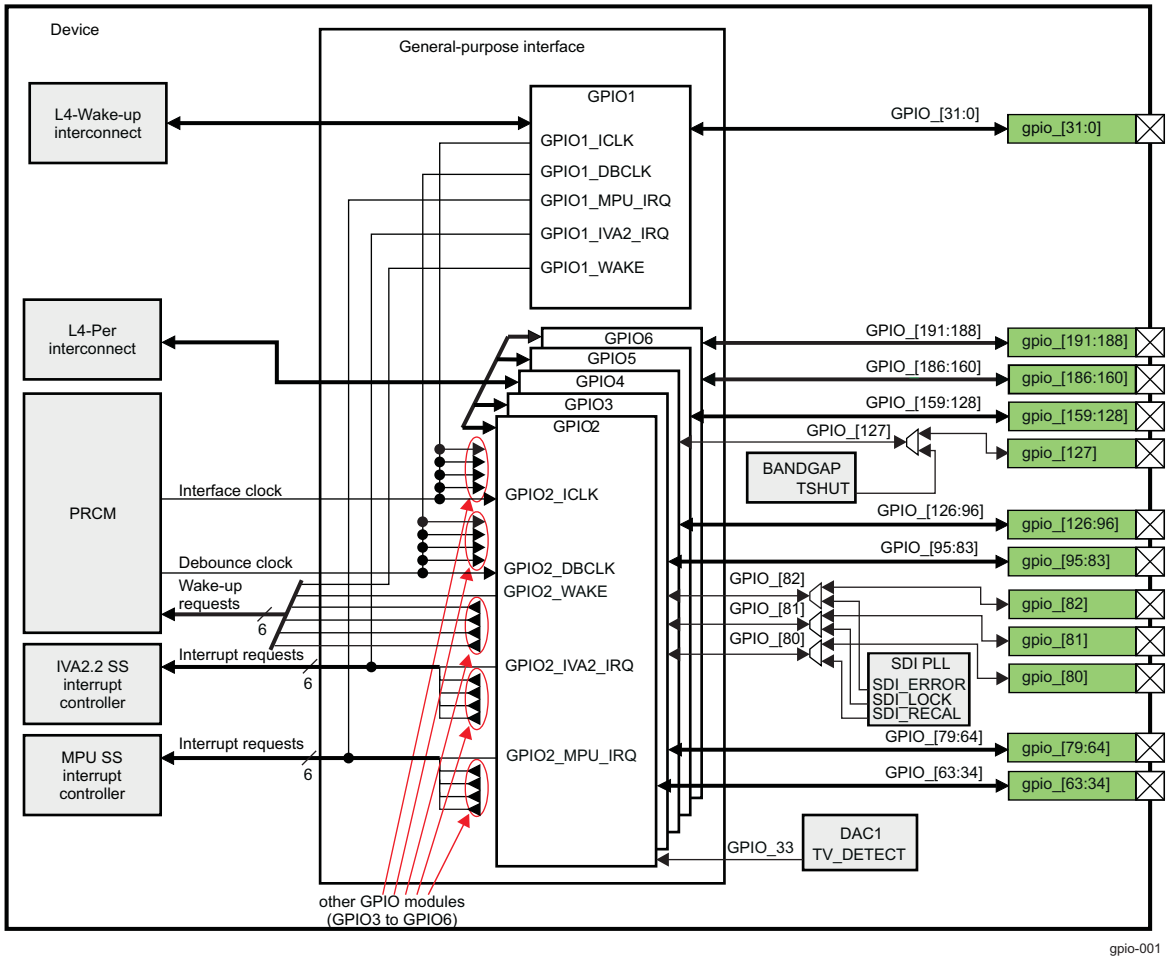
- Synchronous interrupt requests in active mode from each channel are processed by two identical interrupt generation submodules used independently by the imaging video and audio accelerator (IVA2.2) and the microprocessor unit (MPU) subsystems. One of these interrupts is mapped on the IVA2.2 subsystem interrupt controller and the other on the MPU subsystem interrupt controller.
- Asynchronous wake-up requests in idle mode from input channels are merged together to issue one wake-up signal per GPIO module.
- Data input (capture)/output (drive)
- Power management support

The general-purpose interface has 12 interrupt lines (two interrupt lines per GPIO module instance).

Each GPIO module produces a wake-up request signal to the power, reset, and clock management (PRCM) module.

Figure 21-1 shows an overview of the general-purpose interface.

Figure 21-1. General-Purpose Interface Overview



Each channel in the GPIO modules has the following features:

- The GPIOi.GPIO_OE register controls the output capability for each pin.
- The output line level reflects the value written in the GPIOi.GPIO_DATAOUT register through the L4 interconnect.
- The input line can be fed to the GPIO module through an optional and configurable debounce cell. (The debouncing time value is global for all ports of one GPIO module, so up to five different debouncing time values are possible.)
- The input line value is sampled into the GPIOi.GPIO_DATAIN register and can be read through the L4 interconnect.
- In active mode, the input line can be used through level and edge detectors to trigger synchronous interrupts. The edge (rising, falling, or both) or the level (logical 0, logical 1, or both) used can be configured.
- In idle mode, the input line can be used to activate the asynchronous wake-up request (on edge detection: rising edge, falling edge, or both).

The module provides an alternative to the atomic test and set operations for the following registers:

- GPIOi.GPIO_DATAOUT
- GPIOi.GPIO_IRQENABLE1
- GPIOi.GPIO_IRQENABLE2
- GPIOi.GPIO_WAKEUPENABLE

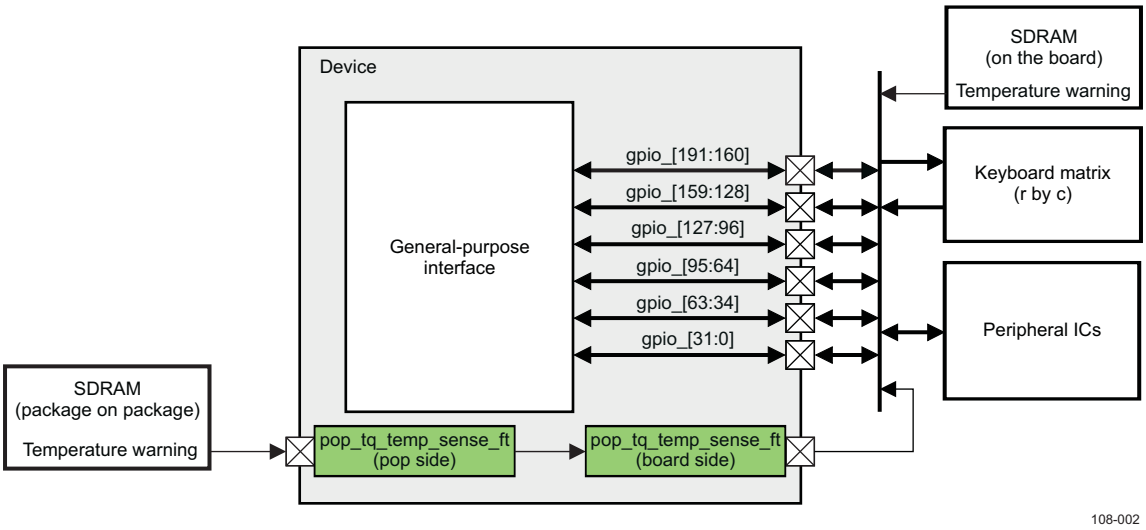
For these registers, the modules implement the set-and-clear protocol register update (see Section 21.5.2, *Set-and-Clear Instructions*).

21.2 General-Purpose Interface Environment

The general-purpose interface combines six GPIO modules for a flexible, user-programmable, general-purpose input/output (I/O) controller. The general-purpose interface implements functions that are not implemented with the dedicated controllers in the device and require simple input and/or output software-controlled signals. The general-purpose interface allows a variety of custom connections and expands the I/O capabilities of the system to the real world.

Figure 21-2 shows a typical application using the general-purpose interface.

Figure 21-2. General-Purpose Interface Typical Application System Overview



NOTE: Temperature Sensing

Most memories provide a temperature sensor to control the auto-refresh duty cycle. The device monitors the temperature of the external memory using the pop_tq_temp_sense_ft ball and a GPIO input. To do this, pop_tq_temp_sense_ft is connected to a GPIO through the customer board. This feature is application-dependent.

CAUTION

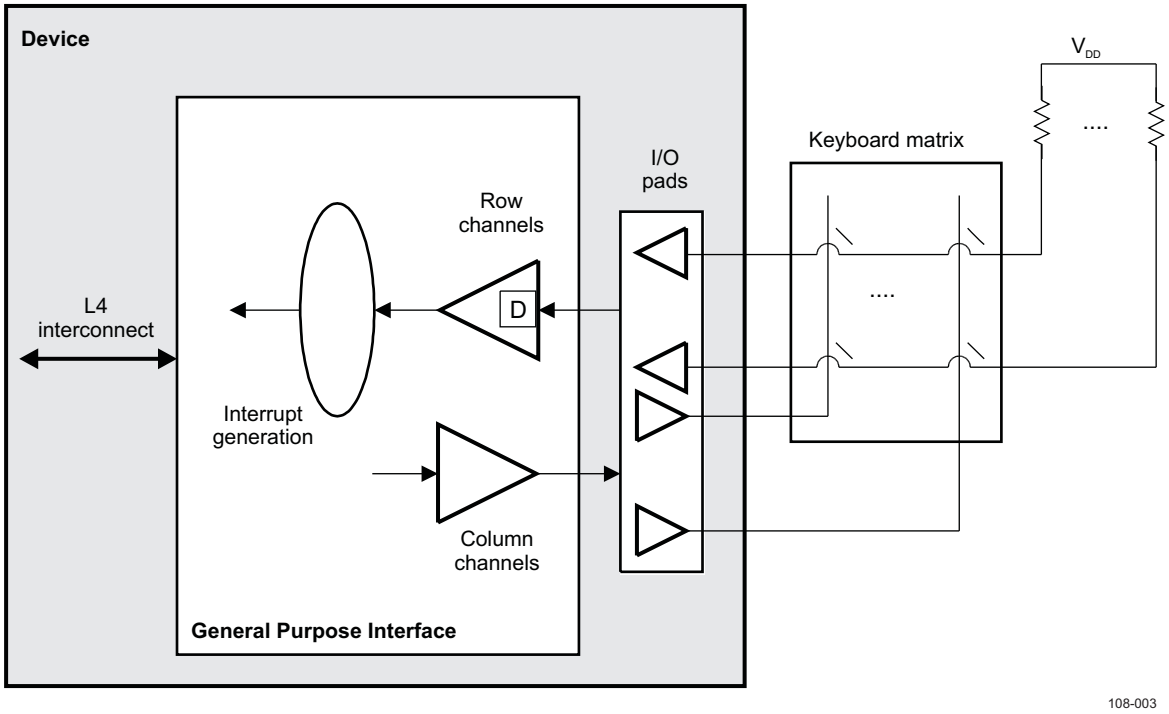
Due to buffer strength, an external serial resistor must be connected to the balls where gpio_120 through gpio_129 are muxed with MMC signals.

The general-purpose interface can physically connect the device to a keyboard matrix and peripheral integrated circuits (ICs).

21.2.1 GPIO as a Keyboard Interface

The general-purpose interface can be used as a keyboard interface. You can dedicate channels based on the keyboard matrix (r * c). Figure 21-3 shows row channels configured as inputs with the input debounce feature enabled. The row channels are driven high with an external pullup. Column channels are configured as outputs and drive a low level.

Figure 21-3. General-Purpose Interface Used as a Keyboard Interface



When a keyboard matrix key is pressed, the corresponding row and column lines are shorted together and a low level is driven on the corresponding row channel. This generates an interrupt based on the proper configuration (see Section 21.5.3, *Interrupt and Wakeup*).

When the keyboard interrupt is received, the processor (MPU and/or IVA2.2 subsystem) can disable the keyboard interrupt and scan the column channels for the key coordinates.

- The scanning sequence has as many states as column channels: For each step in the sequence, the processor drives one column channel low and the others high.
- The processor reads the values of the row channels and thus detects which keys in the column are pressed.

At the end of the scanning sequence, the processor establishes which keys are pressed. The keyboard interface can then be reconfigured in the interrupt waiting state.

21.2.2 General-Purpose Interface Functional Interfaces

21.2.2.1 General-Purpose Interface Pins

Table 21-1 lists the interface pins of the general-purpose interface.

Table 21-1. I/O Pin Description

Signal Name	I/O[1][2]	Description[3][2]	Reset Value
gpio_[31:0]	I/O	GPIO in configuration mode 4.	HiZ
gpio_[186:34]	I/O	GPIO in configuration mode 4.	HiZ
gpio_[191:188]	I/O	GPIO in configuration mode 4.	HiZ

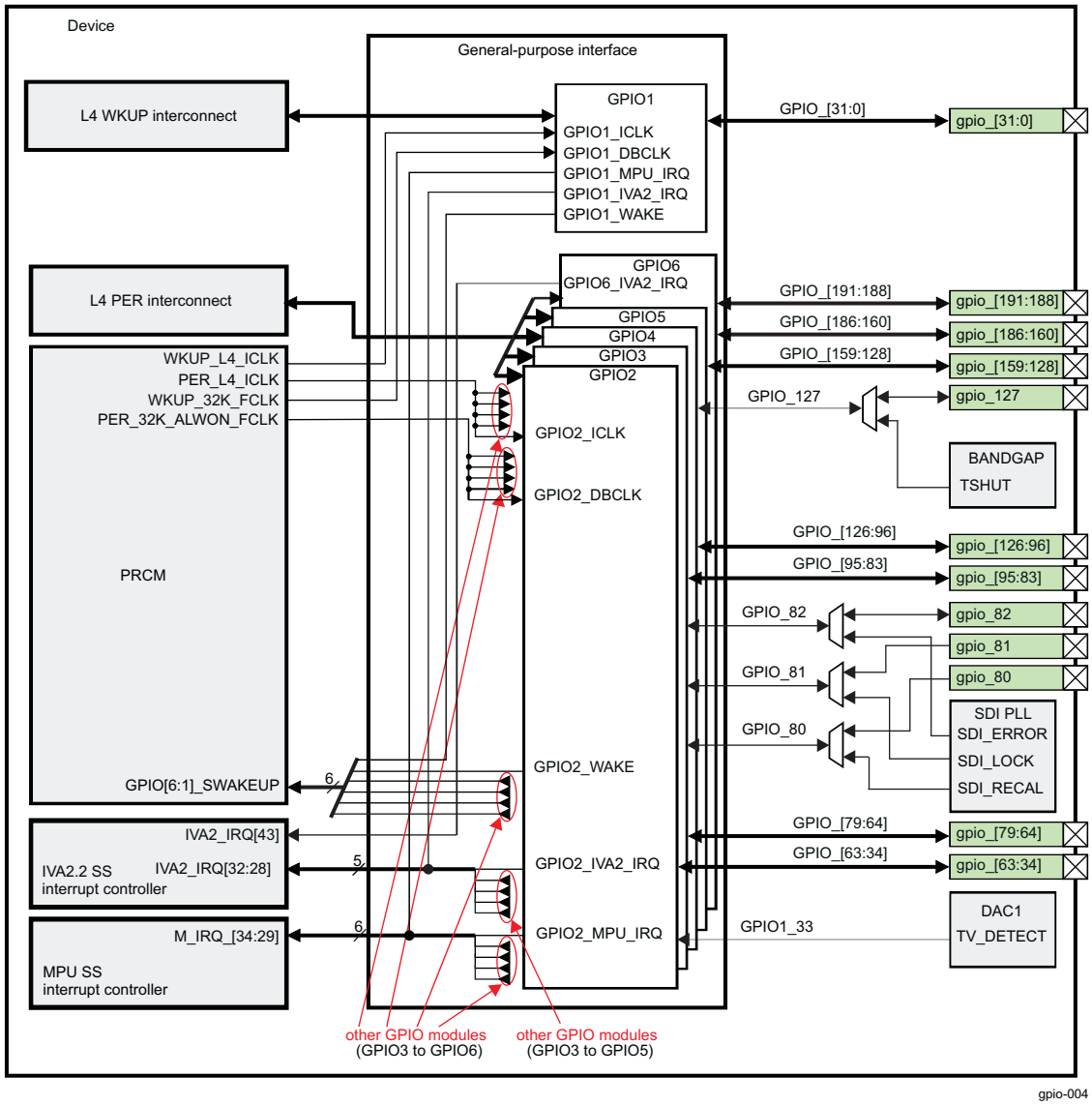
(1) I = Input, O = Output
(2) Some of the pins have special or restricted use. For more information, see Table 21-5.
(3) See Chapter 6, *System Control Module*, for more information about pin configuration modes.

21.3 General-Purpose Interface Integration

21.3.1 Description

Figure 21-4 highlights the general-purpose interface integration in the device.

Figure 21-4. General-Purpose Interface Integration Overview



21.3.1.1 Clocking, Reset, and Power-Management Scheme

21.3.1.1.1 Clocking

Each GPIO module uses two clocks:

- Debounce clock: The 32-KHz debounce clock, GPIOi_DBCLK, (where i = 1, 2, 3, 4, 5, and 6, with one debounce clock per module), comes from the PRCM module and is used for the debounce cell logic (without the corresponding configuration registers). This cell can sample the input line and filters the input level using a programmed delay.

  For GPIO2 to GPIO6, this clock is controlled by the EN_GPIOi (where i = 2 to 6) bit PRCM.CM_FCLKEN_PER (0: disabled, 1: enabled the clock). For GPIO1, this clock is controlled by EN_GPIO1 bit PRCM.CM_FCLKEN_WKUP[3] (0: disabled, 1: enabled the clock) for GPIO1.

- Interface clock: The interface clock, GPIOi_ICLK (where i = 1, 2, 3, 4, 5, and 6), comes from the PRCM module and is used throughout the GPIO module (except within the debounce cell logic). The interface clock clocks the data exchanges between the L4 interconnect and the internal logic. The clock-gating features allow module power consumption to be adapted to the activity.

  For GPIO1, this clock is controlled by the EN_GPIO1 bit PRCM.CM_ICLKEN_WKUP[3] (0: disabled, 1:enabled the clock) and AUTO_GPIO1 bit PRCM.CM_AUTOIDLE_WKUP[3] (enables/disables automatic control of the interface clock). For GPIO2 to GPIO6, this clock is controlled by the EN_GPIOi (where i = 2 to 6) bit PRCM.CM_ICLKEN_PER (0: disabled, 1: enabled the clock) and AUTO_GPIOi (where i = 2 to 6) bit PRCM.CM_AUTOIDLE_PER (enables/disables automatic control of the interface clock). Table 25-2 describes the GPIO module clocks.

Table 21-2. Clocks

Attribute	Frequency	Name	Mapping	Comments
Debounce clock	32 KHz	GPIOi_DBCLK, where I = 2 to 6	PER_32K_ALWON_FCLK	Source is PRCM module.
		GPIO1_DBCLK	WKUP_32K_CLK	
Interface clock	Depends on PRCM registers settings	GPIOi_ICLK, where I = 2 to 6	PER_L4_ICLK	
		GPIO1_ICLK	WKUP_L4_ICLK	

21.3.1.1.2 Reset

The general-purpose interface can be reset by using the domain reset (hardware reset) or by setting a dedicated configuration bit (software reset) in each GPIO module.

- Hardware reset: The GPIO2 to GPIO6 modules are attached to the PER_RST reset domain. The GPIO1 module is attached to the WKUP_RST reset domain.

  The hardware reset has a global reset action on the GPIO modules of the general-purpose interface. All configuration registers and internal logic are reset when it is active (low level). In each GPIO module, the RESETDONE bit GPIOi.GPIO_SYSSTATUS[0] monitors the internal reset status; it is set when the reset completes. For more information, see , *Power, Reset, and Clock Management*.
- Software reset: Each GPIO module has its own software reset using the GPIOi.GPIO_SYSCONFIG[1] SOFTRESET bit (where i = 1, 2, 3, 4, 5, or 6). The software reset has the same effect as the hardware reset signal, but this reset can be applied on one module or more.

  Writing 1 to SOFTRESET bit GPIOi.GPIO_SYSCONFIG[1](where i = 1, 2, 3, 4, 5, or 6) resets the module. Bit value 1 remains until the reset is complete. When the software reset is complete, the GPIOi.GPIO_SYSCONFIG[1] SOFTRESET bit is automatically reset to 0 and has the same effect as the hardware reset. The GPIOi.GPIO_SYSSTATUS[0] RESETDONE is cleared during a software reset. This bit is set to 1 when the software reset is complete.

21.3.1.1.3 Power Domain

The GPIO1 module is attached to the WKUP power domain (see , *Power, Reset, and Clock Management*). This domain is composed of the logic permanently supplied to manage domain power state transitions and detect wake-up events. The WKUP power domain is continuously active. The GPIO2 to GPIO6 modules are attached to the PER power domain (see , *Power, Reset, and Clock Management*). The PER power domain is not active continuously.

21.3.1.1.4 Power Management

21.3.1.1.4.1 Idle Scheme

To save dynamic consumption, an efficient idle scheme is based on the following:

- An efficient local autoclock gating for each module
- The implementation of control sideband signals between the PRCM module and each module

This enhanced idle control allows clocks to be activated and deactivated safely without requiring a complex software management.

The idle mode request, idle acknowledge, and wake-up request (GPIOi_SWAKEUP, where i = 1, 2, 3, 4, 5, and 6) are sideband signals between the PRCM module and the general-purpose interface (see Section 21.3.1.2, *Hardware Requests*).

21.3.1.1.4.2 Operating Modes

The following four operating modes are defined for the modules:

- Active mode: The module runs synchronously on the interface clock; interrupts can be generated based on the configuration and external signals.
- Idle mode: Power-saving mode with the module in a waiting state. The interface clock can be stopped, an interrupt cannot be generated, and a wake-up signal can be generated based on the configuration and external signals.

  If the debounce clock provided by the PRCM module is active, the debounce cell can sample and filter the input to generate a wake-up event. If the debounce clock is inactive, the debounce cell gates all input signals and thus cannot be used.
- Inactive mode: The module has no activity. The interface clock can be stopped, an interrupt cannot be generated, and the wake-up feature is inhibited.
- Disabled mode: The module is not used. The internal clock paths are gated, and an interrupt or wake-up request cannot be generated.

The idle and inactive modes are configured within the module and activated on request by the PRCM module (see , *Power, Reset, and Clock Management*) through sideband signals (see Section 21.3.1.1.4.3, *System Power Management and Wake-up*).

The disabled mode is set by software through a dedicated configuration bit, the GPIOi.GPIO_CTRL[0] DISABLEMODULE bit (0: the module is enabled and clocks are not gated; 1: the module is disabled and clocks are gated). It unconditionally gates the internal clock paths that are not used for the L4 interconnect.

21.3.1.1.4.3 System Power Management and Wake-Up

The PRCM module can require the GPIO modules to be idled for power saving purposes.

The general-purpose interface has six identical idle mode request/acknowledge (handshake) mechanisms with the PRCM module (see Figure 21-4 and Section 21.3.1.2, *Hardware Requests*): one per GPIO module. The general-purpose interface allows the GPIO modules to enter idle mode based on the GPIOi.GPIO_SYSCONFIG[4:3] IDLEMODE field.

The idle acknowledge depends on the configuration and activity of each GPIO module:

- Smart-idle mode (recommended)

  When the GPIO module is configured in smart-idle mode (GPIOi.GPIO_SYSCONFIG[4:3] IDLEMODE field [10]) and receives an idle request from the PRCM module (for GPIO2 to GPIO6: the corresponding bits in the PRCM.CM_FCLKEN_PER and PRCM.CM_ICLKEN_PER registers cleared to 0 or the corresponding bit in the PRCM.CM_AUTOIDLE_PER bit set to 1 and L4 interface clock idle transitions; for GPIO1: PRCM.CM_FCLKEN_ WKUP[3] EN_GPIO1 bit cleared to 0, PRCM.CM_ICLKEN_WKUP[3] EN_GPIO1 bit cleared to 0, or PRCM.CM_AUTOIDLE_WKUP[3] AUTO_ GPIO1 bit set to 1 and L4 interface clock idle transitions), the GPIO module checks for more activity (capture of the input GPIO pins in the GPIOi.GPIO_DATAIN register is complete with no pending interrupt; all interrupt status bits are cleared) ; and there is no access to GPIO.GPIO_DEBOUNCINGTIME register pending to be synchronized.

  Idle acknowledge is then asserted and the module enters in idle-mode. It waits for active system clock gating by the PRCM module (when all peripherals supplied by the same L4 interface clock domain are also ready for idle).

  Idle mode (that is, when the PRCM module gates the interface clock), no interrupt occurs and the module is ready to issue a wake-up request.

  When the expected transition occurs on an enabled GPIO input pin, the GPIO module exits from idle

mode, if the GPIOi.GPIO_SYSCONFIG[2] ENAWAKEUP bit is set to 1 (wake-up capability enabled), and the corresponding bit in the PRCM.PM_WKEN_PER register is also set to 1 for the GPIO2 to GPIO6 modules, and/or the PRCM.PM_WKEN_WKUP[3] EN_GPIO1 bit is also set to 1 for GPIO1.

- Force-idle mode

  When the GPIO module is configured in force-idle mode (GPIOi.GPIO_SYSCONFIG[4:3] IDLEMODE field [00]) and receives an idle request from the PRCM module (for the GPIO2 to GPIO6: the corresponding bits in the PRCM.CM_FCLKEN_PER and PRCM.CM_ICLKEN_PER registers cleared to 0 or the corresponding bit in the PRCM.CM_AUTOIDLE_PER bit set to 1 and L4 interface clock idle transitions; for the GPIO1: PRCM.CM_FCLKEN_ WKUP[3] EN_GPIO1 bit cleared to 0, PRCM.CM_ICLKEN_WKUP[3] EN_GPIO1 bit cleared to 0, or PRCM.CM_AUTOIDLE_WKUP[3] AUTO_ GPIO1 bit set to 1 and L4 interface clock idle transitions), the GPIO module waits unconditionally for active system clock gating by the PRCM module. (This occurs only when all peripherals supplied by the same L4 interface clock domain are also ready for idle.)

  When in idle mode (that is, when the PRCM module gates the interface clock), the module (in inactive mode) has no activity, the interface clock paths are gated, an interrupt cannot be generated, and the wake-up feature is totally inhibited.

- No-idle mode

  When the GPIO module is configured in no-idle mode (GPIOi.GPIO_SYSCONFIG[4:3] IDLEMODE field [01]) and receives an idle request from the PRCM module (for the GPIO2 to GPIO6: the corresponding bits in the PRCM.CM_FCLKEN_PER and PRCM.CM_ICLKEN_PER registers cleared to 0 or the corresponding bit in the PRCM.CM_AUTOIDLE_PER bit set to 1 and L4 interface clock idle transitions; for the GPIO1: PRCM.CM_FCLKEN_ WKUP[3] EN_GPIO1 bit cleared to 0, PRCM.CM_ICLKEN_WKUP[3] EN_GPIO1 bit cleared to 0 or PRCM.CM_AUTOIDLE_WKUP[3] AUTO_ GPIO1 bit set to 1 and L4 interface clock idle transitions), the GPIO module does not go to the idle mode and the idle acknowledge is never sent.

NOTE: The GPIO2 to GPIO6 idle state can be checked by reading the corresponding status bits in the PRCM.CM_IDLEST_PER register (0: active, 1: idle) and is idle only when the GPIO2 to GPIO6 modules are configured in smart-idle mode and have asserted their idle acknowledge.

The GPIO1 idle state can be checked by the PRCM.CM_IDLEST_WKUP[3] ST_GPIO1 bit (0: idle, 1: active) and is idle only when the GPIO1 module is configured in smart-idle mode and has asserted its idle acknowledge.

The GPIO2 to GPIO6 wake-up status can be checked by accessing the corresponding bits in the PRCM.PM_WKST_PER register (read 0: no wakeup occurred; read 1: wakeup occurred; write 1: status bit reset).

The GPIO1 wake-up status can also be checked by the PRCM.PM_WKST_WKUP[3] ST_GPIO1 bit (read 0: no wakeup occurred; read 1: wakeup occurred; write 1: status bit reset).

21.3.1.1.4.4 Module Power Saving

The GPIO module has local power management by internal clock-gating features:

- Internal interface clock gating: The clock for the L4 interconnect logic can be gated when the module is not accessed, if the GPIOi.GPIO_SYSCONFIG[0] AUTOIDLE bit is set. Otherwise, this logic is free-running on the interface clock.
- Clock gating for the input data sample logic: Clock for the input data sample logic can be gated when the data in register is not accessed.
- Clock gating for the event detection logic: EachGPIO module implements four clock groups used for the logic in the synchronous events detection. Each group of eight input GPIO pins has a separate enable signal depending on the edge/level detection register setting. If a group requires no detection, the corresponding clock is gated off (see Section 21.5.1, *Power Saving by Grouping the Edge/Level Detection*). All channels are also gated using a one-out-of-N scheme. N is the GATINGRATIO field GPIOi.GPIO_CTRL[2:1] and can take the values 1 (b00), 2 (0b01), 4 (b10), or 8 (0b11). The interface clock is enabled for this logic one cycle every N cycles. When N is equal to 1, there is no gating and

this logic is free-running on the interface clock. When N is 2, 4, or 8, this logic is running at the equivalent frequency of interface clock frequency divided by N.

- Inactive mode: In inactive mode, all internal clock paths are gated.
- Disabled mode: All internal clock paths not used for the L4 interconnect are gated. The GPIOi.GPIO_CTRL[0] DISABLEMODULE bit controls a clock-gating feature at the module level. When set to 1, this bit forces clock gating for all internal clock paths. Module internal activity is suspended. The L4 interconnect is not affected by this bit.

  The interface clock gating is controlled with the GPIOi.GPIO_SYSCONFIG[0] AUTOIDLE bit, which is used to save power when the module is not used because of the multiplexing configuration selected at the chip level. This bit has precedence over all other internal configuration bits.

21.3.1.2 Hardware Requests

21.3.1.2.1 Interrupt Requests

All interrupt sources (the 32 input GPIO channels) are merged to issue two synchronous interrupt requests in each GPIO module. Thus, the general-purpose interface has 12 interrupt lines (two interrupt lines per GPIO module instance).

Synchronous interrupt request lines 1 and 2 are active depending on their respective interrupt enable 1 and 2 registers (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2).

- Synchronous interrupt request line 1 is mapped on the MPU interrupt controller.
- Synchronous interrupt request line 2 is mapped on the IVA2.2 interrupt controller.

Table 21-3 lists the interrupt lines that are driven out from the general-purpose interface to the MPU subsystem and IVA2.2 subsystem interrupt cont

Table 21-3. Interrupts

Name	Mapping	Comments
GPIO1 Module		
GPIO1_MPU_IRQ	M_IRQ_29	Destination is the MPU subsystem interrupt controller.
GPIO1_IVA2_IRQ	IVA2_IRQ[28]	Destination is the IVA2.2 subsystem interrupt controller.
GPIO2 Module		
GPIO2_MPU_IRQ	M_IRQ_30	Destination is the MPU subsystem interrupt controller.
GPIO2_IVA2_IRQ	IVA2_IRQ[29]	Destination is the IVA2.2 subsystem interrupt controller.
GPIO3 Module		
GPIO3_MPU_IRQ	M_IRQ_31	Destination is the MPU subsystem interrupt controller.
GPIO3_IVA2_IRQ	IVA2_IRQ[30]	Destination is the IVA2.2 subsystem interrupt controller.
GPIO4 Module		
GPIO4_MPU_IRQ	M_IRQ_32	Destination is the MPU subsystem interrupt controller.
GPIO4_IVA2_IRQ	IVA2_IRQ[31]	Destination is the IVA2.2 subsystem interrupt controller.
GPIO5 Module		
GPIO5_MPU_IRQ	M_IRQ_33	Destination is the MPU subsystem interrupt controller.
GPIO5_IVA2_IRQ	IVA2_IRQ[32]	Destination is the IVA2.2 subsystem interrupt controller.
GPIO6 Module		
GPIO6_MPU_IRQ	M_IRQ_34	Destination is the MPU subsystem interrupt controller.
GPIO6_IVA2_IRQ	IVA2_IRQ[43]	Destination is the IVA2.2 subsystem interrupt controller.

21.3.1.2.1.1 Wake-Up Generation

The GPIO1 module of the general-purpose interface is attached to the WKUP power domain (see , *Power, Reset, and Clock Management)* and can wake up the system.

NOTE: The GPIO2 to GPIO6 modules belong to the PER power domain and thus have wake-up system capability only when the PER power domain is active.

All wake-up sources (the 32 input GPIO channels) are merged together to issue a single asynchronous wake-up request in each GPIO module following the expected transition(s) (based on register programming). Each GPIO module generates a wake-up signal to the PRCM module.

NOTE: Only gpio_1, gpio_9, gpio_10, gpio_11, gpio_30, and gpio_31 can be used to generate a direct wake-up event. The other GPIO1 pins cannot be used to generate a direct wake-up event because they are connected to the device I/O pad logic in the CORE power domain (VDD2). When the CORE power domain is off, the I/O pins of the GPIO1 module, which are supplied by VDD2, cannot generate a wake-up event.

The asynchronous wake-up request line is active based on the GPIOi.GPIO_WAKEUPENABLE register (where i = 1, 2, 3, 4, 5, and 6).

CAUTION

The wake-up capabilities of the GPIO2 to GPIO6 modules are operational only when the PER power domain is active.

Table 21-4 shows the wake-up signals mapping.

Table 21-4. Wake-Up Signals

Name	Mapping	Comments
GPIOi_WAKE	GPIOi_SWAKEUP	Where i = 1, 2, 3, 4, 5, and 6. Destination is the PRCM module.

Table 21-5 describes the GPIO channels.

Table 21-5. GPIO Channel Description

Channel Number	Type (1)	Mapping	Wake-Up Feature	Comments
GPIO1 Module				
[31:0]	I/O	gpio_[31:0]	Yes	GPIO (2)
GPIO2 Module				
[0]	I	-	No	Not available on external balls. Read value is always 0.
[1]	I	TV_DETECT	Yes (3)	Internal TV detection signal from the 10-bit composite/luma video DAC1
[31:2]	I/O	gpio_[63:34]	Yes (3)	GPIO (2)
GPIO3 Module				
[15:0]	I/O	gpio_[79:64]	Yes (3)	GPIO (2)
[16]	I/O	gpio_80	Yes (3)	GPIO (2)
	I	SDI_RECAL	Yes (3)	Internal SDI_RECAL signal from the SDI PLL module (4)
[17]	I/O	gpio_81	Yes (3)	GPIO (2)
	I	SDI_LOCK	Yes (3)	Internal SDI_LOCK signal from the SDI PLL module (4)
[18]	I/O	gpio_82	Yes (3)	GPIO (2)

(1) I = Input, O = Output
(2) Configuration mode 4. See Chapter 6, *System Control Module*.
(3) Only when the PER power domain is active
(4) All configuration modes except configuration mode 4. See Chapter 6, *System Control Module*.

Table 21-5. GPIO Channel Description (continued)

Channel Number	Type [1]	Mapping	Wake-Up Feature	Comments
	I	SDI_ERROR	Yes [3]	Internal SDI_ERROR signal from the SDI PLL module [4]
[31:19]	I/O	gpio_[95:83]	Yes [3]	GPIO [2]
GPIO4 Module				
[30:0]	I/O	gpio_[126:96][5]	Yes [3]	GPIO [2]
[31]	I/O	gpio_127	Yes [3]	GPIO [2]
	I	TSHUT	Yes [3]	Internal TSHUT signal from the BANDGAP module for the SRAMs LDOs [4]
GPIO5 Module				
[31:0]	I/O	gpio_[159:128]	Yes [3]	GPIO [2]
GPIO6 Module				
[26:0]	I/O	gpio_[186:160]	Yes [3]	GPIO [2]
[27]	I	-	No	Not available on external balls. Read value is always 0.
[31:28]	I/O	gpio_[191:188]]	Yes [3]	GPIO [2]

[5] All signals are I/O type, except gpio_99, gpio_100, gpio_112, gpio_113, gpio_114, and gpio_115, which are input-only signals.

NOTE: The thermal shutdown comparator output signal (TSHUT) is an output from the BANDGAP module. This signal is high during normal operation and goes low during a thermal shutdown event. When channel 31 of the GPIO4 is not connected to a ball of the device (the corresponding pin is configured in a mode different from the configuration mode 4; see Chapter 6, *System Control Module*, for more information about pin configuration), TSHUT is connected to channel 31 of the GPIO4, and an interrupt can be generated when a high-to-low transition occurs on TSHUT whether or not the interrupt generation for channel 31 of the GPIO4 is correctly configured.

21.4 General-Purpose Interface Functional Description

Figure 21-5 shows the general-purpose interface description.

Figure 21-5. General-Purpose Interface Description

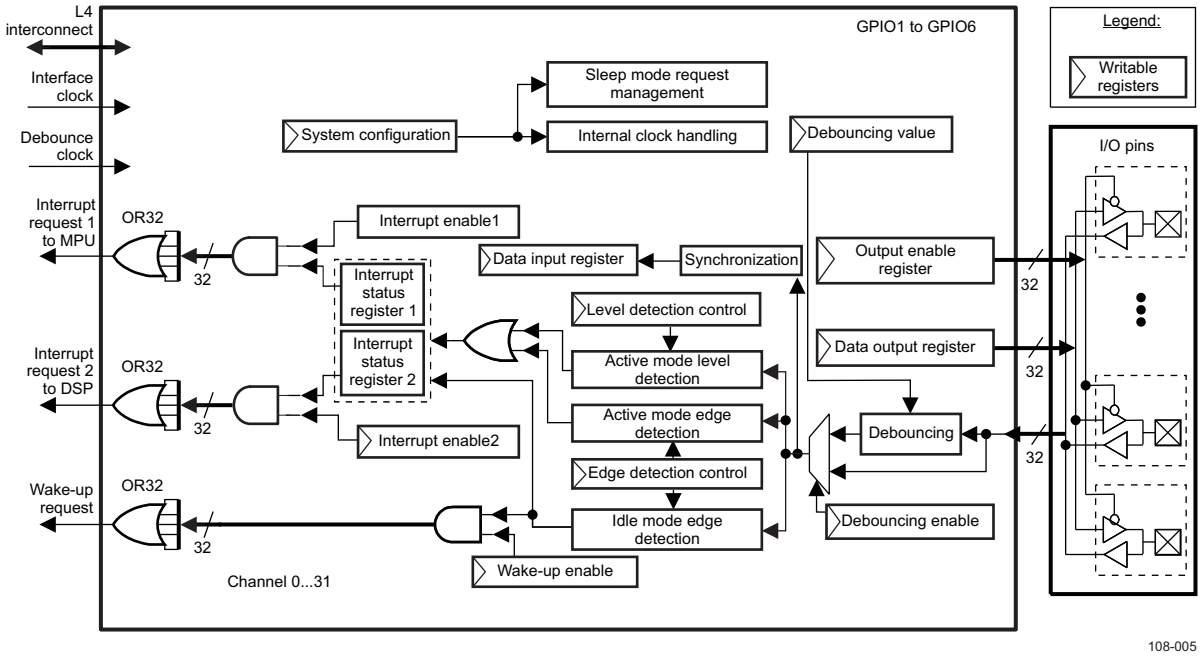
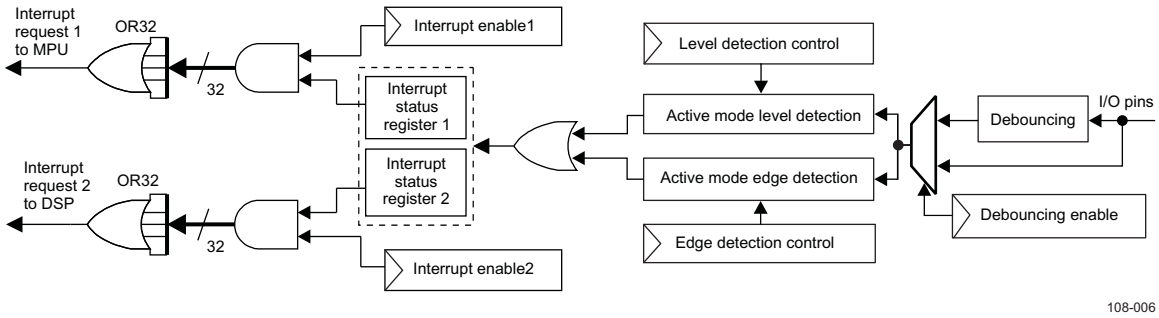


Figure 21-5 details the GPIO modules in the general-purpose interface block diagram with their configuration registers and their main functional paths:

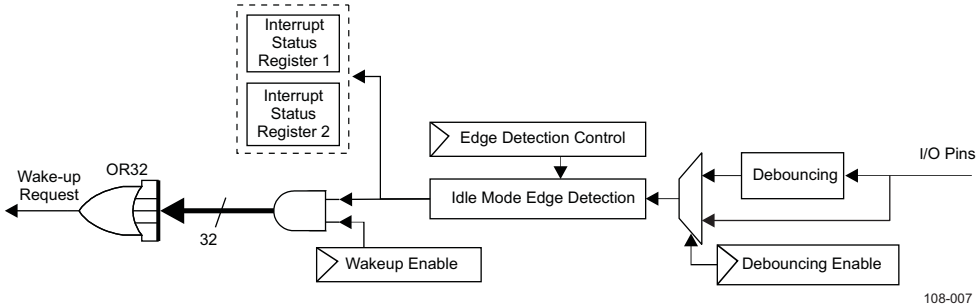
- The synchronous path (for active mode operation) used to generate a synchronous interrupt request on expected event detection on any input GPIO; the synchronous interrupt request lines 1 and 2 are active based on their respective interrupt enable 1 and 2 registers (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2). See Figure 21-6.

Figure 21-6. Synchronous Path



- The asynchronous path (for idle mode operation) used to generate an asynchronous wake-up request on the expected edge detection on any input GPIO; the asynchronous wake-up request line is active based on the wake-up enable register. See Figure 21-7.

Figure 21-7. Asynchronous Path



- The blocks handling the internal clock (clock gating) and managing the sleep mode request/acknowledge protocol (enabling the synchronous path in active mode and the asynchronous path in idle mode).

21.4.1 Interrupt and Wake-Up Features

21.4.1.1 Synchronous Path: Interrupt Request Generation

The general-purpose interface has 12 interrupt lines (two interrupt lines per GPIO module instance). The 12 interrupt signals are GPIOi_MPU_IRQ (used by the MPU subsystem) and GPIOi_IVA2_IRQ (used by the IVA2.2 subsystem), where i = 1, 2, 3, 4, 5, and 6.

Synchronous interrupt requests from each channel are processed by two identical interrupt generation submodules used independently by the IVA2.2 subsystem and the MPU subsystem. Each submodule controls its own synchronous interrupt request line and has its own interrupt enable (GPIOi.GPIO_IRQENABLE1 or GPIOi.GPIO_IRQENABLE2) and interrupt status (GPIOi.GPIO_IRQSTATUS1 or GPIOi.GPIO_IRQSTATUS2) registers. The interrupt enable register selects the channel(s) considered for the interrupt request generation, and the interrupt status register determines which channel(s) activate the interrupt request. Event detection on GPIO channels is reflected in the interrupt status registers independent of the content of the interrupt enable registers.

In active mode, when the GPIO configuration registers are set to enable the interrupt generation (see Section 21.5.3, *Interrupt and Wakeup*), a synchronous path samples the transitions and levels on the input GPIO with the internally gated interface clock (see Section 21.3.1.1.4.4, *Module Power Saving*). When an event matches the programmed settings (see Section 21.5.3, *Interrupt and Wakeup*), the corresponding bit in the interrupt status register is set to 1 and, on the following interface clock cycle, the interrupt lines 1 and/or 2 are activated (depending on the interrupt enable registers).

Because of the sampling operation, the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the internally gated interface clock period (the internally gated interface clock period equals *N* times the interface clock period; see Section 21.3.1.1.4.4, *Module Power Saving*). This minimum pulse width must be met before and after any expected level transition detection. Level detection requires the selected level to be stable for at least two times the internally gated interface clock period to trigger a synchronous interrupt.

Because the module is synchronous, latency is minimal between the expected event occurrence and the activation of the interrupt line(s). This latency must not exceed four internally gated interface clock cycles + one interface clock cycle when the debounce feature is not used.

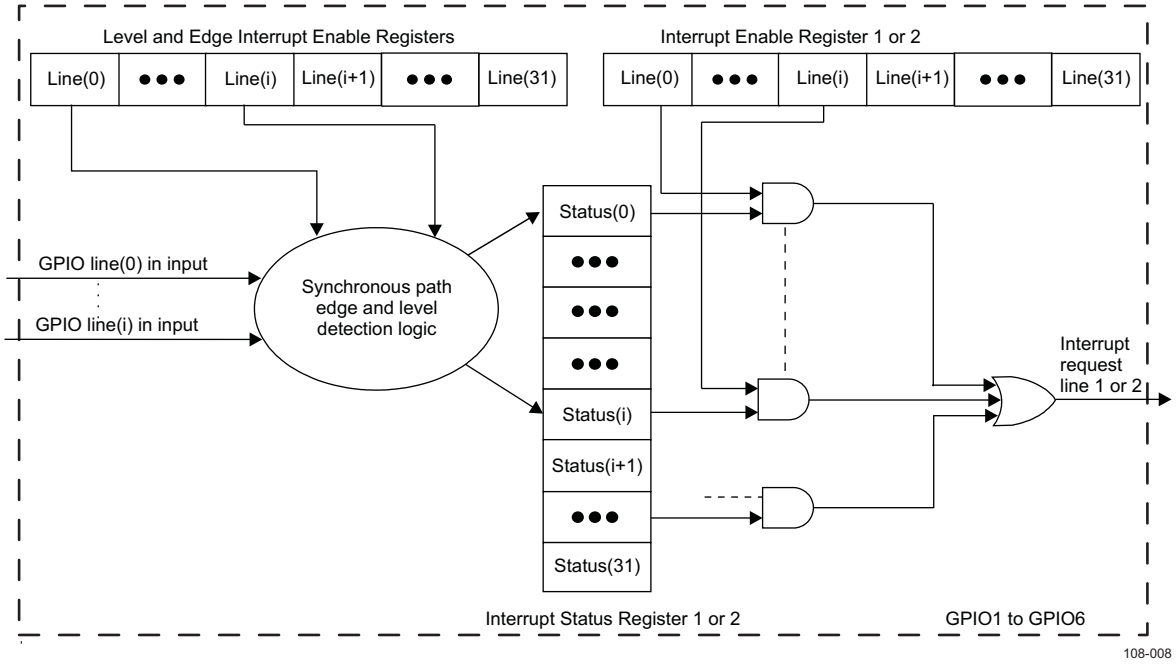
When the debounce feature is active, the latency depends on the debouncing time register (GPIOi.GPIO_DEBOUNCINGTIME) value (see Section 21.5.5, *Debouncing Time*) and is less than three internally gated interface clock cycles + two interface clock cycle + GPIOi.GPIO_DEBOUNCINGTIME register value debounce clock cycles + three debounce clock cycles.

Synchronous interrupt request line 1 is mapped on the MPU interrupt controller.

Synchronous interrupt request line 2 is mapped on the IVA2.2 interrupt controller.

Figure 21-8 shows an overview of the interrupt request generation.

Figure 21-8. Interrupt Request Generation



21.4.1.2 Asynchronous Path: Wake-Up Request Generation

The general-purpose interface has six wake-up lines (one wake-up line per GPIO module instance) connected to the PRCM module.

Asynchronous wake-up requests from input channels are merged to issue one wake-up signal to the system per GPIO module. The wake-up enable register (GPIOi.GPIO_WAKEUPENABLE) selects the channel(s) considered for the wake-up request generation. The asynchronous wake-up request is reflected into the synchronous interrupt status registers (GPIOi.GPIO_IRQSTATUS1 and GPIOi.GPIO_IRQSTATUS2).

In idle mode (the interface clock is shut down and the GPIO configuration registers are programmed; see Section 21.5.3, *Interrupt and Wakeup*), an asynchronous path detects the expected transition(s) on a GPIO input (based on register programming) and activates an asynchronous wake-up request by the sideband signal (GPIOi_SWAKEUP, where i = 1, 2, 3, 4, 5, and 6), if the wake-up enable register is set.

When the system is awakened, the interface clock is restarted and synchronously set to 1 based on the input GPIO pin triggering the wake-up request and the corresponding bit in the interrupt status registers (GPIOi.GPIO_IRQSTATUS1 and GPIOi.GPIO_IRQSTATUS2). On the following internal clock cycle, the interrupt lines 1 and/or 2 are active (active low) when the corresponding bits are set in the interrupt enable registers (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2).

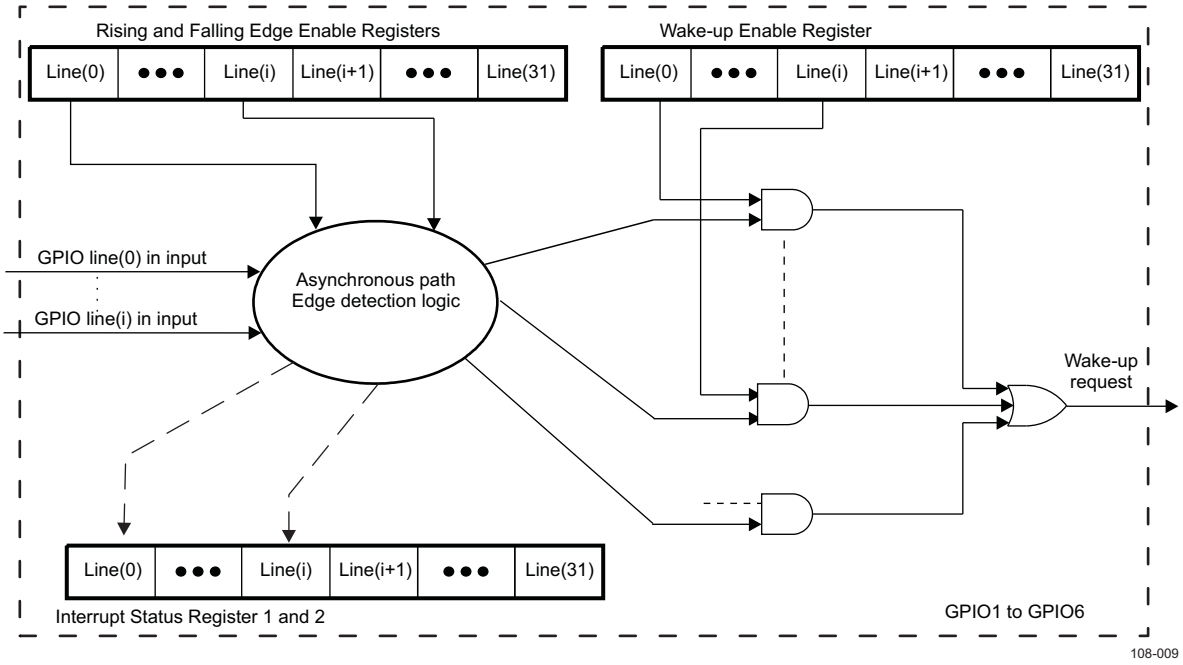
NOTE: When debouncing is not enabled, a minimum input pulse width does not trigger the wake-up request because there is no sampling operation.

When debouncing is enabled, the minimum pulse width is set by the specified debouncing time.

The GPIOi.GPIO_SYSCONFIG[2] ENAWAKEUP bit enables or disables the GPIO wake-up feature globally. If the bit is 0, the wake-up enable register (GPIOi.GPIO_WAKEUPENABLE) has no effect.

Figure 21-9 shows an overview of the wake-up request generation.

Figure 21-9. Wake-Up Request Generation



21.4.1.3 Interrupt (or Wake-Up) Line Release

When the host processor (the MPU and/or IVA2.2 subsystem in the device) receives an interrupt request issued by the GPIO module, it reads the corresponding interrupt status register (GPIOi.GPIO_IRQSTATUS1 or GPIOi.GPIO_IRQSTATUS2) to determine which GPIO input triggered the interrupt (or the wake-up request).

After servicing the interrupt (or acknowledging the wake-up request), the processor resets the status bit and releases the interrupt line by writing 1 in the corresponding bit of the interrupt status register. If there is still a pending interrupt request to serve (all bits in the interrupt status register that are not masked by the interrupt enable register are not cleared), the interrupt line is reasserted.

NOTE: The status bit must be reset to re-enter idle mode.

21.5 General-Purpose Interface Basic Programming Model

21.5.1 Power Saving by Grouping the Edge/Level Detection

Each GPIO module implements four gated clocks used by the edge/level detection logic to save power. Each group of eight input GPIO pins generates a separate enable signal depending on the edge/level detection register setting (because the input is 32 bits, four groups of eight inputs are defined for each GPIO module). If a group requires no edge/level detection, then the corresponding clock is gated (cut off). Grouping the edge/level enable can save the power consumption of the module as described in the following example.

If any of the registers:

GPIOi.GPIO_LEVELDETECT0
GPIOi.GPIO_LEVELDETECT1
GPIOi.GPIO_RISINGDETECT
GPIOi.GPIO_FALLINGDETECT

are set to 0x01 01 01 01, then all clocks are active (power consumption is high).

are set to 0x00 00 00 FF, then a single clock is active (power saving).

NOTE: When the clocks are enabled by writing to the GPIOi.GPIO_LEVELDETECT0, GPIOi.GPIO_LEVELDETECT1, GPIOi.GPIO_RISINGDETECT, and GPIOi.GPIO_FALLINGDETECT registers, the detection starts after five clock cycles. This period is required to clean the synchronization edge/level detection pipeline.

The mechanism is independent of each clock group. If the clock has been started before and a new setting is performed, the following is recommended: First, set the new detection required; second, disable the previous setting (if necessary). In this way, the corresponding clock is not gated and the detection starts immediately.

21.5.2 Set-and-Clear Instructions

21.5.2.1 Description

The GPIO module implements the set-and-clear protocol register update for the GPIOi.GPIO_DATAOUT, GPIOi.GPIO_IRQENABLE1, GPIOi.GPIO_IRQENABLE2, and GPIOi.GPIO_WAKEUPENABLE registers. This protocol is an alternative to the atomic test and set operations and consists of writing operations at dedicated addresses (one address for setting bit[s] and one address for clearing bit[s]). The data to write is 1 at bit position(s) to clear (or to set) and 0 at unaffected bit(s). Registers can be accessed in two ways:

- Standard: Full register read and write operations at the primary register address
- Set and clear (recommended): Separate addresses are provided to set (and clear) bits in registers. Writing 1 at these addresses sets (or clears) the corresponding bit into the equivalent register; writing a 0 has no effect.

Therefore, for these registers, three addresses are defined for one unique physical register. Reading these addresses has the same effect and returns the register value.

21.5.2.2 Clear Instruction

21.5.2.2.1 Clear Registers Addresses

Clear interrupt enable registers (GPIOi.GPIO_CLEARIRQENABLE1 and GPIOi.GPIO_CLEARIRQENABLE2).

A write operation in the clear interrupt enable1 (or enable2) register clears the corresponding bit in the interrupt enable1 (or enable2) register when the written bit is 1; a written bit at 0 has no effect.

A read of the clear interrupt enable1 (or enable2) register returns the value of the interrupt enable1 (or enable2) register

Clear wake-up enable register (GPIOi.GPIO_CLEARWKUENA).

A write operation in the clear wake-up enable register clears the corresponding bit in the wake-up enable register when the written bit is 1; a written bit at 0 has no effect.

A read of the clear wake-up enable register returns the value of the wake-up enable register.

Clear data output register (GPIOi.GPIO_CLEARDATAOUT).

A write operation in the clear data output register clears the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect.

A read of the clear data output register returns the value of the data output register.

21.5.2.2.2 Clear Instruction Example

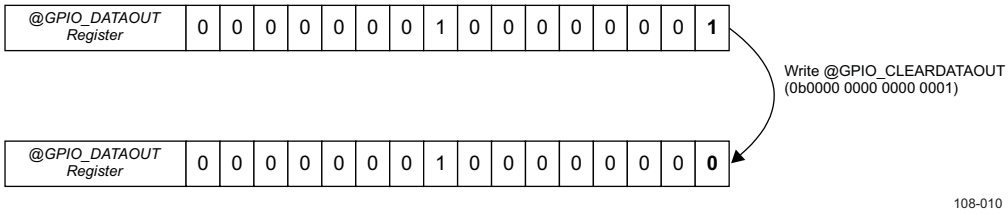
Assume the data output register (or one of the interrupt/wake-up enable register) contains the binary value, 0b0000 0001 0000 0001, and you want to clear the bit 0.

With the clear instruction feature, write 0b0000 0000 0000 0001 at the address of the clear data output register (or at the address of the clear interrupt/wake-up enable register). After this write operation, a reading of the data output register (or the interrupt/wake-up enable register) returns 0b0000 0001 0000 0000; the bit 0 is cleared.

NOTE: Although the general-purpose interface registers are 32 bits wide, only the less-significant 16 bits are represented in this example.

Figure 21-10 shows an example of a clear instruction.

Figure 21-10. Write @GPIO_CLEARDATAOUT Register Example



21.5.2.3 Set Instruction

21.5.2.3.1 Set Registers Addresses

Set interrupt enable registers (GPIOi.GPIO_SETIRQENABLE1 and GPIOi.GPIO_SETIRQENABLE2).

A write operation in the set interrupt enable1 (or enable2) register sets the corresponding bit in the interrupt enable1 (or enable2) register when the written bit is 1; a written bit at 0 has no effect.

A read of the set interrupt enable1 (or enable2) register returns the value of the interrupt enable1 (or enable2) register.

Set wake-up enable register (GPIOi.GPIO_SETWKUENA).

A write operation in the set wake-up enable register sets the corresponding bit in the wake-up enable register when the written bit is 1; a written bit at 0 has no effect.

A read of the set wake-up enable register returns the value of the wake-up enable register.

Set data output register (GPIOi.GPIO_SETDATAOUT).

A write operation in the set data output register sets the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect.

A read of the set data output register returns the value of the data output register.

21.5.2.3.2 Set Instruction Example

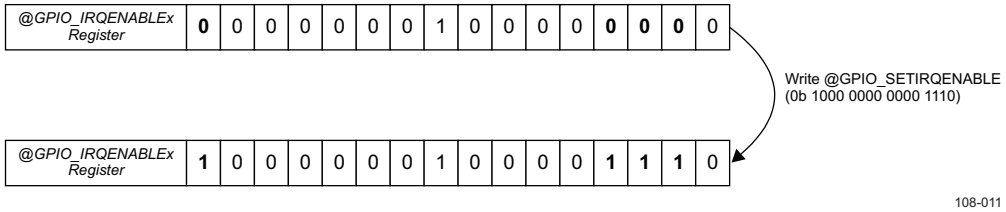
Assume the interrupt enable1 (or enable2) register (or the data output register) contains the binary value, 0b0000 0001 0000 0000, and you want to set the bits 15, 3, 2, and 1.

With the set instruction feature, write 0b1000 0000 0000 1110 at the address of the set interrupt enable1 (or enable2) register (or at the address of the set data output register). After this write operation, a reading of the interrupt enable1 (or enable2) register (or the data output register) returns 0b1000 0001 0000 1110; the bits 15, 3, 2, and 1 are set.

NOTE: Although the general-purpose interface registers are 32 bits wide, only the less-significant 16 bits are represented in this example.

Figure 21-11 shows an example of a set instruction.

Figure 21-11. Write @GPIO_SETIRQENABLEx Register Example



The set wake-up enable register offers the same feature with the wake-up enable register.

21.5.3 Interrupt and Wakeup

21.5.3.1 Involved Configuration Registers

- Interrupt enable registers (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2)

  The interrupt enable1 (or interrupt enable2) register allows masking of the expected transition on input GPIO to prevent the generation of an interrupt request on line1 (or line2). The interrupt enable registers are programmed synchronously with the interface clock.

  These registers can be accessed with direct read/write operations or using the alternate set and clear protocol register update feature. This feature enables to set or clear specific bits of these registers with a single write access to the corresponding set interrupt enable1 (or interrupt enable2) registers (or to the clear interrupt enable1 [or interrupt enable2] registers) address (see Section 21.5.2, *Set-and-Clear Instructions*).

- Wake-up enable register (GPIOi.GPIO_WAKEUPENABLE)

  The wake-up enable register allows masking of the expected transition on input GPIO to prevent the generation of a wake-up request. The wake-up enable register is programmed synchronously with the interface clock before any idle mode request coming from the host processor.

  This register can be accessed with direct read/write operations or by using the alternate set and clear protocol register update feature. This feature allows setting or clearing specific bits of this register with a single write access to the set wake-up enable register (or to the clear wake-up enable register) address (see Section 21.5.2, *Set-and-Clear Instructions*).

  NOTE: It must be a correlation between Wake-up enable and interrupt enable registers. If a GPIO pin has a Wake-up configured on it, it should also have the corresponding interrupt enabled (on one of the 2 interrupt lines). Otherwise, it is possible to have a Wake-up event, but after exiting the Idle state, no interrupt will be generated, thus the corresponding bit from the interrupt status register will not be cleared, and the module will not acknowledge a future Idle Request.

- Interrupt status registers (GPIOi.GPIO_IRQSTATUS1 and GPIOi.GPIO_IRQSTATUS2)

  The interrupt status1 (or interrupt status2) register determines which of the input GPIO pins triggered the interrupt line1 (or interrupt line2) request (or the wake-up line).

  When a bit in this register is set to 1, it indicates that the corresponding GPIO pin is requesting the interrupt (or the wake-up). To reset a bit in this register, write 1 to the appropriate bit. However, an interrupt cannot be generated by writing 1 to the interrupt status1 (or interrupt status2) register.

If 0 is written to a bit in this register, the value remains unchanged. The interrupt status1 (or interrupt status2) register is synchronous with the interface clock. In idle mode, the event is detected via an asynchronous path, and the corresponding bit in the interrupt status1 and interrupt status2 registers are set when the GPIO module is awake.

NOTE: The wake-up capabilities of the GPIO2 to GPIO6 modules are operational only when the PER power domain is active.

21.5.3.2 Description

To generate interrupt request to a host processor (the MPU and/or DSP subsystem in the device) at a defined event (level or edge logic transition) occurring on a GPIO pin (interrupt source), the GPIO configuration registers must be programmed as follows:

1. The GPIO channel must be configured as input by the output enable register (write 1 to the corresponding bit of the GPIOi.GPIO_OE register).
2. The expected event(s) on the GPIO input to trigger the interrupt request must be selected in the low-level interrupt enable register (write 1or 0 to the corresponding bit of GPIOi.GPIO_LEVELDETECT0), and/or high-level interrupt enable register (write 1 or 0 to the corresponding bit of GPIOi.GPIO_LEVELDETECT1), and/or rising-edge interrupt/wake-up enable register (write 1or 0 to the corresponding bit of GPIOi.GPIO_RISINGDETECT), and/or falling edge interrupt/wake-up enable register (write 1or 0 to the corresponding bit of GPIOi.GPIO_FALLINGDETECT).

   NOTE: Interrupt generation on both edges on one input is configured by setting the corresponding bit to 1 in the rinsing detect enabling register (GPIOi.GPIO_RISINGDETECT) and falling detect enabling register (GPIOi.GPIO_FALLINGDETECT) along with the interrupt enable by setting the corresponding bit to 1 in on one or both interrupt enable registers (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2).

   Enabling at the same time high level detection and low level detection for one given pin makes a constant interrupt generator.

3. Interrupts from the GPIO channel must be enabled in the interrupt 1 enable register (write 1 to the corresponding bit of GPIOi.GPIO_IRQENABLE1 register) and/or the interrupt 2 enable register (write 1 to the corresponding bit of GPIOi.GPIO_IRQENABLE2 register).

To configure a GPIO module to sent a wake-up request to the PRCM at a defined event (logic transition) occurring on a GPIO pin (wake-up source), the GPIO configuration registers must be programmed as follows:

1. The GPIO pin must be configured as input by the output enable register (write 1 to the corresponding bit of the GPIOi.GPIO_OE register).
2. The expected event(s) on the GPIO input to trigger the wake-up request must be selected in the rising-edge interrupt/wake-up enable register (write 1 or 0 to the corresponding bit of GPIOi.GPIO_RISINGDETECT) and/or falling-edge interrupt/wake-up enable register (write 1 or 0 to the corresponding bit of GPIOi.GPIO_FALLINGDETECT). The wake-up request can only be generated on edge transitions.
3. The GPIO channel must be enabled in the wake-up enable register (write 1 to the corresponding bit of the GPIOi.GPIO_WAKEUPENABLE).
4. The wake-up request generation on the expected transition occurring on the GPIO input pins must enable for the module (write 1 to the corresponding bit of the GPIOi.GPIO_SYSCONFIG[2] ENAWAKEUP) .

CAUTION

For each GPIO channel used, do not forget to configure the corresponding pad configuration registers in Chapter 6 , *System Control Module*.

After servicing the interrupt, the status bit in the interrupt status register (GPIOi.GPIO_IRQSTATUS1 or GPIOi.GPIO_IRQSTATUS2) must be reset and the interrupt line released (by writing 1 in the corresponding bit of the interrupt status register) before enabling an interrupt for the GPIO channel in the interrupt enable register (GPIOi.GPIO_IRQENABLE1 or GPIOi.GPIO_IRQENABLE2) to prevent the occurrence of unexpected interrupts when enabling an interrupt for the GPIO channel.

21.5.4 Data Input (Capture)/Output (Drive)

The output enable register (GPIOi.GPIO_OE) controls the output/input capability for each pin. At reset, all the GPIO-related pins are configured as input and output capabilities are disabled. This register is not used within the module. Its only function is to carry the pads configuration.

When configured as an output (the desired bit reset in the GPIOi.GPIO_OE register), the value of the corresponding bit in the GPIOi.GPIO_DATAOUT register is driven on the corresponding GPIO pin. Data is written to the data output register synchronously with the interface clock. This register can be accessed with read/write operations or by using the alternate set and clear protocol register update feature. This feature lets you set or clear specific bits of this register with a single write access to the set output data register (GPIOi.GPIO_SETDATAOUT) or to the clear output data register (GPIOi.GPIO_CLEARDATAOUT) address (see Section 21.5.2, *Set-and-Clear Instructions*). If the application uses a pin as an output and does not want interrupt/wake-up generation from this pin, the application must properly configure the wake-up enable (GPIOi.GPIO_ WAKEUPENABLE) and the interrupt enable (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2) registers.

When configured as an input (the desired bit set to 1 in the GPIOi.GPIO_OE register), the state of the input can be read from the corresponding bit in the GPIOi.GPIO_DATAIN register. The input data is sampled synchronously with the interface clock and then captured in the data input register synchronously with the interface clock (see Section 21.5.2, *Set-and-Clear Instructions*). When the GPIO pin levels change, they are captured into this register after two interface clock cycles (the required cycles to synchronize and to write data). If the application uses a pin as an input, the application must properly configure the wake-up enable (GPIOi.GPIO_WAKEUPENABLE) and the interrupt enable (GPIOi.GPIO_IRQENABLE1 and GPIOi.GPIO_IRQENABLE2) registers to the interrupt and wake-up feature as needed.

21.5.5 Debouncing Time

To enable the debounce feature for a pin, the GPIO configuration registers must be programmed as follows:

1. The GPIO pin must be configured as input in the output enable register (write 1 to the corresponding bit of the GPIOi.GPIO_OE register)
2. The deboucing time must be set in the deboucing time register (GPIOi.GPIO_DEBOUNCINGTIME)

   The debouncing value register (GPIOi.GPIO_DEBOUNCINGTIME) is used to set the debouncing time for all input lines in the GPIO module. The value is global for all the ports of one GPIO module, so up to six different debouncing values are possible. The debounce cell is running with the debounce clock (32 kHz). This register represents the number of the clock cycle(s) (one cycle is 31 microseconds long) to be used.

   The following formula describes the required input stable time to be propagated to the debounced output:

   Required input line stable = (GPIOi.GPIO_DEBOUNCINGTIME[7:0] DEBOUNCVAL field value + 1) x 31 µs.

   where GPIOi.GPIO_DEBOUNCINGTIME[7:0] field DEBOUNCVAL value is from 0 to 255.
3. The debouncing feature must be enabled in the debouncing enable register (write 1 to the corresponding bit of the GPIOi.GPIO_DEBOUNCENABLE register)

21.6 General-Purpose Interface Register Manual

This section summarizes the hardware interface for the GPIO product. Each module instance within the design is shown, together with the module register map and bit definitions for each bit field.

Table 21-6 shows the base address and address space for the GPIO module instances.

Table 21-6. Instance Summary

Module Name	Base Address	Size
GPIO1	0x4831 0000	4K bytes
GPIO2	0x4905 0000	4K bytes
GPIO3	0x4905 2000	4K bytes
GPIO4	0x4905 4000	4K bytes
GPIO5	0x4905 6000	4K bytes
GPIO6	0x4905 8000	4K bytes

21.6.1 General-Purpose Interface Register Mapping Summary

All module registers are 8-, 16-, or 32-bit accessible through the L4 interconnect (little endian encoding). Access to registers is direct; no shadow registers are implemented.

Table 21-7 through Table 21-8 describe the GPIO register offset addresses.

Table 21-7. GPIO1 to GPIO3 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address (GPIO1)	Physical Address (GPIO2)	Physical Address (GPIO3)
GPIO_REVISION	R	32	0x000	0x4831 0000	0x4905 0000	0x4905 2000
GPIO_SYSCONFIG	RW	32	0x010	0x4831 0010	0x4905 0010	0x4905 2010
GPIO_SYSSTATUS	R	32	0x014	0x4831 0014	0x4905 0014	0x4905 2014
GPIO_IRQSTATUS1	RW	32	0x018	0x4831 0018	0x4905 0018	0x4905 2018
GPIO_IRQENABLE1	RW	32	0x01C	0x4831 001C	0x4905 001C	0x4905 201C
GPIO_WAKEUPENABLE	RW	32	0x020	0x4831 0020	0x4905 0020	0x4905 2020
GPIO_IRQSTATUS2	RW	32	0x028	0x4831 0028	0x4905 0028	0x4905 2028
GPIO_IRQENABLE2	RW	32	0x02C	0x4831 002C	0x4905 002C	0x4905 202C
GPIO_CTRL	RW	32	0x030	0x4831 0030	0x4905 0030	0x4905 2030
GPIO_OE	RW	32	0x034	0x4831 0034	0x4905 0034	0x4905 2034
GPIO_DATAIN	R	32	0x038	0x4831 0038	0x4905 0038	0x4905 2038
GPIO_DATAOUT	RW	32	0x03C	0x4831 003C	0x4905 003C	0x4905 203C
GPIO_LEVELDETECT0	RW	32	0x040	0x4831 0040	0x4905 0040	0x4905 2040
GPIO_LEVELDETECT1	RW	32	0x044	0x4831 0044	0x4905 0044	0x4905 2044
GPIO_RISINGDETECT	RW	32	0x048	0x4831 0048	0x4905 0048	0x4905 2048
GPIO_FALLINGDETECT	RW	32	0x04C	0x4831 004C	0x4905 004C	0x4905 204C
GPIO_DEBOUNCENABLE	RW	32	0x050	0x4831 0050	0x4905 0050	0x4905 2050
GPIO_DEBOUNCINGTIME	RW	32	0x054	0x4831 0054	0x4905 0054	0x4905 2054
GPIO_CLEARIRQENABLE 1	RW	32	0x060	0x4831 0060	0x4905 0060	0x4905 2060
GPIO_SETIRQENABLE1	RW	32	0x064	0x4831 0064	0x4905 0064	0x4905 2064
GPIO_CLEARIRQENABLE 2	RW	32	0x070	0x4831 0070	0x4905 0070	0x4905 2070
GPIO_SETIRQENABLE2	RW	32	0x074	0x4831 0074	0x4905 0074	0x4905 2074
GPIO_CLEARWKUENA	RW	32	0x080	0x4831 0080	0x4905 0080	0x4905 2080
GPIO_SETWKUENA	RW	32	0x084	0x4831 0084	0x4905 0084	0x4905 2084
GPIO_CLEARDATAOUT	RW	32	0x090	0x4831 0090	0x4905 0090	0x4905 2090

Table 21-7. GPIO1 to GPIO3 Register Summary (continued)

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address (GPIO1)	Physical Address (GPIO2)	Physical Address (GPIO3)
GPIO_SETDATAOUT	RW	32	0x094	0x4831 0094	0x4905 0094	0x4905 2094

Table 21-8. GPIO4 to GPIO6 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address (GPIO4)	Physical Address (GPIO5)	Physical Address (GPIO6)
GPIO_REVISION	R	32	0x000	0x4905 4000	0x4905 6000	0x4905 8000
GPIO_SYSCONFIG	RW	32	0x010	0x4905 4010	0x4905 6010	0x4905 8010
GPIO_SYSSTATUS	R	32	0x014	0x4905 4014	0x4905 6014	0x4905 8014
GPIO_IRQSTATUS1	RW	32	0x018	0x4905 4018	0x4905 6018	0x4905 8018
GPIO_IRQENABLE1	RW	32	0x01C	0x4905 401C	0x4905 601C	0x4905 801C
GPIO_WAKEUPENABLE	RW	32	0x020	0x4905 4020	0x4905 6020	0x4905 8020
GPIO_IRQSTATUS2	RW	32	0x028	0x4905 4028	0x4905 6028	0x4905 8028
GPIO_IRQENABLE2	RW	32	0x02C	0x4905 402C	0x4905 602C	0x4905 802C
GPIO_CTRL	RW	32	0x030	0x4905 4030	0x4905 6030	0x4905 8030
GPIO_OE	RW	32	0x034	0x4905 4034	0x4905 6034	0x4905 8034
GPIO_DATAIN	R	32	0x038	0x4905 4038	0x4905 6038	0x4905 8038
GPIO_DATAOUT	RW	32	0x03C	0x4905 403C	0x4905 603C	0x4905 803C
GPIO_LEVELDETECT0	RW	32	0x040	0x4905 4040	0x4905 6040	0x4905 8040
GPIO_LEVELDETECT1	RW	32	0x044	0x4905 4044	0x4905 6044	0x4905 8044
GPIO_RISINGDETECT	RW	32	0x048	0x4905 4048	0x4905 6048	0x4905 8048
GPIO_FALLINGDETECT	RW	32	0x04C	0x4905 404C	0x4905 604C	0x4905 804C
GPIO_DEBOUNCENABLE	RW	32	0x050	0x4905 4050	0x4905 6050	0x4905 8050
GPIO_DEBOUNCINGTIME	RW	32	0x054	0x4905 4054	0x4905 6054	0x4905 8054
GPIO_CLEARIRQENABLE 1	RW	32	0x060	0x4905 4060	0x4905 6060	0x4905 8060
GPIO_SETIRQENABLE1	RW	32	0x064	0x4905 4064	0x4905 6064	0x4905 8064
GPIO_CLEARIRQENABLE 2	RW	32	0x070	0x4905 4070	0x4905 6070	0x4905 8070
GPIO_SETIRQENABLE2	RW	32	0x074	0x4905 4074	0x4905 6074	0x4905 8074
GPIO_CLEARWKUENA	RW	32	0x080	0x4905 4080	0x4905 6080	0x4905 8080
GPIO_SETWKUENA	RW	32	0x084	0x4905 4084	0x4905 6084	0x4905 8084
GPIO_CLEARDATAOUT	RW	32	0x090	0x4905 4090	0x4905 6090	0x4905 8090
GPIO_SETDATAOUT	RW	32	0x094	0x4905 4094	0x4905 6094	0x4905 8094

The write latency for all the R/W registers is immediate (with respect to the interface clock)

NOTE: When two write accesses in the GPIOi.GPIO_DEBOUNCINGTIME register are performed in less than two debounce clock cycles (32 kHz) + four interface clock cycles, the first write access latency is immediate, but the second write access is acknowledged only after this interval ends.

In the register descriptions in this section, when one single register carries an individual configuration or setting for all the channels of the module, one bit in the register is dedicated to each channel. The bit and the corresponding channel are identified with the same number: bit 0 refers to channel 0, bit 1 refers to channel 1, and so on, up to 31.

21.6.2 Register Descriptions

Table 21-9 through Table 21-59 describe the register bits.

Table 21-9. GPIO_REVISION

Address Offset	0x000		
Physical Address	0x4831 0000	**Instance**	GPIO1
	0x4905 0000		GPIO2
	0x4905 2000		GPIO3
	0x4905 4000		GPIO4
	0x4905 6000		GPIO5
	0x4905 8000		GPIO6
Description	This register contains the IP revision code.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								GPIOREVISION							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Read returns 0	R	0x000000
7:0	GPIOREVISION	IP revision [7:4] Major revision [3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1	R	See (1)

(1) TI internal data

Table 21-10. Register Call Summary for Register GPIO_REVISION

General-Purpose Interface Register Manual

- General-Purpose Interface Register Mapping Summary: [0] [1]

Table 21-11. GPIO_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4831 0010	**Instance**	GPIO1
	0x4905 0010		GPIO2
	0x4905 2010		GPIO3
	0x4905 4010		GPIO4
	0x4905 6010		GPIO5
	0x4905 8010		GPIO6
Description	This register controls the various parameters of the L4 interconnect.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																											IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:5	RESERVED	Write 0s for future compatibility. Read returns 0	RW	0x0000000
4:3	IDLEMODE	Power Management, Req/Ack control	RW	0x0
		0x0: Force-idle. An idle request is acknowledged unconditionally		
		0x1: No-idle. An idle request is never acknowledged		
		0x2: Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module		
		0x3: reserved do not use		
2	ENAWAKEUP	Wakeup capability enabled/disabled	RW	0x0
		0x0: Wakeup disable		
		0x1: Wakeup enable		
1	SOFTRESET	Software reset. This bit is automatically reset by the hardware. During reads, it always returns 0.	RW	0x0
		0x0: Normal mode		
		0x1: The module is reset		
0	AUTOIDLE	Internal interface clock gating strategy	RW	0x0
		0x0: interface clock is free-running		
		0x1: Automatic interface clock gating strategy is applied, based on the L4 interconnect activity		

Table 21-12. Register Call Summary for Register GPIO_SYSCONFIG

General-Purpose Interface Integration • Clocking, Reset, and Power-Management Scheme: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
General-Purpose Interface Functional Description • Asynchronous Path: Wake-Up Request Generation: [10]
General-Purpose Interface Basic Programming Model • Description: [11]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [12] [13]

Table 21-13. GPIO_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4831 0014	**Instance**	GPIO1
	0x4905 0014		GPIO2
	0x4905 2014		GPIO3
	0x4905 4014		GPIO4
	0x4905 6014		GPIO5
	0x4905 8014		GPIO6
Description	This register provides status information about the module, excluding the interrupt status information.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								RESERVED							RESETDONE

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Read returns 0	R	0x000000
7:1	RESERVED	Read returns 0	R	0x00
0	RESETDONE	Internal reset monitoring 0x0: Internal module reset in on-going 0x1: Reset completed	R	

Table 21-14. Register Call Summary for Register GPIO_SYSSTATUS

General-Purpose Interface Integration • Clocking, Reset, and Power-Management Scheme: [0] [1]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [2] [3]

Table 21-15. GPIO_IRQSTATUS1

Address Offset	0x018		
Physical Address	0x4831 0018	**Instance**	GPIO1
	0x4905 0018		GPIO2
	0x4905 2018		GPIO3
	0x4905 4018		GPIO4
	0x4905 6018		GPIO5
	0x4905 8018		GPIO6
Description	This register provides IRQ 1 status information.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQSTATUS1																															

Bits	Field Name	Description	Type	Reset
31:0	IRQSTATUS1	Interrupt 1 Status Register. Write a 1 in the corresponding bit to clear it to 0. Write 0 in the corresponding bit does not affect its value. 0x0: IRQ channel N not triggered 0x1: IRQ channel N triggered	RW	0x00000000

Table 21-16. Register Call Summary for Register GPIO_IRQSTATUS1

General-Purpose Interface Functional Description • Synchronous Path: Interrupt Request Generation: [0] • Asynchronous Path: Wake-Up Request Generation: [1] [2] • Interrupt (or Wake-Up) Line Release: [3]
General-Purpose Interface Basic Programming Model • Involved Configuration Registers: [4] • Description: [5]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [6] [7]

Table 21-17. GPIO_IRQENABLE1

Address Offset	0x01C		
Physical Address	0x4831 001C	**Instance**	GPIO1
	0x4905 001C		GPIO2
	0x4905 201C		GPIO3
	0x4905 401C		GPIO4
	0x4905 601C		GPIO5
	0x4905 801C		GPIO6
Description	This register provides IRQ 1 enable information.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQENABLE1																															

Bits	Field Name	Description	Type	Reset
31:0	IRQENABLE1	Interrupt 1 Enable Register	RW	0x00000000
		0x0: disable IRQ generation for channel N		
		0x1: enable IRQ generation for channel N		

Table 21-18. Register Call Summary for Register GPIO_IRQENABLE1

General-Purpose Interface Overview
- Global Features: [0]

General-Purpose Interface Integration
- Hardware Requests: [1]

General-Purpose Interface Functional Description
- General-Purpose Interface Functional Description: [2]
- Synchronous Path: Interrupt Request Generation: [3]
- Asynchronous Path: Wake-Up Request Generation: [4]

General-Purpose Interface Basic Programming Model
- Description: [5]
- Involved Configuration Registers: [6]
- Description: [7] [8] [9]
- Data Input (Capture)/Output (Drive): [10] [11]

General-Purpose Interface Register Manual
- General-Purpose Interface Register Mapping Summary: [12] [13]
- Register Descriptions: [14] [15] [16] [17]

Table 21-19. GPIO_WAKEUPENABLE

Address Offset	0x020		
Physical Address	0x4831 0020	**Instance**	GPIO1
	0x4905 0020		GPIO2
	0x4905 2020		GPIO3
	0x4905 4020		GPIO4
	0x4905 6020		GPIO5
	0x4905 8020		GPIO6
Description	This register provides wake-up enable information.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUPEN																															

Bits	Field Name	Description	Type	Reset
31:0	WAKEUPEN	Wake Up Enable Register 0x0: disable wakeup generation for channel N 0x1: enable wakeup generation for channel N	RW	0x00000000

Table 21-20. Register Call Summary for Register GPIO_WAKEUPENABLE

General-Purpose Interface Overview • Global Features: [0]
General-Purpose Interface Integration • Hardware Requests: [1]
General-Purpose Interface Functional Description • Asynchronous Path: Wake-Up Request Generation: [2] [3]
General-Purpose Interface Basic Programming Model • Description: [4] • Involved Configuration Registers: [5] • Description: [6] • Data Input (Capture)/Output (Drive): [7]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [8] [9] • Register Descriptions: [10] [11] [12] [13]

Table 21-21. GPIO_IRQSTATUS2

Address Offset	0x028		
Physical Address	0x4831 0028	**Instance**	GPIO1
	0x4905 0028		GPIO2
	0x4905 2028		GPIO3
	0x4905 4028		GPIO4
	0x4905 6028		GPIO5
	0x4905 8028		GPIO6
Description	This register provides IRQ 2 status information.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQSTATUS2																															

Bits	Field Name	Description	Type	Reset
31:0	IRQSTATUS2	Interrupt 2 Status Register. Write a 1 in the corresponding bit to clear it to 0. Write 0 in the corresponding bit does not affect its value. 0x0: IRQ channel N not triggered 0x1: IRQ channel N triggered	RW	0x00000000

Table 21-22. Register Call Summary for Register GPIO_IRQSTATUS2

General-Purpose Interface Functional Description • Synchronous Path: Interrupt Request Generation: [0] • Asynchronous Path: Wake-Up Request Generation: [1] [2] • Interrupt (or Wake-Up) Line Release: [3]

Table 21-22. Register Call Summary for Register GPIO_IRQSTATUS2 (continued)

General-Purpose Interface Basic Programming Model • Involved Configuration Registers: [4] • Description: [5]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [6] [7]

Table 21-23. GPIO_IRQENABLE2

Address Offset	0x02C		
Physical Address	0x4831 002C	**Instance**	GPIO1
	0x4905 002C		GPIO2
	0x4905 202C		GPIO3
	0x4905 402C		GPIO4
	0x4905 602C		GPIO5
	0x4905 802C		GPIO6
Description	This register provides IRQ 2 enable information.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQENABLE2																															

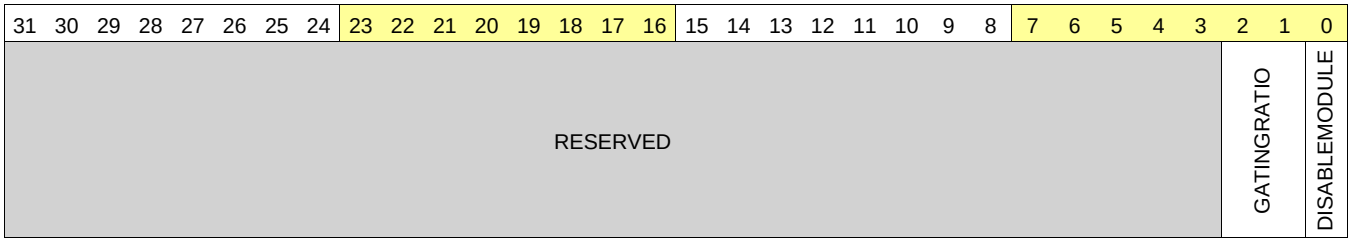
Bits	Field Name	Description	Type	Reset
31:0	IRQENABLE2	Interrupt 2 Enable Register 0x0: disable IRQ generation for channel N 0x1: enable IRQ generation for channel N	RW	0x00000000

Table 21-24. Register Call Summary for Register GPIO_IRQENABLE2

General-Purpose Interface Overview • Global Features: [0]
General-Purpose Interface Integration • Hardware Requests: [1]
General-Purpose Interface Functional Description • General-Purpose Interface Functional Description: [2] • Synchronous Path: Interrupt Request Generation: [3] • Asynchronous Path: Wake-Up Request Generation: [4]
General-Purpose Interface Basic Programming Model • Description: [5] • Involved Configuration Registers: [6] • Description: [7] [8] [9] • Data Input (Capture)/Output (Drive): [10] [11]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [12] [13] • Register Descriptions: [14] [15] [16] [17]

Table 21-25. GPIO_CTRL

Address Offset	0x030		
Physical Address	0x4831 0030	**Instance**	GPIO1
	0x4905 0030		GPIO2
	0x4905 2030		GPIO3
	0x4905 4030		GPIO4
	0x4905 6030		GPIO5
	0x4905 8030		GPIO6
Description	This register controls the clock gating functionality.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:3	RESERVED	Read returns 0	RW	0x00000000
2:1	GATINGRATIO	Gating Ratio	RW	0x1
		0x0: Functional clock is interface clock.		
		0x1: Functional clock is interface clock divided by 2.		
		0x2: Functional clock is interface clock divided by 4.		
		0x3: Functional clock is interface clock divided by 8.		
0	DISABLEMODULE	Module Disable	RW	0x0
		0x0: Module is enabled, clocks are not gated		
		0x1: Module is disabled, clocks are gated		

Table 21-26. Register Call Summary for Register GPIO_CTRL

General-Purpose Interface Integration
• Clocking, Reset, and Power-Management Scheme: [0] [1] [2]
General-Purpose Interface Register Manual
• General-Purpose Interface Register Mapping Summary: [3] [4]

Table 21-27. GPIO_OE

Address Offset	0x034		
Physical Address	0x4831 0034	**Instance**	GPIO1
	0x4905 0034		GPIO2
	0x4905 2034		GPIO3
	0x4905 4034		GPIO4
	0x4905 6034		GPIO5
	0x4905 8034		GPIO6
Description	This register is used to enable the pins output capabilities. Its only function is to carry the pads configuration.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTPUTEN																															

Bits	Field Name	Description	Type	Reset
31:0	OUTPUTEN	Output Data Enable 0x0: The corresponding GPIO port is configured as output 0x1: The corresponding GPIO port is configured as input	RW	0xFFFFFFFF

Table 21-28. Register Call Summary for Register GPIO_OE

General-Purpose Interface Overview • Global Features: [0]
General-Purpose Interface Basic Programming Model • Description: [1] [2] • Data Input (Capture)/Output (Drive): [3] [4] [5] • Debouncing Time: [6]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [7] [8]

Table 21-29. GPIO_DATAIN

Address Offset	0x038		
Physical Address	0x4831 0038	**Instance**	GPIO1
	0x4905 0038		GPIO2
	0x4905 2038		GPIO3
	0x4905 4038		GPIO4
	0x4905 6038		GPIO5
	0x4905 8038		GPIO6
Description	This register is used to register the data that is read from the GPIO pins.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAINPUT																															

Bits	Field Name	Description	Type	Reset
31:0	DATAINPUT	Sampled Input Data	R	0x00000000

Table 21-30. Register Call Summary for Register GPIO_DATAIN

General-Purpose Interface Overview • Global Features: [0]
General-Purpose Interface Integration • Clocking, Reset, and Power-Management Scheme: [1]
General-Purpose Interface Basic Programming Model • Data Input (Capture)/Output (Drive): [2]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [3] [4]

Table 21-31. GPIO_DATAOUT

Address Offset	0x03C		
Physical Address	0x4831 003C	**Instance**	GPIO1
	0x4905 003C		GPIO2
	0x4905 203C		GPIO3
	0x4905 403C		GPIO4
	0x4905 603C		GPIO5
	0x4905 803C		GPIO6
Description	This register is used for setting the value of the GPIO output pins		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUTPUT																															

Bits	Field Name	Description	Type	Reset
31:0	DATAOUTPUT	Output Data	RW	0x00000000

Table 21-32. Register Call Summary for Register GPIO_DATAOUT

General-Purpose Interface Overview
- Global Features: [0] [1]

General-Purpose Interface Basic Programming Model
- Description: [2]
- Data Input (Capture)/Output (Drive): [3]

General-Purpose Interface Register Manual
- General-Purpose Interface Register Mapping Summary: [4] [5]
- Register Descriptions: [6] [7] [8] [9]

Table 21-33. GPIO_LEVELDETECT0

Address Offset	0x040		
Physical Address	0x4831 0040	**Instance**	GPIO1
	0x4905 0040		GPIO2
	0x4905 2040		GPIO3
	0x4905 4040		GPIO4
	0x4905 6040		GPIO5
	0x4905 8040		GPIO6
Description	This register is used to enable/disable for each input lines the low-level (0) detection to be used for the interrupt request generation.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOWLEVEL																															

Bits	Field Name	Description	Type	Reset
31:0	LOWLEVEL	Low Level Interrupt Enable	RW	0x00000000
		0x0: disable the IRQ assertion on low level detect		
		0x1: enable the IRQ assertion on low level detect		

Table 21-34. Register Call Summary for Register GPIO_LEVELDETECT0

General-Purpose Interface Basic Programming Model • Power Saving by Grouping the Edge/Level Detection: [0] [1] • Description: [2]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [3] [4]

Table 21-35. GPIO_LEVELDETECT1

Address Offset	0x044		
Physical Address	0x4831 0044	**Instance**	GPIO1
	0x4905 0044		GPIO2
	0x4905 2044		GPIO3
	0x4905 4044		GPIO4
	0x4905 6044		GPIO5
	0x4905 8044		GPIO6
Description	This register is used to enable/disable for each input lines the high-level (1) detection to be used for the interrupt request generation.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIGHLEVEL																															

Bits	Field Name	Description	Type	Reset
31:0	HIGHLEVEL	High Level Interrupt Enable 0x0: disable the IRQ assertion on high level detect 0x1: enable the IRQ assertion on high level detect	RW	0x00000000

Table 21-36. Register Call Summary for Register GPIO_LEVELDETECT1

General-Purpose Interface Basic Programming Model • Power Saving by Grouping the Edge/Level Detection: [0] [1] • Description: [2]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [3] [4]

Table 21-37. GPIO_RISINGDETECT

Address Offset	0x048		
Physical Address	0x4831 0048	**Instance**	GPIO1
	0x4905 0048		GPIO2
	0x4905 2048		GPIO3
	0x4905 4048		GPIO4
	0x4905 6048		GPIO5
	0x4905 8048		GPIO6
Description	This register is used to enable/disable for each input lines the rising-edge (transition 0=>1) detection to be used for the interrupt request and the wake-up generation.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RISINGEDGE																															

Bits	Field Name	Description	Type	Reset
31:0	RISINGEDGE	Rising Edge Interrupt/Wakeup Enable 0x0: disable IRQ/Wakeup on rising edge detect 0x1: enable IRQ/Wakeup on rising edge detect	RW	0x00000000

Table 21-38. Register Call Summary for Register GPIO_RISINGDETECT

General-Purpose Interface Basic Programming Model • Power Saving by Grouping the Edge/Level Detection: [0] [1] • Description: [2] [3] [4]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [5] [6]

Table 21-39. GPIO_FALLINGDETECT

Address Offset	0x04C		
Physical Address	0x4831 004C	**Instance**	GPIO1
	0x4905 004C		GPIO2
	0x4905 204C		GPIO3
	0x4905 404C		GPIO4
	0x4905 604C		GPIO5
	0x4905 804C		GPIO6
Description	This register is used to enable/disable for each input lines the falling-edge (transition 1=>0) detection to be used for the interrupt request and the wake-up generation.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FALLINGEDGE																															

Bits	Field Name	Description	Type	Reset
31:0	FALLINGEDGE	Falling Edge Interrupt/Wakeup Enable 0x0: disable IRQ/Wakeup on falling edge detect 0x1: enable IRQ/Wakeup on falling edge detect	RW	0x00000000

Table 21-40. Register Call Summary for Register GPIO_FALLINGDETECT

General-Purpose Interface Basic Programming Model • Power Saving by Grouping the Edge/Level Detection: [0] [1] • Description: [2] [3] [4]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [5] [6]

Table 21-41. GPIO_DEBOUNCENABLE

Address Offset	0x050		
Physical Address	0x4831 0050	**Instance**	GPIO1
	0x4905 0050		GPIO2
	0x4905 2050		GPIO3
	0x4905 4050		GPIO4
	0x4905 6050		GPIO5
	0x4905 8050		GPIO6
Description	This register is used to enable/disable the debouncing feature for each input line.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEBOUNCEEN																															

Bits	Field Name	Description	Type	Reset
31:0	DEBOUNCEEN	Input Debounce Enable 0x0: disable debouncing feature on the corresponding input port 0x1: enable debouncing feature on the corresponding input port	RW	0x00000000

Table 21-42. Register Call Summary for Register GPIO_DEBOUNCENABLE

General-Purpose Interface Basic Programming Model • Debouncing Time: [0]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-43. GPIO_DEBOUNCINGTIME

Address Offset	0x054		
Physical Address	0x4831 0054	**Instance**	GPIO1
	0x4905 0054		GPIO2
	0x4905 2054		GPIO3
	0x4905 4054		GPIO4
	0x4905 6054		GPIO5
	0x4905 8054		GPIO6
Description	This register controls debouncing time (the value is global for all ports).		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								DEBOUNCEVAL							

Bits	Field Name	Description	Type	Reset
31:8	RESERVED	Read returns 0	RW	0x000000
7:0	DEBOUNCEVAL	Input Debouncing Value in 31 microsecond steps. debouncing time = (DEBOUNCEVAL+1) x 31 s	RW	0x00

Table 21-44. Register Call Summary for Register GPIO_DEBOUNCINGTIME

General-Purpose Interface Integration • Clocking, Reset, and Power-Management Scheme: [0]
General-Purpose Interface Functional Description • Synchronous Path: Interrupt Request Generation: [1] [2]
General-Purpose Interface Basic Programming Model • Debouncing Time: [3] [4] [5] [6]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [7] [8] [9]

Table 21-45. GPIO_CLEARIRQENABLE1

Address Offset	0x060		
Physical Address	0x4831 0060	**Instance**	GPIO1
	0x4905 0060		GPIO2
	0x4905 2060		GPIO3
	0x4905 4060		GPIO4
	0x4905 6060		GPIO5
	0x4905 8060		GPIO6
Description	Clear to 0 the corresponding bits in the GPIO_IRQENABLE1 register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLEARIRQEN1																															

Bits	Field Name	Description	Type	Reset
31:0	CLEARIRQEN1	Clear Interrupt Enable 1 0x0: no effect 0x1: Clear the corresponding bit in the GPIO_IRQENABLE1 register	RW	0x00000000

Table 21-46. Register Call Summary for Register GPIO_CLEARIRQENABLE1

General-Purpose Interface Basic Programming Model
- Clear Instruction: [0]

General-Purpose Interface Register Manual
- General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-47. GPIO_SETIRQENABLE1

Address Offset	0x064		
Physical Address	0x4831 0064	**Instance**	GPIO1
	0x4905 0064		GPIO2
	0x4905 2064		GPIO3
	0x4905 4064		GPIO4
	0x4905 6064		GPIO5
	0x4905 8064		GPIO6
Description	Set to 1 the corresponding bits in the GPIO_IRQENABLE1 register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETIRQEN1																															

Bits	Field Name	Description	Type	Reset
31:0	SETIRQEN1	Set Interrupt Enable 1 0x0: no effect 0x1: Set the corresponding bit in the GPIO_IRQENABLE1 register	RW	0x00000000

Table 21-48. Register Call Summary for Register GPIO_SETIRQENABLE1

General-Purpose Interface Basic Programming Model • Set Instruction: [0]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-49. GPIO_CLEARIRQENABLE2

Address Offset	0x070		
Physical Address	0x4831 0070	**Instance**	GPIO1
	0x4905 0070		GPIO2
	0x4905 2070		GPIO3
	0x4905 4070		GPIO4
	0x4905 6070		GPIO5
	0x4905 8070		GPIO6
Description	Clear to 0 the corresponding bits in the GPIO_IRQENABLE2 register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLEARIRQEN2																															

Bits	Field Name	Description	Type	Reset
31:0	CLEARIRQEN2	Clear Interrupt Enable 2	RW	0x00000000
		0x0: no effect		
		0x1: Clear the corresponding bit in the GPIO_IRQENABLE2 register		

Table 21-50. Register Call Summary for Register GPIO_CLEARIRQENABLE2

General-Purpose Interface Basic Programming Model • Clear Instruction: [0]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-51. GPIO_SETIRQENABLE2

Address Offset	0x074		
Physical Address	0x4831 0074	**Instance**	GPIO1
	0x4905 0074		GPIO2
	0x4905 2074		GPIO3
	0x4905 4074		GPIO4
	0x4905 6074		GPIO5
	0x4905 8074		GPIO6
Description	Set to 1 the corresponding bits in the GPIO_IRQENABLE2 register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETIRQEN2																															

Bits	Field Name	Description	Type	Reset
31:0	SETIRQEN2	Set Interrupt Enable 2 0x0: no effect 0x1: Set the corresponding bit in the GPIO_IRQENABLE2 register	RW	0x00000000

Table 21-52. Register Call Summary for Register GPIO_SETIRQENABLE2

General-Purpose Interface Basic Programming Model • Set Instruction: [0]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-53. GPIO_CLEARWKUENA

Address Offset	0x080		
Physical Address	0x4831 0080	**Instance**	GPIO1
	0x4905 0080		GPIO2
	0x4905 2080		GPIO3
	0x4905 4080		GPIO4
	0x4905 6080		GPIO5
	0x4905 8080		GPIO6
Description	Clear to 0 the corresponding bits in the GPIO_WAKEUPENABLE register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLEARWAKEUPEN																															

Bits	Field Name	Description	Type	Reset
31:0	CLEARWAKEUPEN	Clear Wakeup Enable 0x0: no effect 0x1: Clear the corresponding bit in the GPIO_WAKEUPENABLE register	RW	0x00000000

Table 21-54. Register Call Summary for Register GPIO_CLEARWKUENA

General-Purpose Interface Basic Programming Model • Clear Instruction: [0]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-55. GPIO_SETWKUENA

Address Offset	0x084		
Physical Address	0x4831 0084	**Instance**	GPIO1
	0x4905 0084		GPIO2
	0x4905 2084		GPIO3
	0x4905 4084		GPIO4
	0x4905 6084		GPIO5
	0x4905 8084		GPIO6
Description	Set to 1 the corresponding bits in the GPIO_WAKEUPENABLE register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETWAKEUPEN																															

Bits	Field Name	Description	Type	Reset
31:0	SETWAKEUPEN	Set Wakeup Enable 0x0: no effect 0x1: Set the corresponding bit in the GPIO_WAKEUPENABLE register	RW	0x00000000

Table 21-56. Register Call Summary for Register GPIO_SETWKUENA

General-Purpose Interface Basic Programming Model • Set Instruction: [0]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [1] [2]

Table 21-57. GPIO_CLEARDATAOUT

Address Offset	0x090		
Physical Address	0x4831 0090	**Instance**	GPIO1
	0x4905 0090		GPIO2
	0x4905 2090		GPIO3
	0x4905 4090		GPIO4
	0x4905 6090		GPIO5
	0x4905 8090		GPIO6
Description	Clear to 0 the corresponding bits in the GPIO_DATAOUT register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLEARDATAOUT																															

Bits	Field Name	Description	Type	Reset
31:0	CLEARDATAOUT	Clear Data Output Register 0x0: no effect 0x1: Clear the corresponding bit in the GPIO_DATAOUT register	RW	0x00000000

Table 21-58. Register Call Summary for Register GPIO_CLEARDATAOUT

General-Purpose Interface Basic Programming Model • Clear Instruction: [0] • Data Input (Capture)/Output (Drive): [1]
General-Purpose Interface Register Manual • General-Purpose Interface Register Mapping Summary: [2] [3]

Table 21-59. GPIO_SETDATAOUT

Address Offset	0x094		
Physical Address	0x4831 0094	**Instance**	GPIO1
	0x4905 0094		GPIO2
	0x4905 2094		GPIO3
	0x4905 4094		GPIO4
	0x4905 6094		GPIO5
	0x4905 8094		GPIO6
Description	Set to 1 the corresponding bits in the GPIO_DATAOUT register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETDATAOUT																															

Bits	Field Name	Description	Type	Reset
31:0	SETDATAOUT	Set Data Output Register	RW	0x00000000
		0x0: no effect		
		0x1: Set the corresponding bit in the GPIO_DATAOUT register		

Table 21-60. Register Call Summary for Register GPIO_SETDATAOUT

General-Purpose Interface Basic Programming Model
- Set Instruction: [0]
- Data Input (Capture)/Output (Drive): [1]

General-Purpose Interface Register Manual
- General-Purpose Interface Register Mapping Summary: [2] [3]

Ethernet Media Access Controller (EMAC)/ Management Data Input/Output (MDIO) Module

22.1 Introduction

This document provides a functional description of the Ethernet Media Access Controller (EMAC) and physical layer (PHY) device Management Data Input/Output (MDIO) module integrated in the device. Included are the features of the EMAC and MDIO modules, a discussion of their architecture and operation, how these modules connect to the outside world, and a description of the registers for each module.

The EMAC controls the flow of packet data from the system to the PHY. The MDIO module controls PHY configuration and status monitoring.

Both the EMAC and the MDIO modules interface to the system core through a custom interface that allows efficient data transmission and reception. This custom interface is referred to as the EMAC subsystem module and is considered integral to the EMAC/MDIO peripheral.

22.1.1 Purpose of the Peripheral

The EMAC module is used to move data between the device and another host connected to the same network, in compliance with the Ethernet protocol.

22.1.2 Features

The EMAC/MDIO has the following features:

- Synchronous 10/100 Mbit operation.
- CBA3.1 compliant DMA controllers with VBUSP data transfers.
- Selectable RMII Interface.
- Hardware Error handling including CRC.
- Little and Big Endian Support.
- Eight receive channels with VLAN tag discrimination for receive hardware QOS support.
- Eight transmit channels with round-robin or fixed priority for hardware QOS support.
- CPPI 3.0 compliant.
- EtherStats and 802.3Stats RMON statistics gathering.
- Transmit CRC generation selectable on a per channel basis.
- Broadcast frames selectable for reception on a single channel.
- Multicast frames selectable for reception on a single channel.
- Promiscuous receive mode selectable for reception on a single channel (all frames, all good frames, short frames, error frames).
- TI Adaptive Performance Optimization for improved half duplex performance.
- Configurable receive address matching/filtering.
- Emulation Support.
- EMAC Loopback Mode.
- 8K (2048 x 32) internal CPPI buffer descriptor memory.
- MDIO module for PHY Management.

- Programmable interrupt control with selected interrupt pacing.

22.1.3 Functional Block Diagram

After reset, initialization, configuration, and auto-negotiation the host may initiate transmit and receive operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the EMAC STATERAM block. The EMAC transmit DMA controller then fetches the first packet in the packet chain from memory in accordance with the CPPI 3.0 protocol.

The EMAC receive DMA controller then writes packets to external memory in accordance with the CPPI 3.0 protocol. The EMAC is in configuration 2 (CFIG2). Configuration 2 is the 10/100 configuration with configurable address matching.

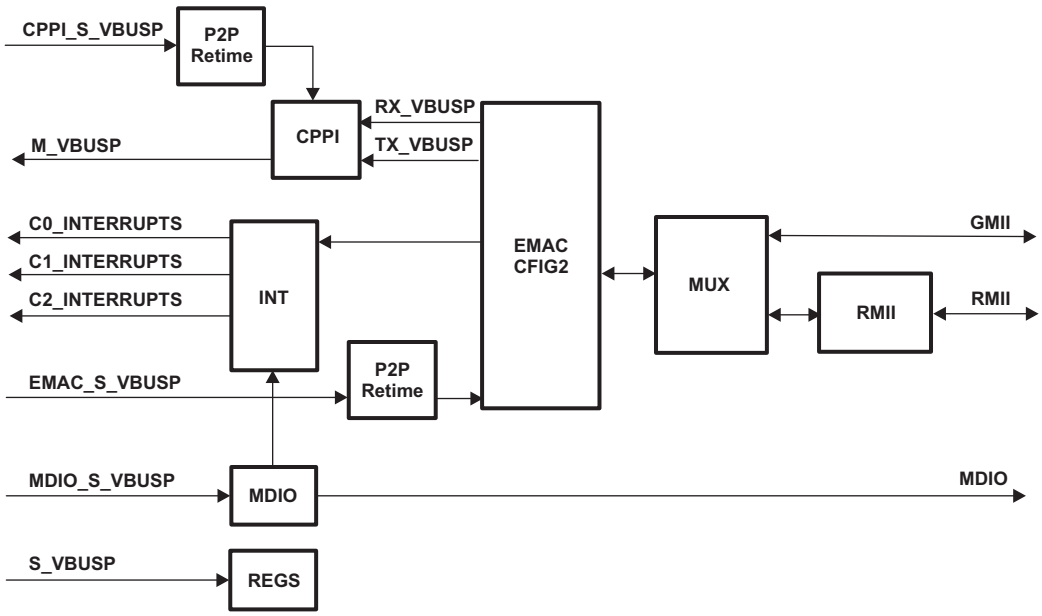
The CPPI block contains a 2048 by 32-bit memory (8K bytes) to be used for transmit and receive buffer descriptors. The CPPI block also contains a CBA 3.1 SCR (Switch Central Resource) to combine the EMAC transmit and receive VBUSP interfaces onto a single master VBUSP interface.

The host configures and initializes the device through the five module slave VBUSP interfaces. The CPPI, REGS, MDIO, and EMAC, modules have slave interfaces. The CPPI and EMAC slave interfaces are input directly to a VBUSP to VBUSP retiming bridge before being input to the switched central resource (SCR) inside the respective modules. The retiming bridges help to minimize long paths.

The SCR handles decode between multiple master/slaves inside the respective modules. The interrupt control block is used to select the interrupts from the EMAC and the MDIO for output to the three CPU's.

The signals required to support gigabit mode are included in the module. However gigabit transfer rates are not supported due to the small 3-cell transmit and receive FIFO's. The upper nibble of the GMII transmit and receive busses can be unused for MII pinout. Also, the RFTCLK input can be tied low for 10/100 operation.

Figure 22-1. EMAC and MDIO Block Diagram



22.1.4 Industry Standard(s) Compliance Statement

The EMAC peripheral conforms to the IEEE 802.3 standard, describing the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer specifications. The IEEE 802.3 standard has also been adopted by ISO/IEC and re-designated as ISO/IEC 8802-3:2000(E).

However, the EMAC deviates from the standard in the way it handles transmit underflow errors. The EMAC MII interface does not use the Transmit Coding Error signal MTXER. Instead of driving the error pin when an underflow condition occurs on a transmitted frame, the EMAC intentionally generates an incorrect checksum by inverting the frame CRC, so that the transmitted frame is detected as an error by the network.

22.2 Architecture

This section discusses the architecture and basic function of the EMAC peripheral.

22.2.1 Clock Control

All internal EMAC logic is clocked synchronously on one clock domain.

The MDIO clock is based on a divide-down of the peripheral clock and is specified to run up to 2.5 MHZ (although typical operation would be 1.0 MHz). Because the peripheral clock frequency is variable, the application software or driver must control the divide-down value. Interface clock is running at 166MHz and the VBUSP clock is running at 166MHz.

The clock sources are provided by the external PHY to the RMII reference clock pin. Data is transmitted and received with respect to the reference clocks of the interface pins.

The RMII interface frequency for the transmit and receive clocks are fixed at 50 MHz for both 10 Mbps and 100 Mbps.

22.2.2 Memory Map

The EMAC peripheral includes internal memory that is used to hold buffer descriptions of the Ethernet packets to be received and transmitted. This internal RAM is 2K × 32 bits in size. Data can be written to and read from the EMAC internal memory by either the EMAC or the CPU. It is used to store buffer descriptors that are 4-words (16-bytes) deep. This 8K local memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention. This EMAC RAM is also referred to as the CPPI buffer descriptor memory because it complies with the Communications Port Programming Interface (CPPI) v3.0 standard.

The packet buffer descriptors can also be placed in other on- and off-chip memories. There are some tradeoffs in terms of cache performance and throughput when descriptors are placed in the system memory, versus when they are placed in the EMAC's internal memory. In general, the EMAC throughput is better when the descriptors are placed in the local EMAC CPPI RAM.

22.2.2.1 CPPI Descriptors

The base address of CPPI descriptors is 5C02 0000h.

The 8K bytes of CPPI memory begin at address 01E2 0000h from the EMAC perspective.

22.2.2.2 EMAC Subsystem Module

The base address of EMAC subsystem is 5C00 0000h.

22.2.2.3 EMAC Module

The base address of EMAC module is 5C01 0000h.

22.2.2.4 MDIO Module

The base address of MDIO module is 5C03 0000h.

22.2.3 Signal Descriptions

22.2.3.1 RMII Receive (RX)

The CPRMII receive (RX) interface converts the input data from the external RMII PHY (or switch) into the required MII (EMAC) signals. The carrier sense and collision signals are determined from the RMII input data stream and transmit inputs as defined in the RMII specification.

An asserted RMII_RXER on any di-bit in the received packet will cause an MII_RXER assertion to the EMAC during the packet. In 10Mbps mode, the error is not required to be duplicated on 10 successive clocks. Any di-bit which has an asserted RMII_RXER during any of the 10 replications of the data will cause the error to be propagated.

Any received packet that ends with an improper nibble boundary aligned RMII_CRS_DV toggle will issue an MII_RXER during the packet to the EMAC. Also, a change in speed or duplex mode during packet operations will cause packet corruption.

The CPRMII can accept receive packets with shortened preambles, but 0x55 followed by a 0x5d is the shortest preamble that will be recognized (1 preamble byte with the start of frame byte). At least one byte of preamble with the start of frame indicator is required to begin a packet. An asserted RMIII_CRS_DV without at least a single correct preamble byte followed by the start of frame indicator will be ignored.

22.2.3.2 RMII Transmit (TX)

The CPRMII transmit (TX) interface converts the EMAC MII input data into the RMII transmit format. The data is then output to the external RMII PHY.

The EMAC does not source the transmit error (MII TXERR) signal. Any transmit frame from the EMAC with an error (i.e. underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be deasserted at all times and is not an input into the CPRMII module. Zeroes are output on RMII_TXD[1:0] for each clock that RMII_TXEN is deasserted.

22.2.3.3 Reduced Media Independent Interface (RMII) Connections

Figure 22-2 shows a device with integrated EMAC and MDIO interfaced via a RMII connection in a typical system.

The individual EMAC and MDIO signals for the RMII interface are summarized in Table 22-1. For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).

Figure 22-2. Ethernet Configuration—RMII Connections

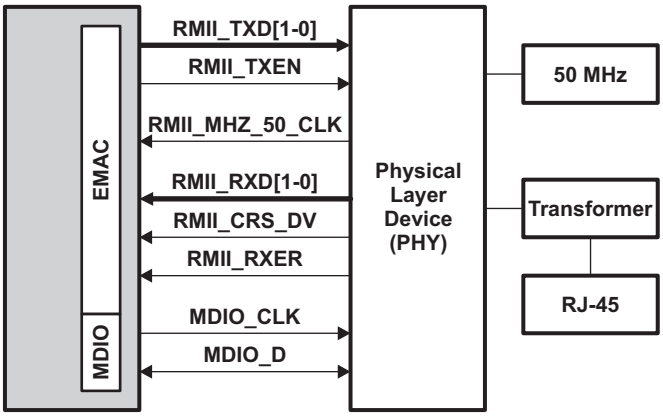


Table 22-1. EMAC and MDIO Signals for RMII Interface

Signal	Type	Description
RMII_TXD[1-0]	O	Transmit data (RMII_TXD). The transmit data pins are a collection of 2 bits of data. RMTDX0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMII_TXEN is asserted.

Table 22-1. EMAC and MDIO Signals for RMII Interface (continued)

Signal	Type	Description
RMII_TXEN	O	Transmit enable (RMII_TXEN). The transmit enable signal indicates that the RMII_TXD pins are generating data for use by the PHY. RMII_TXEN is synchronous to RMII_MHZ_50_CLK.
RMII_MHZ_50_CLK	I	RMII reference clock (RMII_MHZ_50_CLK). The reference clock is used to synchronize all RMII signals. RMII_MHZ_50_CLK must be continuous and fixed at 50 MHz.
RMII_RXD[1-0]	I	Receive data (RMII_RXD). The receive data pins are a collection of 2 bits of data. RMRDX0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMII_CRS_DV is asserted and RMII_RXER is deasserted.
RMII_CRS_DV	I	Carrier sense/receive data valid (RMII_CRS_DV). Multiplexed signal between carrier sense and receive data valid.
RMII_RXER	I	Receive error (RMII_RXER). The receive error signal is asserted to indicate that an error was detected in the received frame.
MDIO_CLK	O	Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin. The frequency of this clock is controlled by the CLKDIV bits in the MDIO control register (CONTROL).
MDIO_D	I/O	Management data input output (MDIO_D). The MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

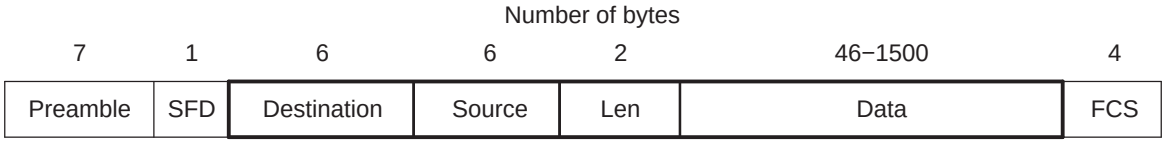
22.2.4 Ethernet Protocol Overview

A brief overview of the Ethernet protocol is given in the following subsections. See the IEEE 802.3 standard document for in-depth information on the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method.

22.2.4.1 Ethernet Frame Format

All the Ethernet technologies use the same frame structure. The format of an Ethernet frame is shown in Figure 22-3 and described in Table 22-2. The Ethernet packet, which is the collection of bytes representing the data portion of a single Ethernet frame on the wire, is shown outlined in bold. The Ethernet frames are of variable lengths, with no frame smaller than 64 bytes or larger than RXMAXLEN bytes (header, data, and CRC).

Figure 22-3. Ethernet Frame Format



Legend: SFD=Start Frame Delimeter; FCS=Frame Check Sequence (CRC)

Table 22-2. Ethernet Frame Description

Field	Bytes	Description
Preamble	7	Preamble. These 7 bytes have a fixed value of 55h and serve to wake up the receiving EMAC ports and to synchronize their clocks to that of the sender's clock.
SFD	1	Start of Frame Delimiter. This field with a value of 5Dh immediately follows the preamble pattern and indicates the start of important data.
Destination	6	Destination address. This field contains the Ethernet MAC address of the EMAC port for which the frame is intended. It may be an individual or multicast (including broadcast) address. When the destination EMAC port receives an Ethernet frame with a destination address that does not match any of its MAC physical addresses, and no promiscuous, multicast or broadcast channel is enabled, it discards the frame.
Source	6	Source address. This field contains the MAC address of the Ethernet port that transmits the frame to the Local Area Network.

Table 22-2. Ethernet Frame Description (continued)

Field	Bytes	Description
Len	2	Length/Type field. The length field indicates the number of EMAC client data bytes contained in the subsequent data field of the frame. This field can also be used to identify the type of data the frame is carrying.
Data	46 to (RXMAXLEN - 18)	Data field. This field carries the datagram containing the upper layer protocol frame, that is, IP layer datagram. The maximum transfer unit (MTU) of Ethernet is (RXMAXLEN - 18) bytes. This means that if the upper layer protocol datagram exceeds (RXMAXLEN - 18) bytes, then the host has to fragment the datagram and send it in multiple Ethernet packets. The minimum size of the data field is 46 bytes. This means that if the upper layer datagram is less then 46 bytes, the data field has to be extended to 46 bytes by appending extra bits after the data field, but prior to calculating and appending the FCS.
FCS	4	Frame Check Sequence. A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence covers the 60 to 1514 bytes of the packet data. Note that this 4-byte field may or may not be included as part of the packet data, depending on how the EMAC is configured.

22.2.4.2 Ethernet's Multiple Access Protocol

Nodes in an Ethernet Local Area Network are interconnected by a broadcast channel -- when an EMAC port transmits a frame, all the adapters on the local network receive the frame. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms are used when the EMAC operates in half-duplex mode. When operating in full-duplex mode, there is no contention for use of a shared medium because there are exactly two ports on the local network.

Each port runs the CSMA/CD protocol without explicit coordination with the other ports on the Ethernet network. Within a specific port, the CSMA/CD protocol works as follows:

1. The port obtains data from upper layer protocols at its node, prepares an Ethernet frame, and puts the frame in a buffer.
2. If the port senses that the medium is idle, it starts to transmit the frame. If the port senses that the transmission medium is busy, it waits until it no longer senses energy (plus an Inter-Packet Gap time) and then starts to transmit the frame.
3. While transmitting, the port monitors for the presence of signal energy coming from other ports. If the port transmits the entire frame without detecting signal energy from other Ethernet devices, the port is done with the frame.
4. If the port detects signal energy from other ports while transmitting, it stops transmitting its frame and instead transmits a 48-bit jam signal.
5. After transmitting the jam signal, the port enters an exponential backoff phase. If a data frame encounters back-to-back collisions, the port chooses a random value that is dependent on the number of collisions. The port then waits an amount of time that is a multiple of this random value and returns to step 2.

22.2.5 Programming Interface of Packet Descriptors

22.2.5.1 CPPI Packet Buffer Descriptors

The buffer descriptor is a central part of the EMAC module and is how the application software describes Ethernet packets to be sent and empty buffers to be filled with incoming packet data. The basic descriptor format is shown in Figure 22-4 and described in Table 22-3.

For example, consider three packets to be transmitted: Packet A is a single fragment (60 bytes), Packet B is fragmented over three buffers (1514 bytes total), and Packet C is a single fragment (1514 bytes). The linked list of descriptors to describe these three packets is shown in Figure 22-5.

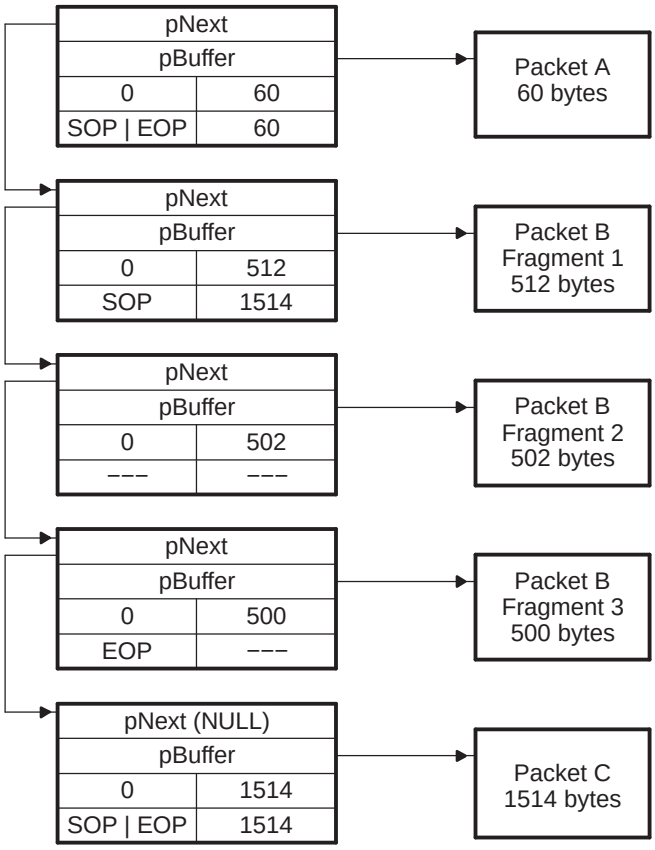
Figure 22-4. Basic Descriptor Format

	Bit Fields	
Word Offset	**31 16**	**15 0**
0	Next Descriptor Pointer	
1	Buffer Pointer	
2	Buffer Offset	Buffer Length
3	Flags	Packet Length

Table 22-3. Basic Descriptor Description

Word Offset	Field	Field Description
0	Next Descriptor Pointer	The next descriptor pointer is used to create a single linked list of descriptors. Each descriptor describes a packet or a packet fragment. When a descriptor points to a single buffer packet or the first fragment of a packet, the start of packet (SOP) flag is set in the flags field. When a descriptor points to a single buffer packet or the last fragment of a packet, the end of packet (EOP) flag is set. When a packet is fragmented, each fragment must have its own descriptor and appear sequentially in the descriptor linked list.
1	Buffer Pointer	The buffer pointer refers to the actual memory buffer that contains packet data during transmit operations, or is an empty buffer ready to receive packet data during receive operations.
2	Buffer Offset	The buffer offset is the offset from the start of the packet buffer to the first byte of valid data. This field only has meaning when the buffer descriptor points to a buffer that actually contains data.
	Buffer Length	The buffer length is the actual number of valid packet data bytes stored in the buffer. If the buffer is empty and waiting to receive data, this field represents the size of the empty buffer.
3	Flags	The flags field contains more information about the buffer, such as, is it the first fragment in a packet (SOP), the last fragment in a packet (EOP), or contains an entire contiguous Ethernet packet (both SOP and EOP). The flags are described in Section 22.2.5.4 and Section 22.2.5.5.
	Packet Length	The packet length only has meaning for buffers that both contain data and are the start of a new packet (SOP). In the case of SOP descriptors, the packet length field contains the length of the entire Ethernet packet, regardless if it is contained in a single buffer or fragmented over several buffers.

Figure 22-5. Typical Descriptor Linked List



22.2.5.2 Transmit and Receive Descriptor Queues

The EMAC module processes descriptors in linked lists as discussed in Section 22.2.5.1. The lists used by the EMAC are maintained by the application software through the use of the head descriptor pointer registers (HDP). Since the EMAC supports eight channels for transmit and receive, there are eight head descriptor pointer registers for both transmit and receive. They are:

- TX*n*HDP - Transmit Channel *n* DMA Head Descriptor Pointer Register
- RX*n*HDP - Receive Channel *n* DMA Head Descriptor Pointer Register

After an EMAC reset and before enabling the EMAC for send and receive, all 16 head descriptor pointer registers must be initialized to 0.

The EMAC uses a simple system to determine if a descriptor is currently owned by the EMAC or by the application software. There is a flag in the buffer descriptor flags called OWNER. When this flag is set, the packet that is referenced by the descriptor is considered to be owned by the EMAC. Note that ownership is done on a packet based granularity, not on descriptor granularity, so only SOP descriptors make use of the OWNER flag. As packets are processed, the EMAC patches the SOP descriptor of the corresponding packet and clears the OWNER flag. This is an indication that the EMAC has finished processing all descriptors up to and including the first with the EOP flag set, indicating the end of the packet (note this may only be one descriptor with both the SOP and EOP flags set).

To add a descriptor or a linked list of descriptors to an EMAC descriptor queue for the first time, the software application simply writes the pointer to the descriptor or first descriptor of a list to the corresponding HDP register. Note that the last descriptor in the list must have its “next” pointer cleared to 0. This is the only way the EMAC has of detecting the end of the list. Therefore, in the case where only a single descriptor is added, its “next descriptor” pointer must be initialized to 0.

The HDP must never be written to while a list is active. To add additional descriptors to a descriptor list already owned by the EMAC, the NULL “next” pointer of the last descriptor of the previous list is patched with a pointer to the first descriptor of the new list. The list of new descriptors to be appended to the existing list must itself be NULL terminated before the pointer patch is performed.

There is a potential race condition where the EMAC may read the “next” pointer of a descriptor as NULL in the instant before an application appends additional descriptors to the list by patching the pointer. This case is handled by the software application always examining the buffer descriptor flags of all EOP packets, looking for a special flag called end of queue (EOQ). The EOQ flag is set by the EMAC on the last descriptor of a packet when the descriptor’s “next” pointer is NULL. This is the way the EMAC indicates to the software application that it believes it has reached the end of the list. When the software application sees the EOQ flag set, the application may at that time submit the new list, or the portion of the appended list that was missed by writing the new list pointer to the same HDP that started the process.

This process applies when adding packets to a transmit list, and empty buffers to a receive list.

22.2.5.3 Transmit and Receive EMAC Interrupts

The EMAC processes descriptors in linked list chains as discussed in Section 22.2.5.1, using the linked list queue mechanism discussed in Section 22.2.5.2.

The EMAC synchronizes descriptor list processing through the use of interrupts to the software application. The interrupts are controlled by the application using the interrupt masks, global interrupt enable, and the completion pointer register (CP). The CP is also called the interrupt acknowledge register.

As the EMAC supports eight channels for both transmit and receive, there are eight completion pointer registers for both. They are:

- TX*n*CP - Transmit Channel *n* Completion Pointer (Interrupt Acknowledge) Register
- RX*n*CP - Receive Channel *n* Completion Pointer (Interrupt Acknowledge) Register

These registers serve two purposes. When read, they return the pointer to the last descriptor that the EMAC has processed. When written by the software application, the value represents the last descriptor processed by the software application. When these two values do not match, the interrupt is active.

Interrupts in the interrupt control module of EMAC subsystem are routed to three independent interrupt cores which are then mapped to CPU interrupt controllers. The system configuration determines whether or not an active interrupt actually interrupts the CPU. In general the following settings are required for basic EMAC transmit and receive interrupts:

1. EMAC transmit and receive interrupts are enabled by setting the mask registers RXINTMASKSET and TXINTMASKSET
2. Global interrupts for the appropriate interrupt core registers are set in the EMAC interrupt control module: C*n*RXEN and C*n*TXEN on core *n*
3. The CPU interrupt controller is configured to accept C*n*_RX_PULSE and C*n*_TX_PULSE interrupts from the interrupt control module of EMAC subsystem

Whether or not the interrupt is enabled, the current state of the receive or transmit channel interrupt can be examined directly by the software application reading the EMAC receive interrupt status (unmasked) register (RXINTSTATRAW) and transmit interrupt status (unmasked) register (TXINTSTATRAW).

After servicing transmit or receive interrupts, the application software must acknowledge both the EMAC and EMAC subsystem interrupts.

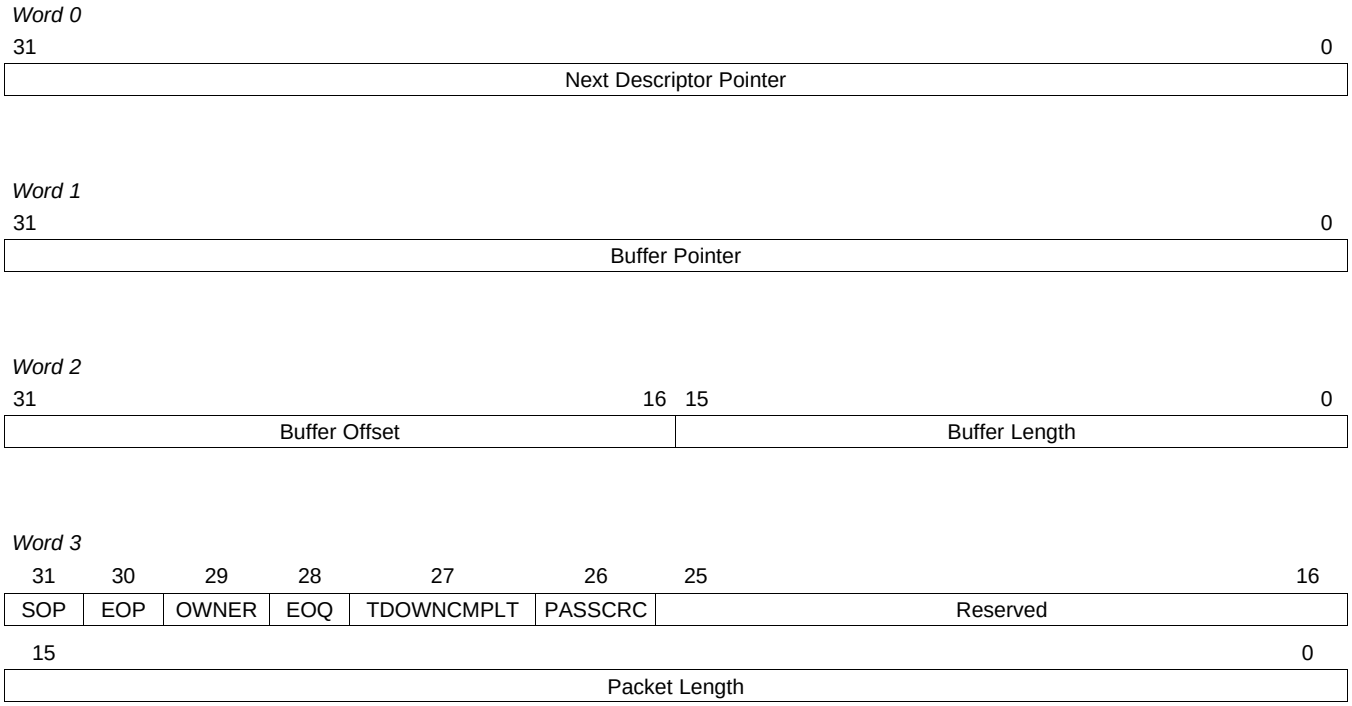
EMAC interrupts are acknowledged when the application software updates the value of TX*n*CP or RX*n*CP with a value that matches the internal value kept by the EMAC. This mechanism ensures that the application software never misses an EMAC interrupt because the interrupt acknowledgment is tied directly to the buffer descriptor processing.

EMAC module interrupts are acknowledged when the application software writes the appropriate C*n*TX or C*n*RX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). The MACEOIVECTOR behaves as an interrupt pulse interlock -- once the EMAC interrupt control module has issued an interrupt pulse to the CPU, it will not generate further pulses of the same type until the original pulse has been acknowledged.

22.2.5.4 CPPI Transmit Buffer Descriptor Format

Transmit buffer descriptors contain a minimum of 16 bytes (4 words) and are 32-bit aligned structures located in host memory. Tx buffer descriptors provide information about the descriptor data buffer and channel queue. Four words are required, but protocol specific control information may be added by using as many extra words as necessary. Transmit buffer descriptors may be linked together to form packets. Buffer descriptor SOP and EOP bits are used to delimit packets. Packets in turn may be linked together to form transmit queues. Each queue consists of a chain of buffer descriptors linked together by Next Descriptor Pointers. The last buffer descriptor in a queue has a zero Next Descriptor Pointer. Each descriptor points to a data buffer yielding a queue of buffers. Each transmit buffer descriptor contains at least four 32-bit words.

Figure 22-6. Transmit Buffer Descriptor Format



Example 22-1. Transmit Buffer Descriptor in C Structure Format

```
/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
  struct _EMAC_Desc *pNext;     /* Pointer to next descriptor in chain  */
  Uint8             *pBuffer;   /* Pointer to data buffer               */
  Uint32            BufOffLen;  /* Buffer Offset(MSW) and Length(LSW)   */
  Uint32            PktFlgLen;  /* Packet Flags(MSW) and Length(LSW)    */
} EMAC_Desc;
/* Packet Flags                                                        */
#define EMAC_DSC_FLAG_SOP               0x80000000u
#define EMAC_DSC_FLAG_EOP               0x40000000u
#define EMAC_DSC_FLAG_OWNER             0x20000000u
#define EMAC_DSC_FLAG_EOQ               0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT        0x08000000u
#define EMAC_DSC_FLAG_PASSCRC           0x04000000u
```

22.2.5.4.1 Next Descriptor Pointer

The next descriptor pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the transmit queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active transmit queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more packets need to be queued for transmit, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the transmitter will halt on the transmit channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

22.2.5.4.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

22.2.5.4.3 Buffer Offset

This 16-bit field indicates how many unused bytes are at the start of the buffer. For example, a value of 0000h indicates that no unused bytes are at the start of the buffer and that valid data begins on the first byte of the buffer, while a value of 000Fh indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

Note that this value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set). It can not be used to specify the offset of subsequent packet fragments. Also, since the buffer pointer may point to any byte–aligned address, this field may be entirely superfluous, depending on the device driver architecture.

The range of legal values for this field is 0 to (Buffer Length – 1).

22.2.5.4.4 Buffer Length

This 16-bit field indicates how many valid data bytes are in the buffer. On single fragment packets, this value is also the total length of the packet data to be transmitted. If the buffer offset field is used, the offset bytes are not counted as part of this length. This length counts only valid data bytes. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

22.2.5.4.5 Packet Length

This 16-bit field specifies the number of data bytes in the entire packet. Any leading buffer offset bytes are not included. The sum of the buffer length fields of each of the packet's fragments (if more than one) must be equal to the packet length. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC. This value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set).

22.2.5.4.6 Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

22.2.5.4.7 End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

22.2.5.4.8 Ownership (OWNER) Flag

When set, this flag indicates that all the descriptors for the given packet (from SOP to EOP) are currently owned by the EMAC. This flag is set by the software application on the SOP packet descriptor before adding the descriptor to the transmit descriptor queue. For a single fragment packet, the SOP, EOP, and OWNER flags are all set. The OWNER flag is cleared by the EMAC once it is finished with all the descriptors for the given packet. Note that this flag is valid on SOP descriptors only.

22.2.5.4.9 End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the transmit queue for a given transmit channel, and that the transmitter has halted. This flag is initially cleared by the software application prior to adding the descriptor to the transmit queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet (the EOP flag is set), and there are no more descriptors in the transmit list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC transmitter for the corresponding channel has halted. This is useful when the application appends additional packet descriptors to a transmit queue list that is already owned by the EMAC. Note that this flag is valid on EOP descriptors only.

22.2.5.4.10 Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a transmit queue is being torn down, or aborted, instead of allowing it to be transmitted. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the SOP descriptor of each packet as it is aborted from transmission.

Note that this flag is valid on SOP descriptors only. Also note that only the first packet in an unsent list has the TDOWNCMPLT flag set. Subsequent descriptors are not processed by the EMAC.

22.2.5.4.11 Pass CRC (PASSCRC) Flag

This flag is set by the software application in the SOP packet descriptor before it adds the descriptor to the transmit queue. Setting this bit indicates to the EMAC that the 4 byte Ethernet CRC is already present in the packet data, and that the EMAC should not generate its own version of the CRC.

When the CRC flag is cleared, the EMAC generates and appends the 4-byte CRC. The buffer length and packet length fields do not include the CRC bytes. When the CRC flag is set, the 4-byte CRC is supplied by the software application and is already appended to the end of the packet data. The buffer length and packet length fields include the CRC bytes, as they are part of the valid packet data. Note that this flag is valid on SOP descriptors only.

Table 22-4. Tx Buffer Descriptor Word 0

Bits	Name	Description
31:0	Next DescriptorPointer	Next Descriptor Pointer: The 32-bit word aligned memory address of the next buffer descriptor in the Tx queue. This is the mechanism used to reference the next buffer descriptor from the current buffer descriptor. If the value of this pointer is zero then the current buffer is the last buffer in the queue. The host sets the Next Descriptor Pointer.

Table 22-5. Tx Buffer Descriptor Word 1

Bits	Name	Description
31:0	Buffer Pointer	Buffer Pointer: The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer desc. The host sets the Buffer Pointer.

Table 22-6. Tx Buffer Descriptor Word 2

Bits	Name	Description
31:16	Buffer Offset	Buffer Offset: The Buffer Offset indicates how many unused bytes are at the start of the buffer (SOP buffers only). A value of zero indicates that there are no unused bytes at the start of the buffer and that valid data begins on the first byte of the buffer. A value of 000fh (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer. The Buffer Offset is valid only on Start Of Packet buffer descriptors and must be zero otherwise. The host sets the Buffer Offset. The Buffer Offset must be less than the Buffer Length.
15:0	Buffer Length	Buffer Length: The Buffer Length field indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. The host sets the Buffer Length.

Table 22-7. Tx Buffer Descriptor Word 3

Bits	Name	Description
31	SOP	Start of Packet: SOP Indicates that the descriptor buffer is the first buffer in the packet. The host sets the SOP bit. 0 = Not start of packet buffer 1 = Start of packet buffer
30	EOP	End of Packet: EOP Indicates that the descriptor buffer is the last buffer in the packet. The host sets the EOP bit. 0 = Not end of packet buffer. 1 = End of packet buffer.
29	Ownership	Ownership: The Ownership bit indicates ownership of the packet and is valid only on SOP. This bit is set by the host and cleared by the port when the packet has been transmitted. The host uses this bit to reclaim buffers. 0 = The packet is owned by the host 1 = The packet is owned by the port
28	EOQ	End Of Queue: The End of Queue bit is set by the port to indicate that all packets in the queue have been transmitted and the Tx queue is empty. This bit is valid only on EOP. 0 = The Tx queue has more packets to transfer. 1 = The port took this descriptor buffer as the last buffer descriptor in the last packet in the queue.
27:16	Protocol Specific	
15:0	Packet Length	Packet Length: The length of the packet in bytes. This field is valid only on SOP and is written by the host. If the Packet Length is less than the sum of the buffer lengths, then the packet data will be truncated. A Packet Length greater than the sum of the buffers is a host error.

22.2.5.5 CPPI Receive Buffer Descriptor Format

Receive buffer descriptors contain a minimum of 16 bytes (4 words) and are 32-bit aligned structures located in host memory. Rx buffer descriptors provide information about the associated data buffer and Rx queue. Four words are required, but protocol specific control information may be added using as many extra words as necessary. The protocol specific control information is valid only on SOP buffers.

Receive buffer descriptors contain a minimum of 16 bytes (4 words) and are 32-bit aligned structures located in host memory. Rx buffer descriptors provide information about the associated data buffer and Rx queue. Four words are required, but protocol specific control information may be added using as many extra words as necessary. The protocol specific control information is valid only on SOP buffers. Receive buffer descriptors may be linked together to form packets. Buffer descriptor SOP and EOP bits are used to delimit packets. Packets in turn are linked together to form receive queues (although an Rx queue may consist of just one packet). Each queue consists of a chain of buffer descriptors linked together by Next Descriptor Pointers. The last buffer descriptor in a queue has a zero Next Descriptor Pointer. Each descriptor points to a data buffer yielding a queue of buffers. Each receive buffer descriptor contains at least four 32-bit words, as described below:

Table 22-8. Rx Buffer Descriptor Word 0

Bits	Name	Description
31:0	Next DescriptorPointer	Next Descriptor Pointer: The 32-bit word aligned memory address of the next buffer descriptor in the Rx queue. This is the mechanism used to reference the next buffer descriptor from the current buffer descriptor. The current buffer is the last buffer in the queue if the value of this pointer is zero. The host sets the Next Descriptor Pointer

Table 22-9. Rx Buffer Descriptor Word 1

Bits	Name	Description
31:0	Buffer Pointer	Buffer Pointer: The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. The host sets the Buffer Pointer.

Table 22-10. Rx Buffer Descriptor Word 2

Bits	Name	Description
31:16	Buffer Offset	Buffer Offset: The Buffer Offset indicates how many unused bytes are at the beginning of the SOP buffer. A value of 0x0000 indicates that there are no unused bytes at the beginning of the buffer and that valid data begins on the first byte of the buffer. A value of 0x000f (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer. The host sets the buffer offset to zero on buffer initialization and the port overwrites the zero value on SOP packets with the Rx DMA State buffer offset value.
15:0	Buffer Length	Buffer Length: The Buffer Length field indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. The host sets the buffer length on buffer initialization. The port will overwrite the host initialized value on an EOP buffer when the number of received data bytes is less than the host initiated value. The port will overwrite the host initialized value on SOP when the Buffer Offset is greater than zero, or the Packet Length is less than the buffer length.

Table 22-11. Rx Buffer Descriptor Word 3

Bits	Name	Description
31	SOP	Start of Packet: SOP Indicates that the descriptor buffer is the first buffer in the packet. The host sets the SOP bit. 0 = Not start of packet buffer 1 = Start of packet buffer
30	EOP	End of Packet: EOP Indicates that the descriptor buffer is the last buffer in the packet. The host sets the EOP bit. 0 = Not end of packet buffer. 1 = End of packet buffer.
29	Ownership	Ownership: The Ownership bit indicates ownership of the packet and is valid only on SOP. This bit is set by the host and cleared by the port when the packet has been received. 0 = The packet is owned by the host. 1 = The packet is owned by the port.
28	EOQ	End Of Queue: The End of Queue bit is set by the port to indicate that the queue empty condition exists. This bit is valid only on EOP. 0 = The Rx queue has more buffers available for reception. 1 = The Descriptor buffer is the last buffer in the last packet in the queue.
27:16	Protocol Specific	
15:0	Packet Length	Packet Length: The length of the packet in bytes. This field is valid only on SOP and is written by the port at the end of the packet reception.

22.2.5.5.1 Next Descriptor Pointer

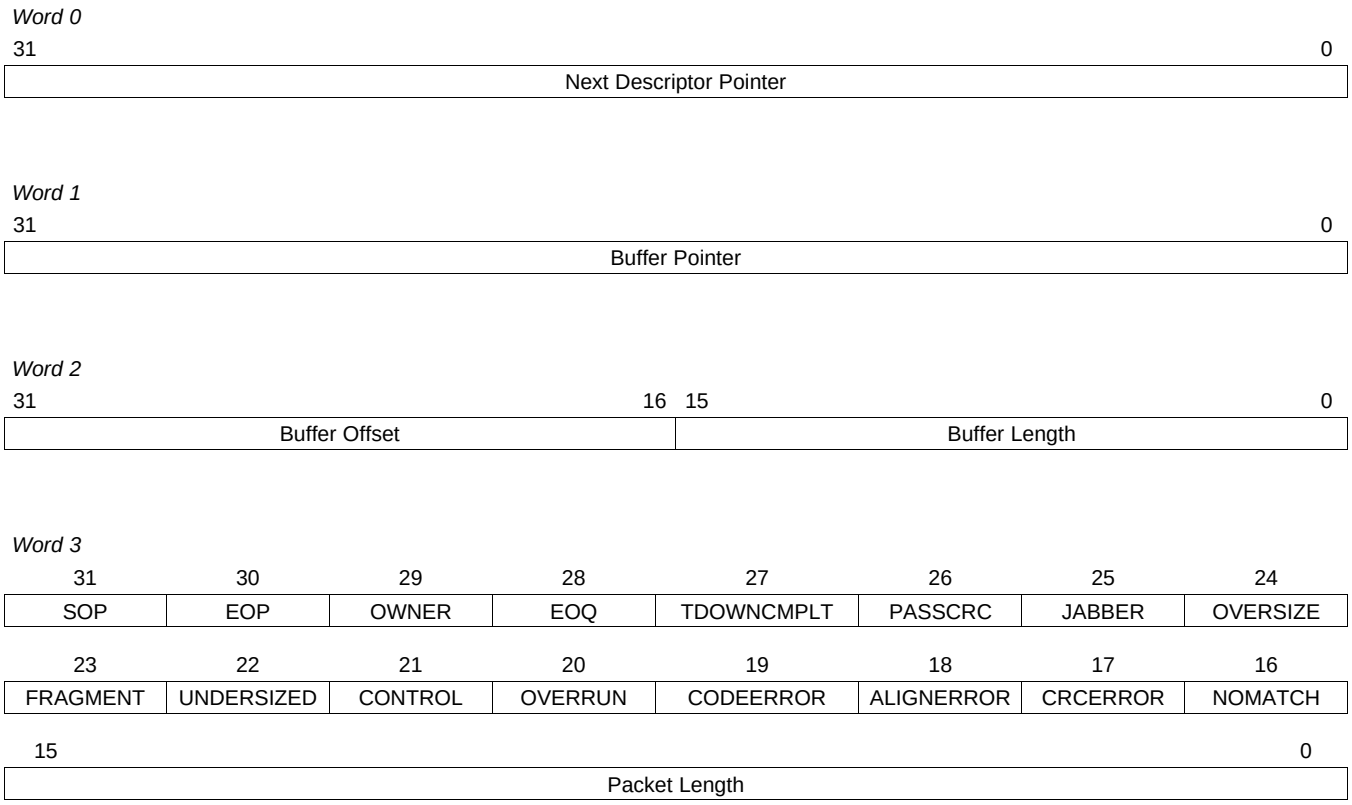
This pointer points to the 32–bit word aligned memory address of the next buffer descriptor in the receive queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active receive queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more empty buffers can be added to the pool, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the receiver will halt the receive channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

22.2.5.5.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

Figure 22-7. Receive Buffer Descriptor Format



Example 22-2. Receive Buffer Descriptor in C Structure Format

```
/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
  struct _EMAC_Desc *pNext;     /* Pointer to next descriptor in chain  *
```

Example 22-2. Receive Buffer Descriptor in C Structure Format (continued)

```
  struct _EMAC_Desc *pNext;     /* Pointer to next descriptor in chain  *
  Uint32            BufOffLen;  /* Buffer Offset(MSW) and Length(LSW)   */
  Uint32
PktFlgLen;  /* Packet Flags(MSW) and Length(LSW)    */
} EMAC_Desc;

/* Packet Flags                                                        */
#define EMAC_DSC_FLAG_SOP               0x80000000u
#define EMAC_DSC_FLAG_EOP               0x40000000u
#define EMAC_DSC_FLAG_OWNER             0x20000000u
#define EMAC_DSC_FLAG_EOQ               0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT        0x08000000u
#define EMAC_DSC_FLAG_PASSCRC           0x04000000u
#define EMAC_DSC_FLAG_JABBER            0x02000000u
#define EMAC_DSC_FLAG_OVERSIZE          0x01000000u
#define EMAC_DSC_FLAG_FRAGMENT          0x00800000u
#define EMAC_DSC_FLAG_UNDERSIZED        0x00400000u
#define EMAC_DSC_FLAG_CONTROL           0x00200000u
#define EMAC_DSC_FLAG_OVERRUN           0x00100000u
#define EMAC_DSC_FLAG_CODEERROR         0x00080000u
#define EMAC_DSC_FLAG_ALIGNERROR        0x00040000u
#define EMAC_DSC_FLAG_CRCERROR          0x00020000u
#define EMAC_DSC_FLAG_NOMATCH           0x00010000u
```

22.2.5.5.3 Buffer Offset

This 16-bit field must be initialized to zero by the software application before adding the descriptor to a receive queue.

Whether or not this field is updated depends on the setting of the RXBUFFEROFFSET register. When the offset register is set to a non-zero value, the received packet is written to the packet buffer at an offset given by the value of the register, and this value is also written to the buffer offset field of the descriptor.

When a packet is fragmented over multiple buffers because it does not fit in the first buffer supplied, the buffer offset only applies to the first buffer in the list, which is where the start of packet (SOP) flag is set in the corresponding buffer descriptor. In other words, the buffer offset field is only updated by the EMAC on SOP descriptors.

The range of legal values for the BUFFEROFFSET register is 0 to (Buffer Length – 1) for the smallest value of buffer length for all descriptors in the list.

22.2.5.5.4 Buffer Length

This 16-bit field is used for two purposes:

- Before the descriptor is first placed on the receive queue by the application software, the buffer length field is first initialized by the software to have the physical size of the empty data buffer pointed to by the buffer pointer field.
- After the empty buffer has been processed by the EMAC and filled with received data bytes, the buffer length field is updated by the EMAC to reflect the actual number of valid data bytes written to the buffer.

22.2.5.5.5 Packet Length

This 16-bit field specifies the number of data bytes in the entire packet. This value is initialized to zero by the software application for empty packet buffers. The value is filled in by the EMAC on the first buffer used for a given packet. This is signified by the EMAC setting a start of packet (SOP) flag. The packet length is set by the EMAC on all SOP buffer descriptors.

22.2.5.5.6 Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on SOP descriptors.

22.2.5.5.7 End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on EOP descriptors.

22.2.5.5.8 Ownership (OWNER) Flag

When set, this flag indicates that the descriptor is currently owned by the EMAC. This flag is set by the software application before adding the descriptor to the receive descriptor queue. This flag is cleared by the EMAC once it is finished with a given set of descriptors, associated with a received packet. The flag is updated by the EMAC on SOP descriptor only. So when the application identifies that the OWNER flag is cleared on an SOP descriptor, it may assume that all descriptors up to and including the first with the EOP flag set have been released by the EMAC. (Note that in the case of single buffer packets, the same descriptor will have both the SOP and EOP flags set.)

22.2.5.5.9 End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the receive queue for a given receive channel, and that the corresponding receiver channel has halted. This flag is initially cleared by the software application prior to adding the descriptor to the receive queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet received (also sets the EOP flag), and there are no more descriptors in the receive list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC receiver for the corresponding channel has halted. This is useful when the application appends additional free buffer descriptors to an active receive queue. Note that this flag is valid on EOP descriptors only.

22.2.5.5.10 Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a receive queue is being torn down, or aborted, instead of being filled with received data. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the descriptor of the first free buffer when the tear down occurs. No additional queue processing is performed.

22.2.5.5.11 Pass CRC (PASSCRC) Flag

This flag is set by the EMAC in the SOP buffer descriptor if the received packet includes the 4-byte CRC. This flag should be cleared by the software application before submitting the descriptor to the receive queue.

22.2.5.5.12 Jabber Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is a jabber frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE. Jabber frames are frames that exceed the RXMAXLEN in length, and have CRC, code, or alignment errors.

22.2.5.5.13 Oversize Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an oversized frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

22.2.5.5.14 Fragment Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is only a packet fragment and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

22.2.5.5.15 Undersized Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is undersized and was not discarded because the RXCSFEN bit was set in the RXMBPENABLE.

22.2.5.5.16 Control Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an EMAC control frame and was not discarded because the RXCMFEN bit was set in the RXMBPENABLE.

22.2.5.5.17 Overrun Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet was aborted due to a receive overrun.

22.2.5.5.18 Code Error (CODEERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a code error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

22.2.5.5.19 Alignment Error (ALIGNERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained an alignment error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

22.2.5.5.20 CRC Error (CRCERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a CRC error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

22.2.5.5.21 No Match (NOMATCH) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet did not pass any of the EMAC's address match criteria and was not discarded because the RXCAFEN bit was set in the RXMBPENABLE. Although the packet is a valid Ethernet data packet, it was only received because the EMAC is in promiscuous mode.

22.2.6 MDIO Module

The MDIO module is used to manage up to 32 physical layer (PHY) devices connected to the Ethernet Media Access Controller (EMAC). The device supports a single PHY being connected to the EMAC at any given time. The MDIO module is designed to allow almost transparent operation of the MDIO interface with little maintenance from the CPU.

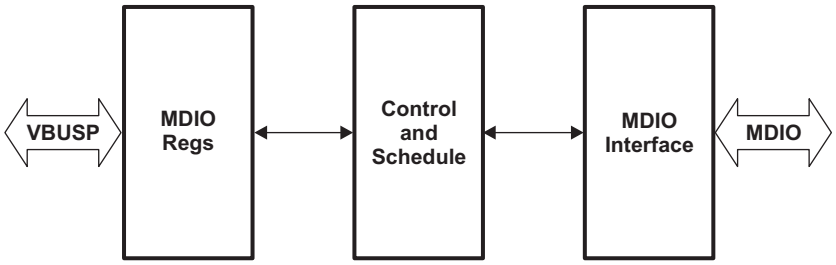
The MDIO module continuously polls 32 MDIO addresses in order to enumerate all PHY devices in the system. Once a PHY device has been detected, the MDIO module reads the MDIO PHY link status register (LINK) to monitor the PHY link state. Link change events are stored in the MDIO module, which can interrupt the CPU. This storing of the events allows the CPU to poll the link status of the PHY device without continuously performing MDIO module accesses. However, when the CPU must access the MDIO module for configuration and negotiation, the MDIO module performs the MDIO read or write operation independent of the CPU. This independent operation allows the processor to poll for completion or interrupt the CPU once the operation has completed.

22.2.6.1 MDIO Module Components

The VBUS MII management interface module consists of the following blocks:

- MDIO Regs
- Control and Schedule
- MDIO Interface

Figure 22-8. VBUS MII Management Interface Module



22.2.6.1.1 MDIO Regs

The MDIO regs block provides a VBUSP 2.0 compliant slave interface to the MII Management Interface module. Host interaction with this module is facilitated through the registers in this block.

22.2.6.1.2 Control and Schedule

The control and register logic in the MII Management Interface module contain the state machine and scheduling logic which control the wire side operation.

22.2.6.1.3 MDIO Interface

The MDIO interface block provides the serial interface to the MDIO interface.

22.2.6.2 MDIO Module Operational Overview

The MDIO module implements the 802.3 serial management interface to interrogate and control an Ethernet PHY, using a shared two-wired bus. It separately performs autodetection and records the current link status of up to 32 PHYs, polling all 32 MDIO addresses.

Application software uses the MDIO module to configure the autonegotiation parameters of the PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC.

In this device, the Ethernet PHY attached to the system can be directly controlled and queried. The Media Independent Interface (MII) address of this PHY device is specified in one of the PHYADRMON bits in the MDIO user PHY select register (USERPHYSEL*n*). The MDIO module can be programmed to trigger a CPU interrupt on a PHY link change event, by setting the LINKINTENB bit in USERPHYSEL*n*. Reads and writes to registers in this PHY device are performed using the MDIO user access register (USERACCESS*n*).

The MDIO module powers-up in an idle state until specifically enabled by setting the ENABLE bit in the MDIO control register (CONTROL). At this time, the MDIO clock divider and preamble mode selection are also configured. The MDIO preamble is enabled by default, but can be disabled when the connected PHY does not require it. Once the MDIO module is enabled, the MDIO interface state machine continuously polls the PHY link status (by reading the generic status register) of all possible 32 PHY addresses and records the results in the MDIO PHY alive status register (ALIVE) and MDIO PHY link status register (LINK). The corresponding bit for the connected PHY (0-31) is set in ALIVE, if the PHY responded to the read request. The corresponding bit is set in LINK, if the PHY responded and also is currently linked. In addition, any PHY register read transactions initiated by the application software using USERACCESS*n* causes ALIVE to be updated.

The USERPHYSEL*n* is used to track the link status of the connected PHY address. A change in the link status of the PHY being monitored sets the appropriate bit in the MDIO link status change interrupt registers (LINKINTRAW and LINKINTMASKED), if enabled by the LINKINTENB bit in USERPHYSEL*n*.

While the MDIO module is enabled, the host issues a read or write transaction over the MII management interface using the DATA, PHYADR, REGADR, and WRITE bits in USERACCESS*n*. When the application sets the GO bit in USERACCESS*n*, the MDIO module begins the transaction without any further intervention from the CPU. Upon completion, the MDIO module clears the GO bit and sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. The corresponding USERINTMASKED bit (0 or 1) in the MDIO user command complete interrupt register (USERINTMASKED) may also be set, depending on the mask setting configured in the MDIO user command complete interrupt mask set register (USERINTMASKSET) and the MDIO user interrupt mask clear register (USERINTMASKCLEAR).

A round-robin arbitration scheme is used to schedule transactions that may be queued using both USERACCESS0 and USERACCESS1. The application software must check the status of the GO bit in USERACCESS*n* before initiating a new transaction, to ensure that the previous transaction has completed. The application software can use the ACK bit in USERACCESS*n* to determine the status of a read transaction.

22.2.6.2.1 Initializing the MDIO Module

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO control register (CONTROL).
2. Enable the MDIO module by setting the ENABLE bit in CONTROL.
3. The MDIO PHY alive status register (ALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (LINK) can determine whether this PHY already has a link.
4. Setup the appropriate PHY addresses in the MDIO user PHY select register (USERPHYSEL*n*), and set the LINKINTENB bit to enable a link change event interrupt if desirable.
5. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) to use the MDIO user access register (USERACCESS*n*). Since only one PHY is used in this device, the application software can use one USERACCESS*n* to trigger a completion interrupt; the other USERACCESS*n* is not setup.

22.2.6.2.2 Writing Data To a PHY Register

The MDIO module includes a user access register (USERACCESS*n*) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS*n*) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in USERACCESS*n* corresponding to the PHY and PHY register you want to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in USERACCESS*n* for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

22.2.6.2.3 Reading Data From a PHY Register

The MDIO module includes a user access register (USERACCESS*n*) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS*n*) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in USERACCESS*n* corresponding to the PHY and PHY register you want to read.
3. The read data value is available in the DATA bits in USERACCESS*n* after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in USERACCESS*n*. Once the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

22.2.6.2.4 Example of MDIO Register Access Code

The MDIO module uses the MDIO user access register (USERACCESS*n*) to access the PHY control registers. Software functions that implement the access process may simply be the following four macros:

- PHYREG_read(regadr, phyadr) — Start the process of reading a PHY register
- PHYREG_write(regadr, phyadr, data) — Start the process of writing a PHY register
- PHYREG_wait( ) — Synchronize operation (make sure read/write is idle)
- PHYREG_waitResults(results) — Wait for read to complete and return data read

Note that it is not necessary to wait after a write operation, as long as the status is checked before every operation to make sure the MDIO hardware is idle. An alternative approach is to call PHYREG_wait() after every write, and PHYREG_waitResults( ) after every read, then the hardware can be assumed to be idle when starting a new operation.

The implementation of these macros using the chip support library (CSL) is shown in Example 22-3 (USERACCESS0 is assumed).

Note that this implementation does not check the ACK bit in USERACCESS*n* on PHY register reads (does not follow the procedure outlined in Section 22.2.6.2.3). Since the MDIO PHY alive status register (ALIVE) is used to initially select a PHY, it is assumed that the PHY is acknowledging read operations. It is possible that a PHY could become inactive at a future point in time. An example of this would be a PHY that can have its MDIO addresses changed while the system is running. It is not very likely, but this condition can be tested by periodically checking the PHY state in ALIVE.

Example 22-3. MDIO Register Access Macros

```
#define PHYREG_read(regadr, phyadr)
        MDIO_REGS->USERACCESS0 =
             CSL_FMK(MDIO_USERACCESS0_GO,1u)                    |               /
             CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)            |               /
             CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)

#define PHYREG_write(regadr, phyadr, data)
        MDIO_REGS->USERACCESS0 =
             CSL_FMK(MDIO_USERACCESS0_GO,1u)                    |               /
             CSL_FMK(MDIO_USERACCESS0_WRITE,1)                  |               /
             CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)            |               /
             CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)            |               /
             CSL_FMK(MDIO_USERACCESS0_DATA, data)

#define PHYREG_wait()
        while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) )

#define PHYREG_waitResults( results ) {
        while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) );
        results = CSL_FEXT(MDIO_REGS->USERACCESS0, MDIO_USERACCESS0_DATA); }
```

22.2.7 EMAC Module

This section discusses the architecture and basic function of the EMAC module.

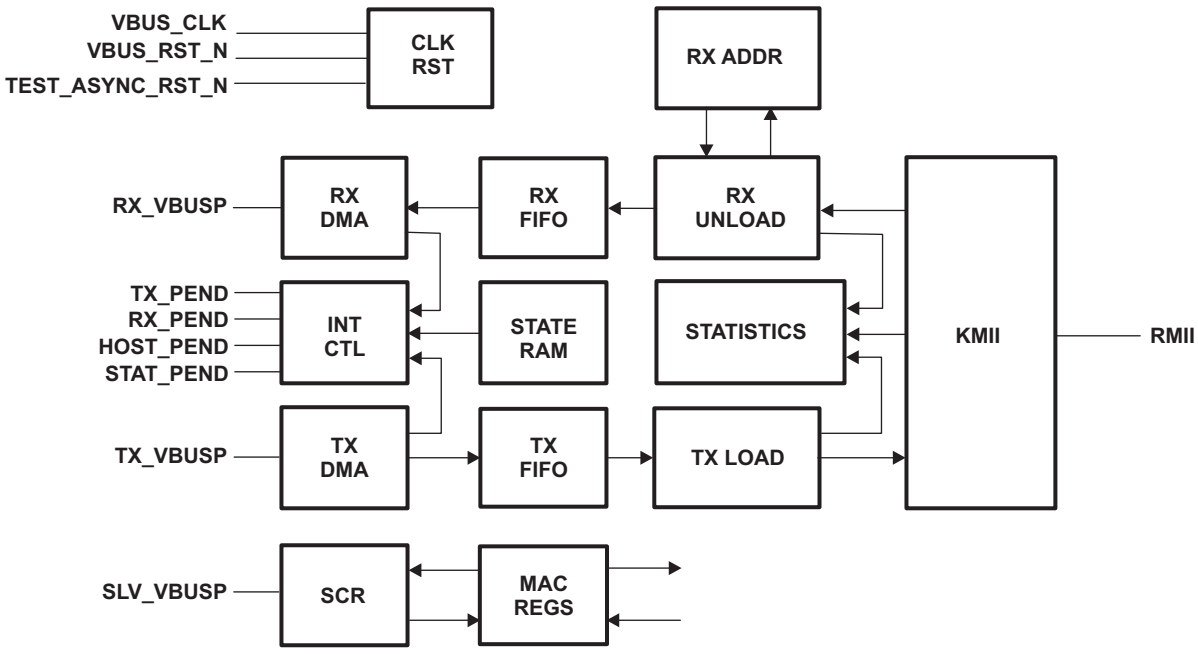
22.2.7.1 EMAC Module Components

The EMAC module (Figure 22-9) interfaces to the outside world Reduced Media Independent Interface (RMII). After reset, initialization, and configuration the host may initiate transmit operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the STATERAM block. The transmit DMA controller then fetches the first packet in the packet chain from memory in accordance with CPPI 3.0 protocol. The DMA controller writes the packet into the transmit FIFO in 64-byte cells. When the threshold number of cells (configurable) have been written to the transmit FIFO, or a complete packet whichever is smaller, the transmit load block then initiates the packet transmission. The KMII block transmits the packet over the RMII in accordance with the 802.3 protocol. Transmit statistics are counted by the statistics submodule.

Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer after host initialization and configuration. The KMII submodule receives packets and strips off the Ethernet related protocol. The packet data is input to the receive unload submodule which checks for address match (in conjunction with the receive address submodule) and processes errors. Accepted packets are then written to the receive FIFO in 64-byte cells. The receive DMA controller then writes the packet data to memory in accordance with CPPI 3.0 protocol. Receive statistics are counted by the statistics block.

The host configures and initializes the device through the five module slave VBUSP interfaces. The CPPI, REGS, MDIO, and EMAC, modules have slave interfaces. The CPPI and EMAC slave interfaces are input directly to a VBUSP to VBUSP retiming bridge before being input to the switched central resource (SCR) inside the respective modules. The retiming bridges help to minimize long paths. The SCR handles decode between multiple master/slaves inside the respective modules. The interrupt control block is used to select the interrupts from the EMAC and the MDIO for output to the three CPU's. The signals required to support gigabit mode are included in the module. However gigabit transfer rates are not supported due to the small 3-cell transmit and receive FIFO's. Also, the RFTCLK input can be tied low for 10/100 operation.

Figure 22-9. EMAC Module Block Diagram



22.2.7.2 EMAC Module Operational Overview

After reset, initialization, and configuration, the host may initiate transmit operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the state RAM block. The transmit DMA controller then fetches the first packet in the packet chain from memory. The DMA controller writes the packet into the transmit FIFO in bursts of 64-byte cells. When the threshold number of cells, configurable using the TXCELLTHRESH bit in the FIFO control register (FIFOCONTROL), have been written to the transmit FIFO, or a complete packet, whichever is smaller, the MAC transmitter then initiates the packet transmission. The SYNC block transmits the packet over the MII or RMII interfaces in accordance with the 802.3 protocol. Transmit statistics are counted by the statistics block.

Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer after host initialization and configuration. The SYNC submodule receives packets and strips off the Ethernet related protocol. The packet data is input to the MAC receiver, which checks for address match and processes errors. Accepted packets are then written to the receive FIFO in bursts of 64-byte cells. The receive DMA controller then writes the packet data to memory. Receive statistics are counted by the statistics block.

The EMAC module operates independently of the CPU. It is configured and controlled by its register set mapped into device memory. Information about data packets is communicated by use of 16-byte descriptors that are placed in an 8K-byte block of RAM in the EMAC subsystem (CPPI buffer descriptor memory).

For transmit operations, each 16-byte descriptor describes a packet or packet fragment in the system's internal or external memory. For receive operations, each 16-byte descriptor represents a free packet buffer or buffer fragment. On both transmit and receive, an Ethernet packet is allowed to span one or more memory fragments, represented by one 16-byte descriptor per fragment. In typical operation, there is only one descriptor per receive buffer, but transmit packets may be fragmented, depending on the software architecture.

An interrupt is issued to the CPU whenever a transmit or receive operation has completed. However, it is not necessary for the CPU to service the interrupt while there are additional resources available. In other words, the EMAC continues to receive Ethernet packets until its receive descriptor list has been exhausted. On transmit operations, the transmit descriptors need only be serviced to recover their associated memory buffer. Thus, it is possible to delay servicing of the EMAC interrupt if there are real-time tasks to perform.

Eight channels are supplied for both transmit and receive operations. On transmit, the eight channels represent eight independent transmit queues. The EMAC can be configured to treat these channels as an equal priority "round-robin" queue or as a set of eight fixed-priority queues. On receive, the eight channels represent eight independent receive queues with packet classification. Packets are classified based on the destination MAC address. Each of the eight channels is assigned its own MAC address, enabling the EMAC module to act like eight virtual MAC adapters. Also, specific types of frames can be sent to specific channels. For example, multicast, broadcast, or other (promiscuous, error, etc.), can each be received on a specific receive channel queue.

The EMAC keeps track of 36 different statistics, plus keeps the status of each individual packet in its corresponding packet descriptor.

22.2.8 MAC Interface

The following sections discuss the operation of the Media Independent Interface (MII) and Reduced Media Independent Interface (RMII) in 10 Mbps and 100 Mbps mode. An IEEE 802.3 compliant Ethernet MAC controls the interface.

22.2.8.1 Data Reception

22.2.8.1.1 Receive Control

Data received from the PHY is interpreted and output to the EMAC receive FIFO. Interpretation involves detection and removal of the preamble and start-of-frame delimiter, extraction of the address and frame length, data handling, error checking and reporting, cyclic redundancy checking (CRC), and statistics control signal generation. Address detection and frame filtering is performed outside the MAC interface.

22.2.8.1.2 Receive Inter-Frame Interval

The 802.3 standard requires an interpacket gap (IPG), which is 96 bit times. However, the EMAC can tolerate a reduced IPG of 8 bit times with a correct preamble and start frame delimiter. This interval between frames must comprise (in the following order):

1. An Interpacket Gap (IPG).
2. A 7-byte preamble (all bytes 55h).
3. A 1-byte start of frame delimiter (5Dh).

22.2.8.1.3 Receive Flow Control

When enabled and triggered, receive flow control is initiated to limit the EMAC from further frame reception. Two forms of receive buffer flow control are available:

- Collision-based flow control for half-duplex mode
- IEEE 802.3x pause frames flow control for full-duplex mode

In either case, receive flow control prevents frame reception by issuing the flow control appropriate for the current mode of operation. Receive flow control prevents reception of frames on the EMAC until all of the triggering conditions clear, at which time frames may again be received by the EMAC.

Receive flow control is enabled by the RXBUFFERFLOWEN bit in the MAC control register (MACCONTROL). The EMAC is configured for collision or IEEE 802.3X flow control using the FULLDUPLEX bit in MACCONTROL. Receive flow control is triggered when the number of free buffers in any enabled receive channel free buffer count register (RX*n*FREEBUFFER) is less than or equal to the receive channel flow control threshold register (RX*n*FLOWTHRESH) value. Receive flow control is independent of receive QOS, except that both use the free buffer values.

22.2.8.1.3.1 Collision-Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in half-duplex mode (the FULLDUPLEX bit is cleared in MACCONTROL). When receive flow control is enabled and triggered, the EMAC generates collisions for received frames. The jam sequence transmitted is the 12-byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3h. The jam sequence begins no later than approximately as the source address starts to be received. Note that these forced collisions are not limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm.

Receive flow control does not depend on the value of the incoming frame destination address. A collision is generated for any incoming packet, regardless of the destination address, if any EMAC enabled channel's free buffer register value is less than or equal to the channel's flow threshold value.

22.2.8.1.3.2 IEEE 802.3x-Based Receive Buffer Flow Control

IEEE 802.3x-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in full-duplex mode (the FULLDUPLEX bit is set in MACCONTROL). When receive flow control is enabled and triggered, the EMAC transmits a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The EMAC transmits a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle or following the completion of the frame currently being transmitted). The pause frame contains the maximum possible value for the pause time (FFFFh). The EMAC counts the receive pause frame time (decrements FF00h to 0) and retransmits an outgoing pause frame, if the count reaches 0. When the flow control request is removed, the EMAC transmits a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval are received normally (provided the receive FIFO is not full).

Pause frames are transmitted if enabled and triggered, regardless of whether or not the EMAC is observing the pause time period from an incoming pause frame.

The EMAC transmits pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01h.
- The 48-bit source address (set using the MACSRCADDRLO and MACSRCADDRHI registers).
- The 16-bit length/type field containing the value 88.08h.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time value of FF.FFh. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request have a pause time value of 00.00h.
- Zero padding to 64-byte data length (EMAC transmits only 64-byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

If the RXBUFFERFLOWEN bit in MACCONTROL is cleared to 0 while the pause time is nonzero, then the pause time is cleared to 0 and a zero count pause frame is sent.

22.2.8.2 Data Transmission

The EMAC passes data to the PHY from the transmit FIFO (when enabled). Data is synchronized to the transmit clock rate. Transmission begins when there are TXCELLTHRESH cells of 64 bytes each, or a complete packet, in the FIFO.

22.2.8.2.1 Transmit Control

A jam sequence is output if a collision is detected on a transmit packet. If the collision was late (after the first 64 bytes have been transmitted), the collision is ignored. If the collision is not late, the controller will back off before retrying the frame transmission. When operating in full-duplex mode, the carrier sense (MII_CRS) and collision-sensing (MII_COL) modes are disabled.

22.2.8.2.2 CRC Insertion

If the SOP buffer descriptor PASSCRC flag is cleared, the EMAC generates and appends a 32-bit Ethernet CRC onto the transmitted data. For the EMAC-generated CRC case, a CRC (or placeholder) at the end of the data is allowed but not required. The buffer byte count value should not include the CRC bytes, if they are present.

If the SOP buffer descriptor PASSCRC flag is set, then the last four bytes of the transmit data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the buffer byte count value. The MAC performs no error checking on the outgoing CRC.

22.2.8.2.3 Adaptive Performance Optimization (APO)

The EMAC incorporates adaptive performance optimization (APO) logic that may be enabled by setting the TXPACE bit in the MAC control register (MACCONTROL). Transmission pacing to enhance performance is enabled when the TXPACE bit is set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions), thereby, increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions, or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision, or excessive collision), the pacing counter is decremented by 1, down to 0.

With pacing enabled, a new frame is permitted to immediately (after one interpacket gap) attempt transmission only if the pacing counter is 0. If the pacing counter is nonzero, the frame is delayed by the pacing delay of approximately four interpacket gap (IPG)delays. APO only affects the IPG preceding the first attempt at transmitting a frame; APO does not affect the back-off algorithm for retransmitted frames.

22.2.8.2.4 Interpacket-Gap (IPG) Enforcement

The measurement reference for the IPG of 96 bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision and MII_CRS is deasserted within approximately 48 bit times of MII_TXEN being deasserted, then 96 bit times is measured from MII_TXEN. If the frame suffered a collision or MII_CRS is not deasserted until more than approximately 48 bit times after MII_TXEN is deasserted, then 96 bit times (approximately, but not less) is measured from MII_CRS.

22.2.8.2.5 Back Off

The EMAC implements the 802.3 binary exponential back-off algorithm.

22.2.8.2.6 Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the EMAC from transmitting any further frames. Incoming pause frames are only acted upon when the FULLDUPLEX and TXFLOWEN bits in the MAC control register (MACCONTROL) are set. Pause frames are not acted upon in half-duplex mode. Pause frame action is taken if enabled, but normally the frame is filtered and not transferred to memory. MAC control frames are transferred to memory, if the RXCMFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is set. The TXFLOWEN and FULLDUPLEX bits affect whether or not MAC control frames are acted upon, but they have no affect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC control frames with an opcode field of 0001h. Incoming pause frames are only acted upon by the EMAC if:

- TXFLOWEN bit is set in MACCONTROL
- The frame's length is 64 to RXMAXLEN bytes inclusive
- The frame contains no CRC error or align/code errors

The pause time value from valid frames is extracted from the two bytes following the opcode. The pause time is loaded into the EMAC transmit pause timer and the transmit pause time period begins. If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- If the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer immediately expires, or
- If the new pause time value is 0, then the transmit pause timer immediately expires, else
- The EMAC transmit pause timer immediately is set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame is discarded).

If the TXFLOWEN bit in MACCONTROL is cleared, then the pause timer immediately expires.

The EMAC does not start the transmission of a new data frame any sooner than 512 bit-times after a pause frame with a nonzero pause time has finished being received (MII_RXDV going inactive). No transmission begins until the pause timer has expired (the EMAC may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received is completed and unaffected.

Incoming pause frames consist of:

- A 48-bit destination address equal to one of the following:
  - The reserved multicast destination address 01.80.C2.00.00.01h
  - Any EMAC 48-bit unicast address. Pause frames are accepted, regardless of whether the channel is enabled or not.
- The 16-bit length/type field containing the value 88.08h.
- The 48-bit source address of the transmitting device.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 bytes. The standard allows pause frames longer than 64 bytes to be discarded or interpreted as valid pause frames. The EMAC recognizes any pause frame between 64 bytes and RXMAXLEN bytes in length.

22.2.8.2.7 Speed, Duplex, and Pause Frame Support

The MAC operates at 10 Mbps or 100 Mbps, in half-duplex or full-duplex mode, and with or without pause frame support as configured by the host.

22.2.9 Packet Receive Operation

22.2.9.1 Receive DMA Host Configuration

To configure the receive DMA for operation the host must:

- Initialize the receive addresses.
- Initialize the receive channel *n* DMA head descriptor pointer registers (RX*n*HDP) to 0.
- Write the MAC address hash *n* registers (MACHASH1 and MACHASH2), if multicast addressing is desired.
- If flow control is to be enabled, initialize:
  - the receive channel *n* free buffer count registers (RX*n*FREEBUFFER)
  - the receive channel *n* flow control threshold register (RX*n*FLOWTHRESH)
  - the receive filter low priority frame threshold register (RXFILTERLOWTHRESH)
- Enable the desired receive interrupts using the receive interrupt mask set register (RXINTMASKSET in EMAC module).
- Enable the receive interrupts of the desired channels in the *Interrupt Core Receive Interrupt Enable Registers* (EMAC subsystem).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
- Setup the receive channel(s) buffer descriptors in host memory as required by CPPI 3.0 and initialize RX*n*HDP.
- Enable the RX DMA controller by setting the RXEN bit in the RXCONTROL register.
- Configure and enable the receive operation, as desired, in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) and by using the receive unicast set register (RXUNICASTSET) and the receive unicast clear register (RXUNICASTCLEAR).

22.2.9.2 Receive Channel Enabling

Each of the eight receive channels has an enable bit (RXCH*n*EN) in the receive unicast set register (RXUNICASTSET) that is controlled using RXUNICASTSET and the receive unicast clear register (RXUNICASTCLEAR). The RXCH*n*EN bits determine whether the given channel is enabled (when set to 1) to receive frames with a matching unicast or multicast destination address.

The RXBROADEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) determines if broadcast frames are enabled or filtered. If broadcast frames are enabled (when set to 1), then they are copied to only a single channel selected by the RXBROADCH bit in RXMBPENABLE.

The RXMULTEN bit in RXMBPENABLE determines if hash matching multicast frames are enabled or filtered. Incoming multicast addresses (group addresses) are hashed into an index in the hash table. If the indexed bit is set then the frame hash matches and will be transferred to the channel selected by the RXMULTCH bit in RXMBPENABLE when multicast frames are enabled. The multicast hash bits are set in the MAC address hash *n* registers (MACHASH1 and MACHASH2).

The RXPROMCH bit in RXMBPENABLE selects the promiscuous channel to receive frames selected by the RXCMFEN, RXCSFEN, RXCEFEN, and RXCAFEN bits. These four bits allow reception of MAC control frames, short frames, error frames, and all frames (promiscuous), respectively.

22.2.9.3 Receive Address Matching

All eight MAC addresses corresponding to the eight receive channels share the upper 40 bits. Only the lower byte is unique for each address. All eight receive addresses should be initialized, because pause frames are acted upon regardless of whether a channel is enabled or not.

A MAC address is written by first writing the address number (channel) to be written into the MAC index register (MACINDEX). The upper 32 bits of address are then written to the MAC address high bytes register (MACADDRHI), which is followed by writing the lower 16 bits of address to the MAC address low bytes register (MACADDRLO). Since all eight MAC addresses share the upper 40 bits of address, MACADDRHI needs to be written only the first time (for the first channel configured).

22.2.9.4 VBUSP Latency

The transmit (and receive) FIFOs each contain a configurable number of 64-byte cells. The EMAC begins transmission of a packet on the wire after tx_cell_thresh cells, or a complete packet, are available in the FIFO. Transmit underrun cannot occur for packet sizes of tx_cell_thresh times 64 bytes (or less) because the entire packet is contained in the FIFO before the packet transmission is started. For larger packet sizes, transmit underrun can occur if the memory latency is greater than the time required to transmit a 64-byte cell on the wire (5.12us in 100 Mbit mode). The latency time includes all buffer descriptor reads for the entire cell data.

Receive overrun will be prevented if the receive memory cell latency is less than the time required to transmit a 64-byte cell on the wire (5.12µs in 100 Mbit mode). The latency time includes any required buffer descriptor reads for the cell data.

22.2.9.5 Hardware Receive QOS Support

Hardware receive quality of service (QOS) is supported, when enabled, by the Tag Protocol Identifier format and the associated Tag Control Information (TCI) format priority field. When the incoming frame length/type value is equal to 81.00h, the EMAC recognizes the frame as an Ethernet Encoded Tag Protocol Type. The two octets immediately following the protocol type contain the 16-bit TCI field. Bits 15-13 of the TCI field contain the received frames priority (0 to 7). The received frame is a low-priority frame, if the priority value is 0 to 3; the received frame is a high-priority frame, if the priority value is 4 to 7. All frames that have a length/type field value not equal to 81.00h are low-priority frames. Received frames that contain priority information are determined by the EMAC as:

- A 48-bit (6 bytes) destination address equal to:
  - The destination station's individual unicast address.
  - The destination station's multicast address (MACHASH1 and MACHASH2).
  - The broadcast address of all ones.
- A 48-byte (6 bytes) source address.
- The 16-bit (2 bytes) length/type field containing the value 81.00h.
- The 16-bit (2 bytes) TCI field with the priority field in the upper 3 bits.
- Data bytes
- The 4 bytes CRC.

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) and the receive channel *n* free buffer count registers (RX*n*FREEBUFFER) are used in conjunction with the priority information to implement receive hardware QOS. Low-priority frames are filtered if the number of free buffers (RX*n*FREEBUFFER) for the frame channel is less than or equal to the filter low threshold (RXFILTERLOWTHRESH) value. Hardware QOS is enabled by the RXQOSEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

22.2.9.6 Host Free Buffer Tracking

The host must track free buffers for each enabled channel (including unicast, multicast, broadcast, and promiscuous), if receive QOS or receive flow control is used. Disabled channel free buffer values are do not cares. During initialization, the host should write the number of free buffers for each enabled channel to the appropriate receive channel *n* free buffer count registers (RX*n*FREEBUFFER). The EMAC decrements the appropriate channel's free buffer value for each buffer used. When the host reclaims the frame buffers, the host should write the channel free buffer register with the number of reclaimed buffers (write to increment). There are a maximum of 65,535 free buffers available. RX*n*FREEBUFFER only needs to be updated by the host if receive QOS or flow control is used.

22.2.9.7 Receive Channel Teardown

The host commands a receive channel teardown by writing the channel number to the receive teardown register (RXTEARDOWN). When a teardown command is issued to an enabled receive channel, the following occurs:

- Any current frame in reception completes normally.
- The TDOWNCMPLT flag is set in the next buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A receive interrupt for the channel is issued to the host.
- The corresponding receive channel *n* completion pointer register (RX*n*CP) contains the value FFFF FFFCh.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to RX*n*CP (note that there is no buffer descriptor in this case). Software may read RX*n*CP to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

22.2.9.8 Receive Frame Classification

Received frames are proper (good) frames, if they are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length (inclusive) and contain no code, align, or CRC errors.

Received frames are long frames, if their frame count exceeds the value in RXMAXLEN. The RXMAXLEN reset (default) value is 5EEh (1518 in decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames; long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames, if their frame count is less than 64 bytes. Short frames that address match and contain no errors are undersized frames; short frames with CRC, code, or alignment errors are fragment frames. If the frame length is less than or equal to 20, then the frame CRC is passed, regardless of whether the RXPASSCRC bit is set or cleared in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

A received long packet always contains RXMAXLEN number of bytes transferred to memory (if the RXCEFEN bit is set in RXMBPENABLE), regardless of the value of the RXPASSCRC bit. Following is an example with RXMAXLEN set to 1518:

- If the frame length is 1518, then the packet is not a long packet and there are 1514 or 1518 bytes transferred to memory depending on the value of the RXPASSCRC bit.
- If the frame length is 1519, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last three bytes are the first three CRC bytes.
- If the frame length is 1520, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last two bytes are the first two CRC bytes.
- If the frame length is 1521, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last byte is the first CRC byte.
- If the frame length is 1522, there are 1518 bytes transferred to memory. The last byte is the last data byte.

22.2.9.9 Promiscuous Receive Mode

When the promiscuous receive mode is enabled by setting the RXCAFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE), nonaddress matching frames that would normally be filtered are transferred to the promiscuous channel. Address matching frames that would normally be filtered due to errors are transferred to the address match channel when the RXCAFEN and RXCEFEN bits in RXMBPENABLE are set. A frame is considered to be an address matching frame only if it is enabled to be received on a unicast, multicast, or broadcast channel. Frames received to disabled unicast, multicast, or broadcast channels are considered nonaddress matching.

MAC control frames address match only if the RXCMFEN bit in RXMBPENABLE is set. The RXCEFEN and RXCSFEN bits in RXMBPENABLE determine whether error frames are transferred to memory or not, but they do not determine whether error frames are address matching or not. Short frames are a special type of error frames.

A single channel is selected as the promiscuous channel by the RXPROMCH bit in RXMBPENABLE. The promiscuous receive mode is enabled by the RXCMFEN, RXCEFEN, RXCSFEN, and RXCAFEN bits in RXMBPENABLE. Table 22-12 shows the effects of the promiscuous enable bits. Proper frames are frames that are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length inclusive and contain no code, align, or CRC errors.

Table 22-12. Receive Frame Treatment Summary

Address Match	RXCAFEN	RXCEFEN	RXCMFEN	RXCSFEN	Receive Frame Treatment
0	0	X	X	X	No frames transferred.
0	1	0	0	0	Proper frames transferred to promiscuous channel.
0	1	0	0	1	Proper/undersized data frames transferred to promiscuous channel.
0	1	0	1	0	Proper data and control frames transferred to promiscuous channel.
0	1	0	1	1	Proper/undersized data and control frames transferred to promiscuous channel.
0	1	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to promiscuous channel. No control or undersized/fragment frames are transferred.
0	1	1	0	1	Proper/undersized/fragment/oversize/jabber/code/ align/CRC data frames transferred to promiscuous channel. No control frames are transferred.
0	1	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to promiscuous channel. No undersized frames are transferred.
0	1	1	1	1	All nonaddress matching frames with and without errors transferred to promiscuous channel.
1	X	0	0	0	Proper data frames transferred to address match channel.
1	X	0	0	1	Proper/undersized data frames transferred to address match channel.
1	X	0	1	0	Proper data and control frames transferred to address match channel.
1	X	0	1	1	Proper/undersized data and control frames transferred to address match channel.
1	X	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to address match channel. No control or undersized frames are transferred.
1	X	1	0	1	Proper/oversize/jabber/fragment/undersized/code/ align/CRC data frames transferred to address match channel. No control frames are transferred.
1	X	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to address match channel. No undersized/fragment frames are transferred.

Table 22-12. Receive Frame Treatment Summary (continued)

Address Match	RXCAFEN	RXCEFEN	RXCMFEN	RXCSFEN	Receive Frame Treatment
1	X	1	1	1	All address matching frames with and without errors transferred to the address match channel

22.2.9.10 Big Endian Mode

When the BIG_ENDIAN input is asserted, the EMAC assumes that packet data is contained in memory in big endian format. When the BIG_ENDIAN input is deasserted, the EMAC assumes that packet data is contained in memory in little endian format. The BIG_ENDIAN input causes big endian packet data to go out on the wire in the same order that little endian packet data goes out on the wire when the input is not asserted (byte 0 first). The BIG_ENDIAN input has no effect on buffer descriptor data reads or writes because buffer descriptor data is a 32-bit quantity (unlike packet data which is an 8- bit quantity).

Table 22-13. Little Endian

High Add			Low Add
Byte 3	Byte 2	Byte 1	Byte 0
Byte 7	Byte 6	Byte 5	Byte 4
Byte 11	Byte 10	Byte 9	Byte 8

Table 22-14. Big Endian

High Add			Low Add
Byte 0	Byte 1	Byte 2	Byte 3
Byte 4	Byte 5	Byte 6	Byte 7
Byte 8	Byte 9	Byte 10	Byte 11

22.2.9.11 Receive Overrun

The types of receive overrun are:

- FIFO start of frame overrun (FIFO_SOF)
- FIFO middle of frame overrun (FIFO_MOF)
- DMA start of frame overrun (DMA_SOF)
- DMA middle of frame overrun (DMA_MOF)

The statistics counters used to track these types of receive overrun are:

- Receive start of frame overruns register (RXSOFOVERRUNS)
- Receive middle of frame overruns register (RXMOFOVERRUNS)
- Receive DMA overruns register (RXDMAOVERRUNS)

Start of frame overruns happen when there are no resources available when frame reception begins. Start of frame overruns increment the appropriate overrun statistic(s) and the frame is filtered.

Middle of frame overruns happen when there are some resources to start the frame reception, but the resources run out during frame reception. In normal operation, a frame that overruns after starting the frame reception is filtered and the appropriate statistic(s) are incremented; however, the RXCEFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) affects overrun frame treatment. Table 22-15 shows how the overrun condition is handled for the middle of frame overrun.

Table 22-15. Middle of Frame Overrun Treatment

Address Match	RXCAFEN	RXCEFEN	Middle of Frame Overrun Treatment
0	0	X	Overrun frame filtered.
0	1	0	Overrun frame filtered.

Table 22-15. Middle of Frame Overrun Treatment (continued)

Address Match	RXCAFEN	RXCEFEN	Middle of Frame Overrun Treatment
0	1	1	As much frame data as possible is transferred to the promiscuous channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN and NOMATCH flags are set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated and be a jabber or oversize).
1	X	0	Overrun frame filtered with the appropriate overrun statistic(s) incremented.
1	X	1	As much frame data as possible is transferred to the address match channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN flag is set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated).

22.2.10 Packet Transmit Operation

The transmit DMA is an eight channel interface. Priority between the eight queues may be either fixed or round-robin as selected by the TXPTYPE bit in the MAC control register (MACCONTROL). If the priority type is fixed, then channel 7 has the highest priority and channel 0 has the lowest priority. Round-robin priority proceeds from channel 0 to channel 7.

22.2.10.1 Transmit DMA Host Configuration

To configure the transmit DMA for operation the host must perform:

- Write the MAC source address low bytes register (MACSRCADDRLO) and the MAC source address high bytes register (MACSRCADDRHI) (used for pause frames on transmit).
- Initialize the transmit channel *n* DMA head descriptor pointer registers (TX*n*HDP) to 0.
- Enable the desired transmit interrupts using the transmit interrupt mask set register (TXINTMASKSET in EMAC module).
- Enable the transmit interrupts of the desired channels in the *Interrupt Core Transmit Interrupt Enable Registers* (EMAC subsystem).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Setup the transmit channel(s) buffer descriptors in host memory.
- Enable the transmit DMA controller by setting the TXEN bit in the transmit control register (TXCONTROL).
- Write the appropriate TX*n*HDP with the pointer to the first descriptor to start transmit operations.

22.2.10.2 Transmit Channel Teardown

The host commands a transmit channel teardown by writing the channel number to the transmit teardown register (TXTEARDOWN). When a teardown command is issued to an enabled transmit channel, the following occurs:

- Any frame currently in transmission completes normally.
- The TDOWNCMPLT flag is set in the next SOP buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A transmit interrupt is issued to inform the host of the channel teardown.
- The corresponding transmit channel *n* completion pointer register (TX*n*CP) contains the value FFFF FFFCh.
- The host should acknowledge a teardown interrupt with an FFFF FFFCh acknowledge value.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to TX*n*CP (note that there is no buffer descriptor in this case). Software may read the interrupt acknowledge location (TX*n*CP) to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

22.2.11 Receive and Transmit Latency

The transmit and receive FIFOs each contain three 64-byte cells. The EMAC begins transmission of a packet on the wire after TXCELLTHRESH (configurable through the FIFO control register) cells, or a complete packet, are available in the FIFO.

Transmit underrun cannot occur for packet sizes of TXCELLTHRESH times 64 bytes (or less). For larger packet sizes, transmit underrun occurs if the memory latency is greater than the time required to transmit a 64-byte cell on the wire; this is 5.12 µs in 100 Mbps mode and 51.2 µs in 10 Mbps mode. The memory latency time includes all buffer descriptor reads for the entire cell data.

Receive overrun is prevented if the receive memory cell latency is less than the time required to transmit a 64-byte cell on the wire: 5.12 µs in 100 Mbps mode, or 51.2 µs in 10 Mbps mode. The latency time includes any required buffer descriptor reads for the cell data.

Latency to system's internal and external RAM can be controlled through the use of the transfer node priority allocation register available at the device level. Latency to descriptor RAM is low because RAM is local to the EMAC, as it is part of the EMAC subsystem.

22.2.12 Transfer Node Priority

The device contains a chip-level master priority register that is used to set the priority of the transfer node used in issuing memory transfer requests to system memory.

Although the EMAC has internal FIFOs to help alleviate memory transfer arbitration problems, the average transfer rate of data read and written by the EMAC to internal or external processor memory must be at least that of the Ethernet wire rate. In addition, the internal FIFO system can not withstand a single memory latency event greater than the time it takes to fill or empty a TXCELLTHRESH number of internal 64 byte FIFO cells.

For 100 Mbps operation, these restrictions translate into the following rules:

- The short-term average, each 64-byte memory read/write request from the EMAC must be serviced in no more than 5.12 µs.
- Any single latency event in request servicing can be no longer than (5.12 × TXCELLTHRESH) µs.

22.2.13 Clock Stop

The clock stop interface allows the EMAC to be gracefully commanded to enter into a stopped clock state. When the (CLKSTOP_REQ) input is asserted, the EMAC suspend state is entered and the EMAC will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the TX cell FIFO will be transmitted. For receive, frames that are detected by the EMAC after the suspend state is entered are ignored. No statistics will be kept for ignored frames. The (CLKSTOP_ACK) output will be asserted when the EMAC_F enters the IDLE state indicating that the clock (VBUSP_CLK) may be stopped. (For details on programming the clock stop, see the *System Control Module* chapter.)

22.2.14 Software Reset

The EMAC subsystem software reset register (REGS) and the EMAC submodule software reset register enable the EMAC subsystem to be reset by software. For the EMAC, the reset state is entered at packet boundaries, at which time the EMAC reset occurs. EMAC reset status is determined by reading or polling the EMAC soft reset bit. If the EMAC reset bit is read as a one, then the reset process has not yet completed. The EMAC soft reset process could take up to 2ms. The reset has completed if the EMAC reset bit is read as a zero. After the EMAC soft reset is complete, the EMAC subsystem software reset register bits may be written to complete the software reset operation.

22.2.14.1 Soft Reset of EMAC Submodule

The EMAC software reset register enables the EMAC to be reset by software. Soft reset will occur at frame boundaries. Soft reset occurs when both receive and transmit DMA controllers are not using their respective VBUS interfaces so that there is no VBUS lockup. The soft reset process could take up to 2ms. The reset has completed if the soft reset bit is read as a zero.

22.2.15 Initialization

22.2.15.1 Enabling the EMAC/MDIO Peripheral

When the device is powered on, the EMAC peripheral may be in a disabled state. Before any EMAC specific initialization can take place, the EMAC needs to be enabled; otherwise, its registers cannot be written and the reads will all return a value of zero.

When first enabled, the EMAC peripheral registers are set to their default values. After enabling the peripheral, you may proceed with the module specific initialization.

22.2.15.2 EMAC Subsystem Module Initialization

The EMAC subsystem module is used for global interrupt enables and to pace interrupts using 1ms time windows. There is also an 8K block of CPPI RAM local to the EMAC that is used to hold packet buffer descriptors.

The initialization of the EMAC subsystem module consists of three parts:

1. Configuration of the interrupt to the CPU.
2. Initialization of the EMAC subsystem module:
   - Setting the interrupt pace counts using the EMAC control module registers INTCONTROL, C*n*RXIMAX, and C*n*TXIMAX
   - Initializing the EMAC and MDIO modules
   - Enabling interrupts in the EMAC subsystem module using the EMAC control module interrupt control registers C*n*RXTHRESHEN, C*n*RXEN, C*n*TXEN, and CnMISCEN.

The process of mapping the EMAC interrupts to the CPU is done through the CPU interrupt controller. Once the interrupt is mapped to a CPU interrupt, general masking and unmasking of interrupts (to control reentrancy) should be done at the chip level by manipulating the interrupt core enable mask registers.

22.2.15.3 MDIO Module Initialization

The MDIO module is used to initially configure and monitor one or more external PHY devices. Other than initializing the software state machine (details on this state machine can be found in the IEEE 802.3 standard), all that needs to be done for the MDIO module is to enable the MDIO engine and to configure the clock divider. To set the clock divider, supply an MDIO clock of 1 MHZ. For example, if the peripheral clock is 50 MHz, the divider can be set to 50.

Both the state machine enable and the MDIO clock divider are controlled through the MDIO control register (CONTROL). If none of the potentially connected PHYs require the access preamble, the PREAMBLE bit in CONTROL can also be set to speed up PHY register access.

If the MDIO module is to operate on an interrupt basis, the interrupts can be enabled at this time using the MDIO user command complete interrupt mask set register (USERINTMASKSET) for register access and the MDIO user PHY select register (USERPHYSEL*n*) if a target PHY is already known.

Once the MDIO state machine has been initialized and enabled, it starts polling all 32 PHY addresses on the MDIO bus, looking for an active PHY. Since it can take up to 50 µs to read one register, it can be some time before the MDIO module provides an accurate representation of whether a PHY is available. Also, a PHY can take up to 3 seconds to negotiate a link. Thus, it is advisable to run the MDIO software off a time-based event rather than polling.

For more information on PHY control registers, see your PHY device documentation.

22.2.15.4 EMAC Module Initialization

The EMAC module is used to send and receive data packets over the network. This is done by maintaining up to eight transmit and receive descriptor queues. The EMAC module configuration must also be kept up-to-date based on PHY negotiation results returned from the MDIO module. Most of the work in developing an application or device driver for Ethernet is programming this module.

The following is the initialization procedure a device driver would follow to get the EMAC to the state where it is ready to receive and send Ethernet packets. Some of these steps are not necessary when performed immediately after device reset.

1. If enabled, clear the device interrupt enable bits in the EMAC subsystem module interrupt control registers C*n*RXTHRESHEN, C*n*RXEN, C*n*TXEN, and C*n*MISCEN.
2. Clear the MAC control register (MACCONTROL), receive control register (RXCONTROL), and transmit control register (TXCONTROL) (not necessary immediately after reset).
3. Initialize all 16 header descriptor pointer registers (RX*n*HDP and TX*n*HDP) to 0.
4. Clear all 36 statistics registers by writing 0 (not necessary immediately after reset).
5. Setup the local Ethernet MAC address by programming the MAC index register (MACINDEX), MAC address high bytes register (MACADDRHI), and MAC address low bytes register (MACADDRLO). Be sure to program all eight MAC address registers - whether the receive channel is to be enabled or not. Duplicate the same MAC address across all unused channels. When using more than one receive channel, start with channel 0 and progress upwards.
6. If buffer flow control is to be enabled, initialize the receive channel *n* free buffer count registers (RX*n*FREEBUFFER), receive channel *n* flow control threshold register (RX*n*FLOWTHRESH), and receive filter low priority frame threshold register (RXFILTERLOWTHRESH).
7. Most device drivers open with no multicast addresses, so clear the MAC address hash registers (MACHASH1 and MACHASH2) to 0.
8. Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
9. Initially clear all unicast channels by writing FFh to the receive unicast clear register (RXUNICASTCLEAR). If unicast is desired, it can be enabled now by writing the receive unicast set register (RXUNICASTSET). Some drivers will default to unicast on device open while others will not.
10. Setup the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) with an initial configuration. The configuration is based on the current receive filter settings of the device driver. Some drivers may enable things like broadcast and multicast packets immediately, while others may not.
11. Set the appropriate configuration bits in MACCONTROL (do not set the GMIIEN bit yet).
12. Clear all unused channel interrupt bits by writing the receive interrupt mask clear register (RXINTMASKCLEAR) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
13. Enable the receive and transmit channel interrupt bits in the receive interrupt mask set register (RXINTMASKSET) and the transmit interrupt mask set register (TXINTMASKSET) for the channels to be used, and enable the HOSTMASK and STATMASK bits using the MAC interrupt mask set register (MACINTMASKSET).
14. Initialize the receive and transmit descriptor list queues.
15. Prepare receive by writing a pointer to the head of the receive buffer descriptor list to RX*n*HDP.
16. Enable the receive and transmit DMA controllers by setting the RXEN bit in RXCONTROL and the TXEN bit in TXCONTROL. Then set the GMIIEN bit in MACCONTROL.
17. Enable the device interrupt in EMAC subsystem module registers C*n*RXTHRESHEN, C*n*RXEN, C*n*TXEN, and C*n*MISCEN.

22.2.16 Interrupt Support

22.2.16.1 EMAC Module Interrupt Events and Requests

The EMAC module generates 26 interrupt events:

- TXPEND*n*: Transmit packet completion interrupt for transmit channels 0 through 7
- RXPEND*n*: Receive packet completion interrupt for receive channels 0 through 7
- RXTHRESHPEND*n*: Receive packet completion interrupt for receive channels 0 through 7 when flow control is enabled and the number of free buffers is below the threshold
- STATPEND: Statistics interrupt
- HOSTPEND: Host error interrupt

22.2.16.1.1 Transmit Packet Completion Interrupts

The transmit DMA engine has eight channels, with each channel having a corresponding interrupt (TXPEND*n*). The transmit interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight transmit channel interrupts may be individually enabled by setting the appropriate bit in the transmit interrupt mask set register (TXINTMASKSET) to 1. Each of the eight transmit channel interrupts may be individually disabled by clearing the appropriate bit by writing a 1 to the transmit interrupt mask clear register (TXINTMASKCLEAR). The raw and masked transmit interrupt status may be read by reading the transmit interrupt status (unmasked) register (TXINTSTATRAW) and the transmit interrupt status (masked) register (TXINTSTATMASKED), respectively.

When the EMAC completes the transmission of a packet, the EMAC issues an interrupt to the CPU (via the EMAC subsystem) when it writes the packet's last buffer descriptor address to the appropriate channel queue's transmit completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges an interrupt by writing the address of the last buffer descriptor processed to the queue's associated transmit completion pointer in the transmit DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC port (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has transmitted more packets than the CPU has processed interrupts for), the transmit packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has transferred), the pending interrupt is cleared. The value that the EMAC is expecting is found by reading the transmit channel *n* completion pointer register (TX*n*CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

The application software must acknowledge the EMAC subsystem module after processing packets by writing the appropriate C*n*RX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.16.1.2 Receive Packet Completion Interrupts

The receive DMA engine has eight channels, which each channel having a corresponding interrupt (RXPEND*n*). The receive interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight receive channel interrupts may be individually enabled by setting the appropriate bit in the receive interrupt mask set register (RXINTMASKSET) to 1. Each of the eight receive channel interrupts may be individually disabled by clearing the appropriate bit by writing a 1 in the receive interrupt mask clear register (RXINTMASKCLEAR). The raw and masked receive interrupt status may be read by reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively.

When the EMAC completes a packet reception, the EMAC issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's receive completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges one or more interrupt(s) by writing the address of the last buffer descriptor processed to the queue's associated receive completion pointer in the receive DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has received more packets than the CPU has processed interrupts for), the receive packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has received), the pending interrupt is de-asserted. The value that the EMAC is expecting is found by reading the receive channel *n* completion pointer register (RX*n*CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

The application software must acknowledge the EMAC subsystem module after processing packets by writing the appropriate C*n*TX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.16.1.3 Statistics Interrupt

The statistics level interrupt (STATPEND) is issued when any statistics value is greater than or equal to 8000 0000h, if enabled by setting the STATMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. As long as the most-significant bit of any statistics value is set, the interrupt remains asserted.

The application software must acknowledge the EMAC subsystem module after receiving statistics interrupts by writing the appropriate C*n*MISC key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.16.1.4 Host Error Interrupt

The host error interrupt (HOSTPEND) is issued, if enabled, under error conditions dealing with the handling of buffer descriptors, detected during transmit or receive DMA transactions. The failure of the software application to supply properly formatted buffer descriptors results in this error. The error bit can only be cleared by resetting the EMAC module in hardware.

The host error interrupt is enabled by setting the HOSTMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The host error interrupt is disabled by clearing the appropriate bit by writing a 1 in the MAC interrupt mask clear register (MACINTMASKCLEAR). The raw and masked host error interrupt status may be read by reading the MAC interrupt status (unmasked) register (MACINTSTATRAW) and the MAC interrupt status (masked) register (MACINTSTATMASKED), respectively.

The transmit host error conditions are:

- SOP error
- Ownership bit not set in SOP buffer
- Zero next buffer descriptor pointer with EOP
- Zero buffer pointer
- Zero buffer length
- Packet length error

The receive host error conditions are:

- Ownership bit not set in input buffer
- Zero buffer pointer

The application software must acknowledge the EMAC subsystem module after receiving host error interrupts by writing the appropriate C*n*MISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.16.1.5 Receive Threshold Interrupts

Each of the eight receive channels have a corresponding receive threshold interrupt (RX*n*THRESHPEND). The receive threshold interrupts are level interrupts that remain asserted until the triggering condition is cleared by the host. Each of the eight threshold interrupts may be individually enabled by setting to 1 the appropriate bit in the RXINTMASKSET register. Each of the eight channel interrupts may be individually disabled by clearing to zero the appropriate bit by writing a 1 in the receive interrupt mask clear register (RXINTMASKCLEAR). The raw and masked interrupt receive interrupt status may be read by reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED),respectively.

An RX*n*THRESHPEND interrupt bit is asserted when enabled and when the channel's associated free buffer count (RX*n*FREEBUFFER) is less than or equal to the channel's associated flow control threshold register (RX*n*FLOWTHRESH). The receive threshold interrupts use the same free buffer count and threshold logic as does flow control, but the interrupts are independently enabled from flow control. The threshold interrupts are intended to give the host an indication that resources are running low for a particular channel(s).

The applications software must acknowledge the EMAC subsystem module after receiving threshold interrupts by writing the appropriate C*n*RXTHRESH key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.16.1.6 Pulse Interrupts

EMAC generates both TX_PULSE[7:0] and RX_PULSE[7:0] pulse interrupts.

- **TX_PULSE[7:0]** - Transmit packet completion interrupts for transmit channels 7 to 0.
- **RX_PULSE[7:0]** - Receive packet completion interrupts for receive channels 7 to 0.

22.2.16.1.6.1 Pulse Interrupt Description

The following sections describe the pulse interrupts.

22.2.16.1.6.1.1 Transmit Packet Completion Interrupts

The transmit DMA controller has eight channels with each channel having a corresponding pulse interrupt (TX_PULSE[7:0]). Each of the eight channel pulse interrupts may be individually enabled by setting to one the appropriate bit in the Tx_IntMask_Set register. Each of the eight channel interrupts may be individually disabled by clearing to zero the appropriate bit in the Tx_IntMask_Clear register. When each CPPI packet DMA transmission is complete (not the complete packet transmission on the wire), the EMAC issues a single VBUSP_CLK wide interrupt pulse. No acknowledgment from the host is required.

22.2.16.1.6.1.2 Receive Packet Completion Interrupts

The receive DMA controller has eight channels with each channel having a corresponding pulse interrupt (RX_PULSE[7:0]). Each of the eight channel interrupts may be individually enabled by setting to one the appropriate bit in the Rx_IntMask_Set register. Each of the eight channel interrupts may be individually disabled by clearing to zero the appropriate bit in the Rx_IntMask_Clear register. When a packet DMA reception is complete, the EMAC issues a single VBUSP_CLK wide pulse interrupt. No acknowledgment from the host is required.

22.2.16.2 Proper Interrupt Processing

All the interrupts signaled from the EMAC and MDIO modules are level driven, so if they remain active, their level remains constant; the CPU core may require edge- or pulse-triggered interrupts. In order to properly convert the level-driven interrupt signal to an edge- or pulse-triggered signal, the application software must make use of the interrupt control logic contained in the EMAC subsystem module.

For safe interrupt processing, upon entry to the ISR, the software application should disable interrupts using the EMAC subsystem module registers C*n*RXTHRESHEN, C*n*RXEN, CnTXEN, CnMISCEN, and then reenable them upon leaving the ISR. If any interrupt signals are active at that time, this creates another rising edge on the interrupt signal going to the CPU interrupt controller, thus triggering another interrupt. The EMAC subsystem module also uses the EMAC subsystem module registers INTCONTROL, C*n*TXIMAX, and C*n*RXIMAX to implement interrupt pacing. The application software must acknowledge the EMAC subsystem module by writing the appropriate key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

EMAC modules do not support level interrupt outputs. External logic has been implemented to convert pulse interrupts into level interrupt. Once the MPU writes corresponding bits in this register, this logic registers the pulse interrupt generated by IP and clears the interrupt.

22.2.16.3 Interrupt Multiplexing

The EMAC subsystem module combines different interrupt signals from both the EMAC and MDIO modules into four interrupt signals (C*n*RXTHRESHPULSE, C*n*RXPULSE, C*n*TXPULSE, C*n*MISCPULSE) that are routed to three independent interrupt cores in the subsystem module. Each interrupt core is capable of relaying all four interrupt signals out of the subsystem module. Some devices may have an individual interrupt core dedicated to a specific CPU or interrupt controller. This configuration gives users of devices greater flexibility when allocating system resources for EMAC management.

When an interrupt is generated, the reason for the interrupt can be read from the MAC input vector register (MACINVECTOR) located in the EMAC memory map. MACINVECTOR combines the status of the following 28 interrupt signals: TXPEND*n*, RXPEND*n*, RXTHRESHPEND*n*, STATPEND, HOSTPEND, LINKINT0, and USERINT0.

22.2.16.4 Pulse Interrupts in EMAC SubSystem

The following subsections describe the pulse interrupts that are generated by the EMAC subsystem.

22.2.16.4.1 C(0/1/2) RXTHRESHPULSE Interrupt Description

The C(0/1/2) RXTHRESHPULSEinterrupts are each an immediate (non-paced) pulse interrupt selected from the EMAC RXTHRESHPEND[7:0] interrupts. The receive threshold pending interrupt(s) is selected by setting one or more bits in the C(0/1/2) RxThreshEn[7:0] register. The masked interrupt status can be read in the C(0/1/2) RxThreshStat[7:0] address location. Upon reception of an interrupt, software should perform the following:

- Read the C(0/1/2) RxThreshStat[7:0] address location to determine which channel(s) caused the interrupt.
- Process received packets in order to add more buffers to any channel that is below the threshold value.
- Write the EMAC completion pointer(s).
- Write the appropriate value (0x0, 0x4, or 0x8) to the to the MACEOIVector register in the emac slave address space.
- Write 1 to the bit 2 of LVL_INTR_CLEAR register (0x4800 2594) to clear the interrupt.

22.2.16.4.2 2 C(0/1/2) RXPULSE Interrupt Description

The C(0/1/2) RXPULSE interrupts are each a paced pulse interrupt selected from the EMAC RXPEND[7:0] interrupts. The receive pending interrupt(s) is selected by setting one or more bits in the C(0/1/2) RxEn[7:0] register. The masked interrupt status can be read in the C(0/1/2) RxStat[7:0] address location. Upon reception of an interrupt, software should perform the following:

- Read the C(0/1/2) RxStat[7:0] address location to determine which channel(s) caused the interrupt.
- Process received packets for the interrupting channel(s).
- Write the EMAC completion pointer(s).
- Write the appropriate value (0x1, 0x5, or 0x9) to the MACEOIVector register in the EMAC slave address space.
- Write 1 to the bit 1 in LVL_INTR_CLEAR register (0x4800 2594) to clear the interrupt.

22.2.16.4.3 C(0/1/2) TXPULSE Interrupt Description

The C(0/1/2) TXPULSE interrupts are each a paced pulse interrupt selected from the EMAC TXPEND[7:0] interrupts. The transmit pending interrupt(s) is selected by setting one or more bits in the C(0/1/2) TxEn[7:0] register. The masked interrupt status can be read in the C(0/1/2) TxStat[7:0] address location. Upon reception of an interrupt, software should perform the following:

- Read the C(0/1/2) TxStat[7:0] address location to determine which channel(s) caused the interrupt.
- Process received packets for the interrupting channel(s).
- Write the EMAC completion pointer(s).
- Write the appropriate value (0x2, 0x6, or 0xa) to the MACEOIVector register in the EMAC slave address space.
- Write 1 to the bit 3 in LVL_INTR_CLEAR register (0x4800 2594) to clear the interrupt.

22.2.16.4.4 C(0/1/2) MISCPULSE Interrupt Description

The C(0/1/2) MISCPULSE interrupts are each an immediate (non-paced) pulse interrupt selected from the miscellaneous interrupts (STATPEND, HOSTPEND, MDIOLINKINT[0], MDIOUSERINT[0]). The miscellaneous interrupt(s) is selected by setting one or more bits in the C(0/1/2) MiscEn[3:0] register. The masked interrupt status can be read in the C(0/1/2) MiscStat[3:0] address location. Upon reception of an interrupt, software should perform the following:

- Read the C(0/1/2) MiscStat[3:0] address location to determine which channel(s) caused the interrupt.
- Process the interrupt.
- Write the appropriate value (0x3, 0x7, or 0xb) to the MACEOIVector register in the emac slave address space.
- Write 1 to the bit 0 in LVL_INTR_CLEAR register(0x48002594) to clear the interrupt.

NOTE: MDIOLINKINT[1] and MDIOUSERINT[1] are not connected.

The following 4 pulse interrupts are only connected from the EMAC module to ARM CPU.

- C0MISCPULSE – Core 0 Miscellaneous Interrupt
- C0RXPULSE – Core 0 Receive Interrupt
- C0RXTHRESHPULSE – Core 0 Receive Threshold Interrupt
- C0TXPULSE – Core 0 Transmit Interrupt.

C1 and C2 interrupts are not connected from EMAC Module to ARM CPU.

22.2.16.5 Interrupt Pacing

The receive and transmit pulse interrupts can be paced. The receive threshold and miscellaneous interrupts are not paced. The Interrupt pacing feature limits the number of interrupts that occur during a given period of time. For heavily loaded systems in which interrupts can occur at a very high rate (e.g. 148,800 packets per second for Ethernet), the performance benefit is significant due to minimizing the overhead associated with servicing each interrupt. Interrupt pacing increases the CPU cache hit ratio by minimizing the number of times that large interrupt service routines are moved to and from the CPU instruction cache. Each CPU receive and transmit pulse interrupt contains an interrupt pacing sub-block

(six total). Each sub-block is disabled by default allowing the selected interrupt inputs to pass through unaffected. The interrupt pacing module counts the number of interrupts that occur over a 1ms interval of time. At the end of each 1ms interval, the current number of interrupts is compared with a target number of interrupts (specified by the associated maximum number of interrupts register). Based on the results of the comparison, the length of time during which interrupts are blocked is dynamically adjusted.

The 1ms interval is derived from a 4µs pulse that is created from a prescale counter whose value is set in the INTPRESCALE value in the Interrupt Control register of EMAC subsystem. The INTPRESCALE value should be written with the number of VBUSP_CLK periods in 4µs. The pacing timer determines the interval during which interrupts are blocked and decrements every 4µs. It is reloaded each time a zero count is reached.

The value loaded into the pacing timer is calculated by hardware every 1ms according to the following algorithm:

```
if (intr_count > 2*intr_max)
               pace_timer = 255;

    else if (intr_count > 1.5*intr_max)
               pace_timer = last_pace_timer*2 + 1;

    else if (intr_count > 1.0*intr_max)
               pace_timer = last_pace_timer + 1;

    else if (intr_count > 0.5*intr_max)
               pace_timer = last_pace_timer - 1;

    else if (intr_count != 0)
               pace_timer = last_pace_timer/2;

    else
               pace_timer = 0;
```

If the rate of interrupt inputs is much less than the target interrupt rate specified in the associated maximum interrupts register, then the interrupt is not blocked. If the interrupt rate is greater than the target rate, the interrupt will be "paced' at the rate specified in the interrupt maximum register. The interrupt maximum register should be written with a value between 2 and 63 inclusive indicating the target number of interrupts per milli-second.

22.2.16.6 MDIO Module Interrupt Events and Requests

The MDIO module generates two interrupt events:

- LINKINT0: Serial interface link change interrupt. Indicates a change in the state of the PHY link selected by the USERPHYSEL0 register
- USERINT0: Serial interface user command event complete interrupt selected by the USERACCESS0 register

22.2.16.6.1 Link Change Interrupt

The MDIO module asserts a link change interrupt (LINKINT0) if there is a change in the link state of the PHY corresponding to the address in the PHYADRMON bit in the MDIO register USERPHYSEL0, and if the LINKINTENB bit is also set in USERPHYSEL0. This interrupt event is also captured in the LINKINTRAW bit in the MDIO link status change interrupt register (LINKINTRAW). LINKINTRAW bits 0 and 1 correspond to USERPHYSEL0 and USERPHYSEL1, respectively.

When the interrupt is enabled and generated, the corresponding LINKINTMASKED bit is also set in the MDIO link status change interrupt register (LINKINTMASKED). The interrupt is cleared by writing back the same bit to LINKINTMASKED (write to clear).

The application software must acknowledge the EMAC subsystem after receiving MDIO interrupts by writing the appropriate C*n*MISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.16.6.2 User Access Completion Interrupt

When the GO bit in one of the MDIO register USERACCESS0 transitions from 1 to 0 (indicating completion of a user access) and the corresponding USERINTMASKSET bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) corresponding to USERACCESS0 is set, a user access completion interrupt (USERINT) is asserted. This interrupt event is also captured in the USERINTRAW bit in the MDIO user command complete interrupt register (USERINTRAW). USERINTRAW bits 0 and bit 1 correspond to USERACCESS0 and USERACCESS1, respectively.

When the interrupt is enabled and generated, the corresponding USERINTMASKED bit is also set in the MDIO user command complete interrupt register (USERINTMASKED). The interrupt is cleared by writing back the same bit to USERINTMASKED (write to clear).

The application software must acknowledge the EMAC subsystem module after receiving MDIO interrupts by writing the appropriate C*n*MISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 22.5.12 for the acknowledge key values.

22.2.17 Power Management

The clock stop interface allows the EMAC to be gracefully commanded to enter into a stopped clock state. When the (CLKSTOP_REQ) input is asserted, the EMAC suspend state is entered and the EMAC will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the TX cell FIFO will be transmitted. For receive, frames that are detected by the EMAC after the suspend state is entered are ignored. No statistics will be kept for ignored frames. The (CLKSTOP_ACK) output will be asserted when the EMAC_F enters the IDLE state indicating that the clock (VBUSP_CLK) may be stopped.

22.2.18 Emulation Considerations

22.2.18.1 EMAC Subsystem

The emulation control input (TBEMUSUSP) and EMAC submodule emulation control register allows EMAC subsystem operation to be completely or partially suspended. There is one EMAC subsystem submodule that contains an emulation control register (EMAC). The EMAC submodule emulation control register must be accessed to facilitate EMAC subsystem emulation control. The EMAC subsystem module enters the emulation suspend state if the EMAC submodule is configured for emulation suspend and the emulation suspend input is asserted. EMAC Emulation suspend occurs at packet boundaries. The emulation control feature is implemented for compatibility with other peripherals.

Table 22-16 shows the operation of the emulation control input and register bits.

Table 22-16. Emulation Control

EMUSUSP	SOFT	FREE	Description
0	X	X	Normal operation
1	0	0	Normal operation
1	1	0	Emulation suspend
1	X	1	Normal operation

22.2.18.2 EMAC Submodule

The emulation control input (TBEMUSUP) and register bits (SOFT and FREE in the EMCONTROL register) allow EMAC operation to be suspended. When the emulation suspend state is entered, the EMAC will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the TX cell FIFO will be transmitted. For receive, frames that are detected by the EMAC after the suspend state is entered are ignored. No statistics will be kept for ignored frames. Emulation control is implemented for compatibility with other peripherals.

Table 22-17 shows the operation of the emulation control input and register bits:

Table 22-17. Emulation Control

TBEMUSUSP	SOFT	FREE	Description
0	X	X	Normal operation
1	0	0	Normal operation
1	1	0	Emulation suspend
1	X	1	Normal operation

22.3 EMAC Subsystem Registers

Table 22-18 lists the memory-mapped registers for EMAC subsystem.

The base address of these registers is 0x5c00 0000.

Table 22-18. EMAC Subsystem Registers

Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	Section 22.3.1
4h	SOFTRESET	Software Reset Register	Section 22.3.2
Ch	INTCONTROL	Interrupt Control Register	Section 22.3.3
10h	C0RXTHRESHEN	Interrupt Core 0 Receive Threshold Interrupt Enable Register	Section 22.3.4
14h	C0RXEN	Interrupt Core 0 Receive Interrupt Enable Register	Section 22.3.5
18h	C0TXEN	Interrupt Core 0 Transmit Interrupt Enable Register	Section 22.3.6
1Ch	C0MISCEN	Interrupt Core 0 Miscellaneous Interrupt Enable Register	Section 22.3.7
20h	C1RXTHRESHEN	Interrupt Core 1 Receive Threshold Interrupt Enable Register	Section 22.3.4
24h	C1RXEN	Interrupt Core 1 Receive Interrupt Enable Register	Section 22.3.5
28h	C1TXEN	Interrupt Core 1 Transmit Interrupt Enable Register	Section 22.3.6
2Ch	C1MISCEN	Interrupt Core 1 Miscellaneous Interrupt Enable Register	Section 22.3.7
30h	C2RXTHRESHEN	Interrupt Core 2 Receive Threshold Interrupt Enable Register	Section 22.3.4
34h	C2RXEN	Interrupt Core 2 Receive Interrupt Enable Register	Section 22.3.5
38h	C2TXEN	Interrupt Core 2 Transmit Interrupt Enable Register	Section 22.3.6
3Ch	C2MISCEN	Interrupt Core 2 Miscellaneous Interrupt Enable Register	Section 22.3.7
40h	C0RXTHRESHSTAT	Interrupt Core 0 Receive Threshold Interrupt Status Register	Section 22.3.8
44h	C0RXSTAT	Interrupt Core 0 Receive Interrupt Status Register	Section 22.3.9
48h	C0TXSTAT	Interrupt Core 0 Transmit Interrupt Status Register	Section 22.3.10
4Ch	C0MISCSTAT	Interrupt Core 0 Miscellaneous Interrupt Status Register	Section 22.3.11
50h	C1RXTHRESHSTAT	Interrupt Core 1 Receive Threshold Interrupt Status Register	Section 22.3.8
54h	C1RXSTAT	Interrupt Core 1 Receive Interrupt Status Register	Section 22.3.9
58h	C1TXSTAT	Interrupt Core 1 Transmit Interrupt Status Register	Section 22.3.10
5Ch	C1MISCSTAT	Interrupt Core 1 Miscellaneous Interrupt Status Register	Section 22.3.11
60h	C2RXTHRESHSTAT	Interrupt Core 2 Receive Threshold Interrupt Status Register	Section 22.3.8
64h	C2RXSTAT	Interrupt Core 2 Receive Interrupt Status Register	Section 22.3.9
68h	C2TXSTAT	Interrupt Core 2 Transmit Interrupt Status Register	Section 22.3.10
6Ch	C2MISCSTAT	Interrupt Core 2 Miscellaneous Interrupt Status Register	Section 22.3.11
70h	C0RXIMAX	Interrupt Core 0 Receive Interrupts Per Millisecond Register	Section 22.3.12
74h	C0TXIMAX	Interrupt Core 0 Transmit Interrupts Per Millisecond Register	Section 22.3.13
78h	C1RXIMAX	Interrupt Core 1 Receive Interrupts Per Millisecond Register	Section 22.3.12
7Ch	C1TXIMAX	Interrupt Core 1 Transmit Interrupts Per Millisecond Register	Section 22.3.13
80h	C2RXIMAX	Interrupt Core 2 Receive Interrupts Per Millisecond Register	Section 22.3.12
84h	C2TXIMAX	Interrupt Core 2 Transmit Interrupts Per Millisecond Register	Section 22.3.13

22.3.1 Revision ID Register (REVID)

The Revision ID Register (REVID) is shown in Figure 22-10 and described in Table 22-19.

Figure 22-10. Revision ID Register (REVID)

31	16
REV	
R-4EC8h	

15	0
REV	
R-0100h	

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

Table 22-19. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV		Identifies the revision ID.
		4EC8 0100h	Current revision of the EMAC subsystem.

22.3.2 Software Reset Register (SOFTRESET)

The Software Reset Register (SOFTRESET) is shown in Figure 22-11 and described in Table 22-20.

Figure 22-11. Software Reset Register (SOFTRESET)

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		R/W-0

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

Table 22-20. Software Reset Register (SOFTRESET)

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RESET		Software reset bit for the EMAC subsystem. Clears the interrupt status, control registers, and CPPI Ram on the clock cycle following a write of 1.
		0	No software reset.
		1	Perform a software reset.

22.3.3 Interrupt Control Register (INTCONTROL)

The Interrupt Control Register (INTCONTROL) is shown in Figure 22-12 and described in Table 22-21

Figure 22-12. Interrupt Control Register (INTCONTROL)

31	30	24
INTTEST	Reserved	
R/W-0	R-0	

23	22	21	20	19	18	17	16
Reserved		C2TXPACEEN	C2RXPACEEN	C1TXPACEEN	C1RXPACEEN	C0TXPACEEN	C0RXPACEEN
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	12	11	0
Reserved		INTPRESCALE	
R-0		R/W-0	

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

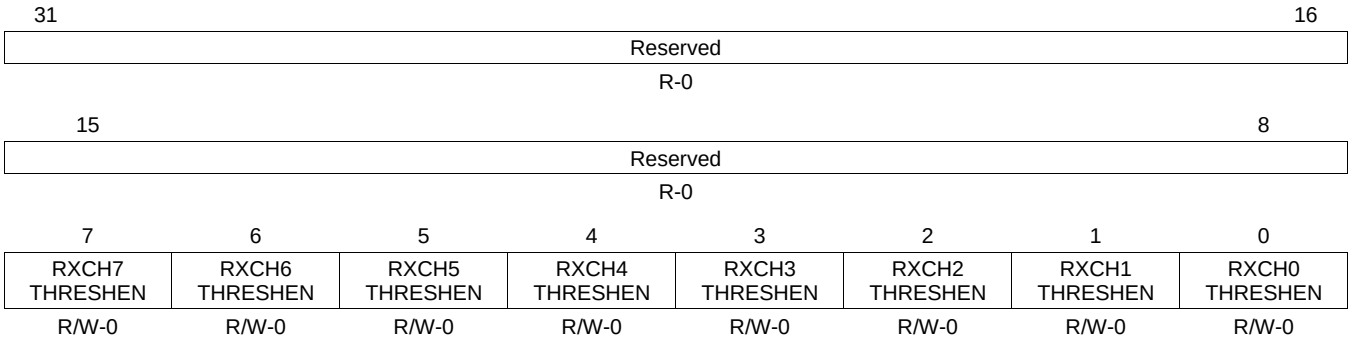
Table 22-21. Interrupt Control Register (INTCONTROL)

Bit	Field	Value	Description
31	INTTEST	0-1	Interrupt Test – Test bit to the interrupt pacing blocks
30-22	Reserved	0	Reserved
21	C2TXPACEEN		Enable pacing for TX interrupt pulse generation on Interrupt Core 2
		0	Pacing for TX interrupts on Core 2 disabled.
		1	Pacing for TX interrupts on Core 2 enabled.
20	C2RXPACEEN		Enable pacing for RX interrupt pulse generation on Interrupt Core 2
		0	Pacing for RX interrupts on Core 2 disabled.
		1	Pacing for RX interrupts on Core 2 enabled.
19	C1TXPACEEN		Enable pacing for TX interrupt pulse generation on Interrupt Core 1
		0	Pacing for TX interrupts on Core 1 disabled.
		1	Pacing for TX interrupts on Core 1 enabled.
18	C1RXPACEEN		Enable pacing for RX interrupt pulse generation on Interrupt Core 1
		0	Pacing for RX interrupts on Core 1 disabled.
		1	Pacing for RX interrupts on Core 1 enabled.
17	C0TXPACEEN		Enable pacing for TX interrupt pulse generation on Interrupt Core 0
		0	Pacing for TX interrupts on Core 0 disabled.
		1	Pacing for TX interrupts on Core 0 enabled.
16	C0RXPACEEN		Enable pacing for RX interrupt pulse generation on Interrupt Core 0
		0	Pacing for RX interrupts on Core 0 disabled.
		1	Pacing for RX interrupts on Core 0 enabled.
15-12	Reserved	0	Reserved
11-0	INTPRESCALE	0-7FFh	Number of VBUSP_CLK periods within a 4us time window.

22.3.4 Interrupt Core Receive Threshold Interrupt Enable Registers (C0RXTHRESHEN-C2RXTHRESHEN)

The Interrupt Core 0-2 Receive Threshold Interrupt Enable Register (C*n*RXTHRESHEN) is shown in Figure 22-13 and described in Table 22-22.

Figure 22-13. Interrupt Core 0-2 Receive Threshold Interrupt Enable Register (C*n*RXTHRESHEN)



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

Table 22-22. Interrupt Core 0-2 Receive Threshold Interrupt Enable Register (C*n*RXTHRESHEN)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 7
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 7.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 7.
6	RXCH6THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 6
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 6.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 6.
5	RXCH5THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 5
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 5.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 5.
4	RXCH4THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 4
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 4.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 4.
3	RXCH3THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 3
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 3.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 3.
2	RXCH2THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 2
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 2.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 2.
1	RXCH1THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 1
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 1.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 1.
0	RXCH0THRESHEN		Enable CnRXTHRESHPULSE interrupt generation for RX Channel 0
		0	CnRXTHRESHPULSE generation is disabled for RX Channel 0.
		1	CnRXTHRESHPULSE generation is enabled for RX Channel 0.

22.3.5 Interrupt Core Receive Interrupt Enable Registers (C0RXEN-C2RXEN)

The Interrupt Core 0-2 Receive Interrupt Enable Register (C*n*RXEN) is shown in Figure 22-14 and described in Table 22-23

Figure 22-14. Interrupt Core 0-2 Receive Interrupt Enable Register (C*n*RXEN)

31							16
Reserved							
R-0							
15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
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 22-23. Interrupt Core 0-2 Receive Interrupt Enable Register (C*n*RXEN)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN		Enable CnRXPULSE interrupt generation for RX Channel 7
		0	CnRXPULSE generation is disabled for RX Channel 7.
		1	CnRXPULSE generation is enabled for RX Channel 7.
6	RXCH6EN		Enable CnRXPULSE interrupt generation for RX Channel 6
		0	CnRXPULSE generation is disabled for RX Channel 6.
		1	CnRXPULSE generation is enabled for RX Channel 6.
5	RXCH5EN		Enable CnRXPULSE interrupt generation for RX Channel 5
		0	CnRXPULSE generation is disabled for RX Channel 5.
		1	CnRXPULSE generation is enabled for RX Channel 5.
4	RXCH4EN		Enable CnRXPULSE interrupt generation for RX Channel 4
		0	CnRXPULSE generation is disabled for RX Channel 4.
		1	CnRXPULSE generation is enabled for RX Channel 4.
3	RXCH3EN		Enable CnRXPULSE interrupt generation for RX Channel 3
		0	CnRXPULSE generation is disabled for RX Channel 3.
		1	CnRXPULSE generation is enabled for RX Channel 3.
2	RXCH2EN		Enable CnRXPULSE interrupt generation for RX Channel 2
		0	CnRXPULSE generation is disabled for RX Channel 2.
		1	CnRXPULSE generation is enabled for RX Channel 2.
1	RXCH1EN		Enable CnRXPULSE interrupt generation for RX Channel 1
		0	CnRXPULSE generation is disabled for RX Channel 1.
		1	CnRXPULSE generation is enabled for RX Channel 1.
0	RXCH0EN		Enable CnRXPULSE interrupt generation for RX Channel 0
		0	CnRXPULSE generation is disabled for RX Channel 0.
		1	CnRXPULSE generation is enabled for RX Channel 0.

22.3.6 Interrupt Core Transmit Interrupt Enable Registers (C0TXEN-C2TXEN)

The Interrupt Core 0-2 Transmit Interrupt Enable Register (C*n*TXEN) is shown in Figure 22-15 and described in Table 22-24

Figure 22-15. Interrupt Core 0-2 Transmit Interrupt Enable Register (C*n*TXEN)

31							16
Reserved							
R-0							
15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
TXCH7EN	TXCH6EN	TXCH5EN	TXCH4EN	TXCH3EN	TXCH2EN	TXCH1EN	TXCH0EN
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 22-24. Interrupt Core 0-2 Transmit Interrupt Enable Register (C*n*TXEN)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TXCH7EN		Enable CnTXPULSE interrupt generation for TX Channel 7
		0	CnTXPULSE generation is disabled for TX Channel 7.
		1	CnTXPULSE generation is enabled for TX Channel 7.
6	TXCH6EN		Enable CnTXPULSE interrupt generation for TX Channel 6
		0	CnTXPULSE generation is disabled for TX Channel 6.
		1	CnTXPULSE generation is enabled for TX Channel 6.
5	TXCH5EN		Enable CnTXPULSE interrupt generation for TX Channel 5
		0	CnTXPULSE generation is disabled for TX Channel 5.
		1	CnTXPULSE generation is enabled for TX Channel 5.
4	TXCH4EN		Enable CnTXPULSE interrupt generation for TX Channel 4
		0	CnTXPULSE generation is disabled for TX Channel 4.
		1	CnTXPULSE generation is enabled for TX Channel 4.
3	TXCH3EN		Enable CnTXPULSE interrupt generation for TX Channel 3
		0	CnTXPULSE generation is disabled for TX Channel 3.
		1	CnTXPULSE generation is enabled for TX Channel 3.
2	TXCH2EN		Enable CnTXPULSE interrupt generation for TX Channel 2
		0	CnTXPULSE generation is disabled for TX Channel 2.
		1	CnTXPULSE generation is enabled for TX Channel 2.
1	TXCH1EN		Enable CnTXPULSE interrupt generation for TX Channel 1
		0	CnTXPULSE generation is disabled for TX Channel 1.
		1	CnTXPULSE generation is enabled for TX Channel 1.
0	TXCH0EN		Enable CnTXPULSE interrupt generation for TX Channel 0
		0	CnTXPULSE generation is disabled for TX Channel 0.
		1	CnTXPULSE generation is enabled for TX Channel 0.

22.3.7 Interrupt Core Miscellaneous Interrupt Enable Registers (C0MISCEN-C2MISCEN)

The Interrupt Core 0-2 Miscellaneous Interrupt Enable Register (C*n*MISCEN) is shown in Figure 22-16 and described in Table 22-25

Figure 22-16. Interrupt Core 0-2 Miscellaneous Interrupt Enable Register (C*n*MISCEN)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		STATPENDEN	HOSTPENDEN	LINKINT0EN	USERINT0EN
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 22-25. Interrupt Core 0-2 Miscellaneous Interrupt Enable Register (C*n*MISCEN)

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	STATPENDEN		Enable CnMISCPULSE interrupt generation when EMAC statistics interrupts are generated
		0	CnMISCPULSE generation is disabled for EMAC STATPEND interrupts.
		1	CnMISCPULSE generation is enabled for EMAC STATPEND interrupts.
2	HOSTPENDEN		Enable CnMISCPULSE interrupt generation when EMAC host interrupts are generated
		0	CnMISCPULSE generation is disabled for EMAC HOSTPEND interrupts.
		1	CnMISCPULSE generation is enabled for EMAC HOSTPEND interrupts.
1	LINKINT0EN		Enable CnMISCPULSE interrupt generation when MDIO LINKINT0 interrupts (corresponding to USERPHYSEL0) are generated
		0	CnMISCPULSE generation is disabled for MDIO LINKINT0 interrupts.
		1	CnMISCPULSE generation is enabled for MDIO LINKINT0 interrupts.
0	USERINT0EN		Enable CnMISCPULSE interrupt generation when MDIO USERINT0 interrupts (corresponding to USERACCESS0) are generated
		0	CnMISCPULSE generation is disabled for MDIO USERINT0.
		1	CnMISCPULSE generation is enabled for MDIO USERINT0.

22.3.8 Interrupt Core Receive Threshold Interrupt Status Registers (C0RXTHRESHSTAT-C2RXTHRESHSTAT)

The Interrupt Core 0-2 Receive Threshold Interrupt Status Register (C*n*RXTHRESHSTAT) is shown in Figure 22-17 and described in Table 22-26

Figure 22-17. Interrupt Core 0-2 Receive Threshold Interrupt Status Register (C*n*RXTHRESHSTAT)

31							16
Reserved							
R-0							

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RXCH7THRESHSTAT	RXCH6THRESHSTAT	RXCH5THRESHSTAT	RXCH4THRESHSTAT	RXCH3THRESHSTAT	RXCH2THRESHSTAT	RXCH1THRESHSTAT	RXCH0THRESHSTAT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

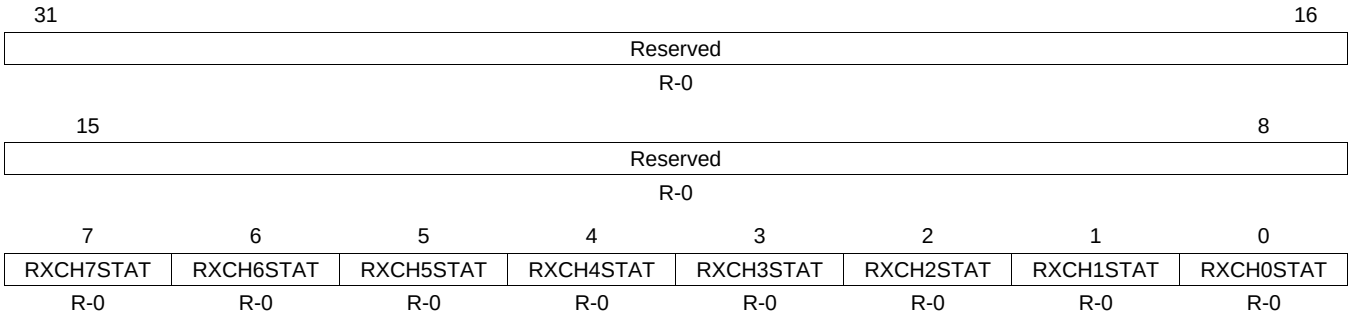
Table 22-26. Interrupt Core 0-2 Receive Threshold Interrupt Status Register (C*n*RXTHRESHSTAT)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7THRESHSTAT		Interrupt status for RX Channel 7 masked by the CnRXTHRESHEN register
		0	RX Channel 7 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 7 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
6	RXCH6THRESHSTAT		Interrupt status for RX Channel 6 masked by the CnRXTHRESHEN register
		0	RX Channel 6 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 6 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
5	RXCH5THRESHSTAT		Interrupt status for RX Channel 5 masked by the CnRXTHRESHEN register
		0	RX Channel 5 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 5 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
4	RXCH4THRESHSTAT		Interrupt status for RX Channel 4 masked by the CnRXTHRESHEN register
		0	RX Channel 4 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 4 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
3	RXCH3THRESHSTAT		Interrupt status for RX Channel 3 masked by the CnRXTHRESHEN register
		0	RX Channel 3 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 3 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
2	RXCH2THRESHSTAT		Interrupt status for RX Channel 2 masked by the CnRXTHRESHEN register
		0	RX Channel 2 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 2 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
1	RXCH1THRESHSTAT		Interrupt status for RX Channel 1 masked by the CnRXTHRESHEN register
		0	RX Channel 1 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 1 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.
0	RXCH0THRESHSTAT		Interrupt status for RX Channel 0 masked by the CnRXTHRESHEN register
		0	RX Channel 0 does not satisfy conditions to generate a CnRXTHRESHPULSE interrupt.
		1	RX Channel 0 satisfies conditions to generate a CnRXTHRESHPULSE interrupt.

22.3.9 Interrupt Core Receive Interrupt Status Registers (C0RXSTAT-C2RXSTAT)

The Interrupt Core 0-2 Receive Interrupt Status Register (C*n*RXSTAT) is shown in Figure 22-18 and described in Table 22-27

Figure 22-18. Interrupt Core 0-2 Receive Interrupt Status Register (C*n*RXSTAT)



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

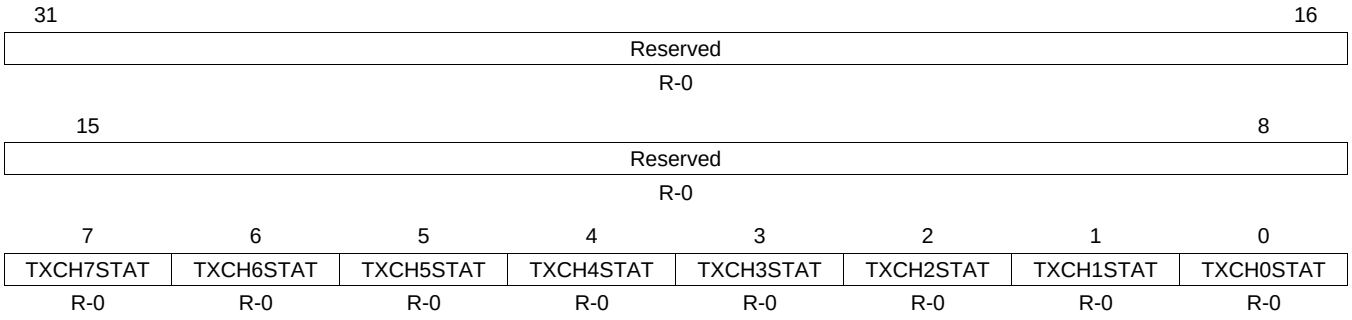
Table 22-27. Interrupt Core 0-2 Receive Interrupt Status Register (C*n*RXSTAT)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7STAT		Interrupt status for RX Channel 7 masked by the CnRXEN register
		0	RX Channel 7 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 7 satisfies conditions to generate a CnRXPULSE interrupt.
6	RXCH6STAT		Interrupt status for RX Channel 6 masked by the CnRXEN register
		0	RX Channel 6 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 6 satisfies conditions to generate a CnRXPULSE interrupt.
5	RXCH5STAT		Interrupt status for RX Channel 5 masked by the CnRXEN register
		0	RX Channel 5 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 5 satisfies conditions to generate a CnRXPULSE interrupt.
4	RXCH4STAT		Interrupt status for RX Channel 4 masked by the CnRXEN register
		0	RX Channel 4 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 4 satisfies conditions to generate a CnRXPULSE interrupt.
3	RXCH3STAT		Interrupt status for RX Channel 3 masked by the CnRXEN register
		0	RX Channel 3 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 3 satisfies conditions to generate a CnRXPULSE interrupt.
2	RXCH2STAT		Interrupt status for RX Channel 2 masked by the CnRXEN register
		0	RX Channel 2 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 2 satisfies conditions to generate a CnRXPULSE interrupt.
1	RXCH1STAT		Interrupt status for RX Channel 1 masked by the CnRXEN register
		0	RX Channel 1 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 1 satisfies conditions to generate a CnRXPULSE interrupt.
0	RXCH0STAT		Interrupt status for RX Channel 0 masked by the CnRXEN register
		0	RX Channel 0 does not satisfy conditions to generate a CnRXPULSE interrupt.
		1	RX Channel 0 satisfies conditions to generate a CnRXPULSE interrupt.

22.3.10 Interrupt Core Transmit Interrupt Status Registers (C0TXSTAT-C2TXSTAT)

The Interrupt Core 0-2 Transmit Interrupt Status Register (C*n*TXSTAT) is shown in Figure 22-19 and described in Table 22-28

Figure 22-19. Interrupt Core 0-2 Transmit Interrupt Status Register (C*n*TXSTAT)



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

Table 22-28. Interrupt Core 0-2 Transmit Interrupt Status Register (C*n*TXSTAT)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TXCH7STAT		Interrupt status for TX Channel 7 masked by the CnTXEN register
		0	TX Channel 7 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 7 satisfies conditions to generate a CnTXPULSE interrupt.
6	TXCH6STAT		Interrupt status for TX Channel 6 masked by the CnTXEN register
		0	TX Channel 6 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 6 satisfies conditions to generate a CnTXPULSE interrupt.
5	TXCH5STAT		Interrupt status for TX Channel 5 masked by the CnTXEN register
		0	TX Channel 5 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 5 satisfies conditions to generate a CnTXPULSE interrupt.
4	TXCH4STAT		Interrupt status for TX Channel 4 masked by the CnTXEN register
		0	TX Channel 4 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 4 satisfies conditions to generate a CnTXPULSE interrupt.
3	TXCH3STAT		Interrupt status for TX Channel 3 masked by the CnTXEN register
		0	TX Channel 3 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 3 satisfies conditions to generate a CnTXPULSE interrupt.
2	TXCH2STAT		Interrupt status for TX Channel 2 masked by the CnTXEN register
		0	TX Channel 2 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 2 satisfies conditions to generate a CnTXPULSE interrupt.
1	TXCH1STAT		Interrupt status for TX Channel 1 masked by the CnTXEN register
		0	TX Channel 1 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 1 satisfies conditions to generate a CnTXPULSE interrupt.
0	TXCH0STAT		Interrupt status for TX Channel 0 masked by the CnTXEN register
		0	TX Channel 0 does not satisfy conditions to generate a CnTXPULSE interrupt.
		1	TX Channel 0 satisfies conditions to generate a CnTXPULSE interrupt.

22.3.11 Interrupt Core Miscellaneous Interrupt Status Registers (C0MISCSTAT-C2MISCSTAT)

The Interrupt Core 0-2 Miscellaneous Interrupt Status Register (C*n*MISCSTAT) is shown in Figure 22-20 and described in Table 22-29

Figure 22-20. Interrupt Core 0-2 Miscellaneous Interrupt Status Register (C*n*MISCSTAT)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		STATPENDSTAT	HOSTPENDSTAT	LINKINT0STAT	USERINT0STAT
R-0		R-0	R-0	R-0	R-0

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

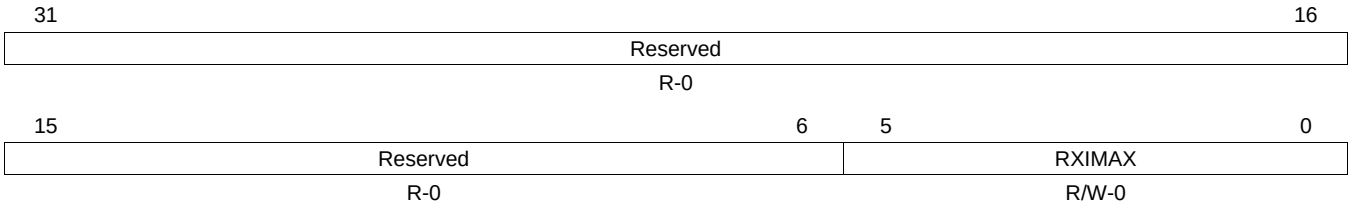
Table 22-29. Interrupt Core 0-2 Miscellaneous Interrupt Status Register (C*n*MISCSTAT)

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	STATPENDSTAT		Interrupt status for EMAC STATPEND masked by the CnMISCEN register
		0	EMAC STATPEND does not satisfy conditions to generate a CnMISCPULSE interrupt.
		1	EMAC STATPEND satisfies conditions to generate a CnMISCPULSE interrupt.
2	HOSTPENDSTAT		Interrupt status for EMAC HOSTPEND masked by the CnMISCEN register
		0	EMAC HOSTPEND does not satisfy conditions to generate a CnMISCPULSE interrupt.
		1	EMAC HOSTPEND satisfies conditions to generate a CnMISCPULSE interrupt.
1	LINKINT0STAT		Interrupt status for MDIO LINKINT0 masked by the CnMISCEN register
		0	MDIO LINKINT0 does not satisfy conditions to generate a CnMISCPULSE interrupt.
		1	MDIO LINKINT0 satisfies conditions to generate a CnMISCPULSE interrupt.
0	USERINT0STAT		Interrupt status for MDIO USERINT0 masked by the CnMISCEN register
		0	MDIO USERINT0 does not satisfy conditions to generate a CnMISCPULSE interrupt.
		1	MDIO USERINT0 satisfies conditions to generate a CnMISCPULSE interrupt.

22.3.12 Interrupt Core Receive Interrupts Per Millisecond Registers (C0RXIMAX-C2RXIMAX)

The Interrupt Core 0-2 Receive Interrupts Per Millisecond Register (C*n*RXIMAX) is shown in Figure 22-21 and described in Table 22-30

Figure 22-21. Interrupt Core 0-2 Receive Interrupts Per Millisecond Register (C*n*RXIMAX)



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

Table 22-30. Interrupt Core 0-2 Receive Interrupts Per Millisecond Register (C*n*RXIMAX)

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	RXIMAX	0-3Fh	RXIMAX is the maximum number of CnRXPULSE interrupts generated per millisecond when CnRXPACEEN is enabled in INTCONTROL.

22.3.13 Interrupt Core Transmit Interrupts Per Millisecond Registers (C0TXIMAX-C2TXIMAX)

The Interrupt Core 0-2 Transmit Interrupts Per Millisecond Register (C*n*TXIMAX) is shown in Figure 22-22 and described in Table 22-31

Figure 22-22. Interrupt Core 0-2 Transmit Interrupts Per Millisecond Register (C*n*TXIMAX)

31	16
Reserved	
R-0	

15	6	5	0
Reserved		TXIMAX	
R-0		R/W-0	

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

Table 22-31. Interrupt Core 0-2 Transmit Interrupts Per Millisecond Register (C*n*TXIMAX)

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	TXIMAX	0-3Fh	TXIMAX is the maximum number of CnTXPULSE interrupts generated per millisecond when CnTXPACEEN is enabled in INTCONTROL.

22.4 MDIO Registers

Table 22-32 lists the memory-mapped registers for the MDIO module. The base address of these registers is 0x5C03 0000.

Table 22-32. Management Data Input/Output (MDIO) Registers

Offset	Acronym	Register Description	Section
0h	REVID	MDIO Revision ID Register	Section 22.4.1
4h	CONTROL	MDIO Control Register	Section 22.4.2
8h	ALIVE	PHY Alive Status register	Section 22.4.3
Ch	LINK	PHY Link Status Register	Section 22.4.4
10h	LINKINTRAW	MDIO Link Status Change Interrupt (Unmasked) Register	Section 22.4.5
14h	LINKINTMASKED	MDIO Link Status Change Interrupt (Masked) Register	Section 22.4.6
20h	USERINTRAW	MDIO User Command Complete Interrupt (Unmasked) Register	Section 22.4.7
24h	USERINTMASKED	MDIO User Command Complete Interrupt (Masked) Register	Section 22.4.8
28h	USERINTMASKSET	MDIO User Command Complete Interrupt Mask Set Register	Section 22.4.9
2Ch	USERINTMASKCLEAR	MDIO User Command Complete Interrupt Mask Clear Register	Section 22.4.10
80h	USERACCESS0	MDIO User Access Register 0	Section 22.4.11
84h	USERPHYSEL0	MDIO User PHY Select Register 0	Section 22.4.12
88h	USERACCESS1	MDIO User Access Register 1	Section 22.4.13
8Ch	USERPHYSEL1	MDIO User PHY Select Register 1	Section 22.4.14

22.4.1 MDIO Revision ID Register (REVID)

The MDIO Revision ID Register (REVID) is shown in Figure 22-23 and described in Table 22-33.

Figure 22-23. MDIO Revision ID Register (REVID)

31	16
REV	
R-0007h	

15	0
REV	
R-0105h	

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

Table 22-33. MDIO Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV		Identifies the MDIO Module revision.
		0007 0105h	Current revision of the MDIO Module.

22.4.2 MDIO Control Register (CONTROL)

The MDIO control register (CONTROL) is shown in Figure 22-24 and described in Table 22-34.

Figure 22-24. MDIO Control Register (CONTROL)

31	30	29	28	24
IDLE	ENABLE	Reserved	HIGHEST_USER_CHANNEL	
R-1	R/W-0	R-0	R-1	

23	21	20	19	18	17	16
Reserved		PREAMBLE	FAULT	FAULTENB	INT_TEST_ENABLE	Reserved
R-0		R/W-0	R/WC-0	R/W-0	R/W-0	R-0

15	0
CLKDIV	
R/W-FFh	

LEGEND: R/W = R = Read only; R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

Table 22-34. MDIO Control Register (CONTROL) Field Descriptions

Bit	Field	Value	Description
31	IDLE		State machine IDLE status bit.
		0	State machine is not in idle state.
		1	State machine is in idle state.
30	ENABLE		State machine enable control bit. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit.
		0	Disables the MDIO state machine.
		1	Enable the MDIO state machine.
29	Reserved	0	Reserved
28-24	HIGHEST_USER_CHANNEL	0-1Fh	Highest user channel that is available in the module. It is currently set to 1. This implies that MDIOUserAccess1 is the highest available user access channel.
23-21	Reserved	0	Reserved
20	PREAMBLE		Preamble disable
		0	Standard MDIO preamble is used.
		1	Disables this device from sending MDIO frame preambles.
19	FAULT		Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to it clears this bit.
		0	No failure
		1	Physical layer fault; the MDIO state machine is reset.
18	FAULTENB		Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection.
		0	Disables the physical layer fault detection.
		1	Enables the physical layer fault detection.
17	INT_TEST_ENABLE		Interrupt test enable. This bit can be set to 1 to enable the host to set the USERINT and LINKINT bits for test purposes.
		0	
		1	
16	Reserved	0	Reserved
15-0	CLKDIV	0-FFFFh	Clock Divider bits. This field specifies the division ratio between the peripheral clock and the frequency of MDIO_CLK. MDIO_CLK is disabled when CLKDIV is cleared to 0. MDIO_CLK frequency = peripheral clock frequency/(CLKDIV + 1).

22.4.3 PHY Acknowledge Status Register (ALIVE)

The PHY acknowledge status register (ALIVE) is shown in Figure 22-25 and described in Table 22-35.

Figure 22-25. PHY Acknowledge Status Register (ALIVE)

31	16
ALIVE	
R/WC-0	

15	0
ALIVE	
R/WC-0	

LEGEND: R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

Table 22-35. PHY Acknowledge Status Register (ALIVE) Field Descriptions

Bit	Field	Value	Description
31-0	ALIVE		MDIO Alive bits. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY; the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.
		0	The PHY fails to acknowledge the access.
		1	The most recent access to the PHY with an address corresponding to the register bit number was acknowledged by the PHY.

22.4.4 PHY Link Status Register (LINK)

The PHY link status register (LINK) is shown in Figure 22-26 and described in Table 22-36.

Figure 22-26. PHY Link Status Register (LINK)

31	16
LINK	
R-0	

15	0
LINK	
R-0	

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

Table 22-36. PHY Link Status Register (LINK) Field Descriptions

Bit	Field	Value	Description
31-0	LINK		MDIO Link state bits. This register is updated after a read of the generic status register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect.
		0	The PHY indicates it does not have a link or fails to acknowledge the read transaction
		1	The PHY with the corresponding address has a link and the PHY acknowledges the read transaction.

22.4.5 MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)

The MDIO link status change interrupt (unmasked) register (LINKINTRAW) is shown in Figure 22-27 and described in Table 22-37.

Figure 22-27. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERPHY1	USERPHY0
R-0		R/WC-0	R/WC-0

LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

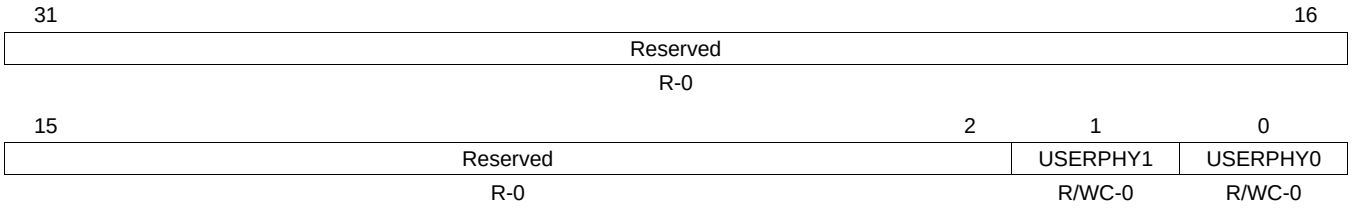
Table 22-37. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERPHY1		MDIO Link change event, raw value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL1. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL1
0	USERPHY0		MDIO Link change event, raw value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL0. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL0

22.4.6 MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)

The MDIO link status change interrupt (masked) register (LINKINTMASKED) is shown in Figure 22-28 and described in Table 22-38.

Figure 22-28. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)



LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

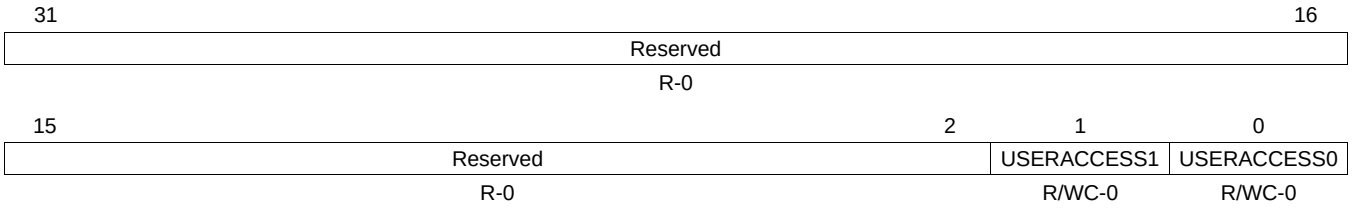
Table 22-38. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERPHY1		MDIO Link change interrupt, masked value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL1 and the corresponding LINKINTENB bit was set. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL1 and the LINKINTENB bit in USERPHYSEL1 is set to 1.
0	USERPHY0		MDIO Link change interrupt, masked value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL0 and the corresponding LINKINTENB bit was set. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL0 and the LINKINTENB bit in USERPHYSEL0 is set to 1.

22.4.7 MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)

The MDIO user command complete interrupt (unmasked) register (USERINTRAW) is shown in Figure 22-29 and described in Table 22-39.

Figure 22-29. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)



LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

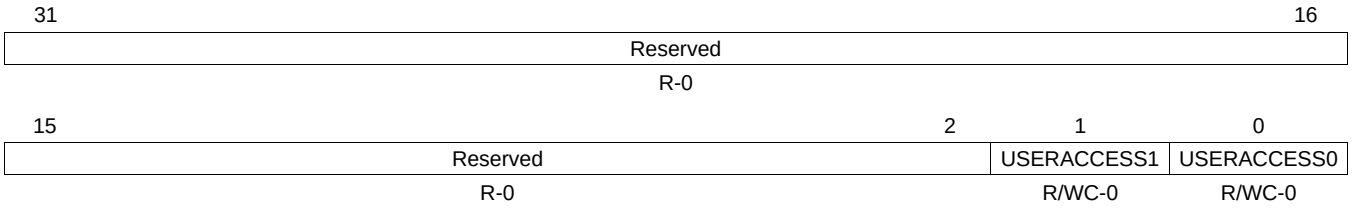
Table 22-39. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		MDIO User command complete event bit. When asserted, the bit indicates that the previously scheduled PHY read or write command using the USERACCESS1 register has completed. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS1 has completed.
0	USERACCESS0		MDIO User command complete event bit. When asserted, the bit indicates that the previously scheduled PHY read or write command using the USERACCESS0 register has completed. Writing a 1 will clear the event and writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS0 has completed.

22.4.8 MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)

The MDIO user command complete interrupt (masked) register (USERINTMASKED) is shown in Figure 22-30 and described in Table 22-40.

Figure 22-30. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)



LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

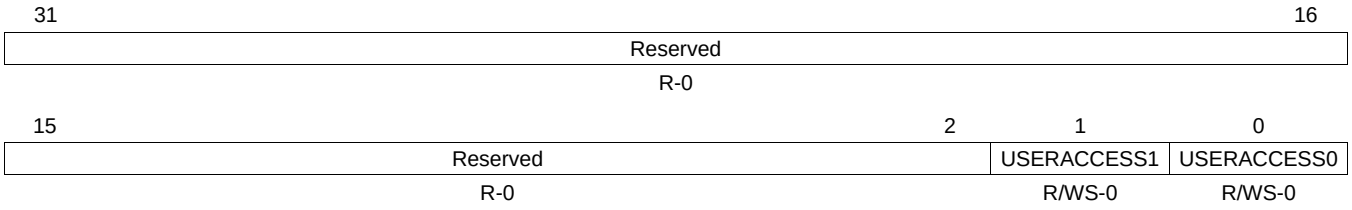
Table 22-40. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		Masked value of MDIO User command complete interrupt. When asserted, The bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS1 register has completed. Writing a 1 will clear the interrupt and writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS1 has completed and the corresponding bit in USERINTMASKSET is set to 1.
0	USERACCESS0		Masked value of MDIO User command complete interrupt. When asserted, The bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS0 register has completed. Writing a 1 will clear the interrupt and writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS0 has completed and the corresponding bit in USERINTMASKSET is set to 1.

22.4.9 MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)

The MDIO user command complete interrupt mask set register (USERINTMASKSET) is shown in Figure 22-31 and described in Table 22-41.

Figure 22-31. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)



LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set; -*n* = value after reset

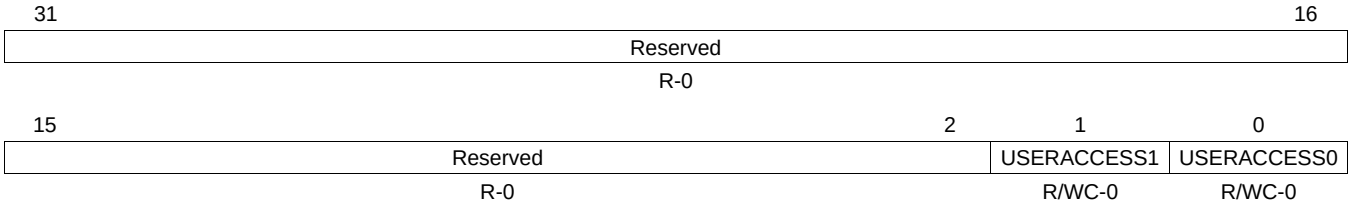
Table 22-41. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		MDIO user interrupt mask set for USERINTMASKED[1]. Setting a bit to 1 will enable MDIO user command complete interrupts for the USERACCESS1 register. MDIO user interrupt for USERACCESS1 is disabled if the corresponding bit is 0. Writing a 0 to this register has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.
0	USERACCESS0		MDIO user interrupt mask set for USERINTMASKED[0]. Setting a bit to 1 will enable MDIO user command complete interrupts for the USERACCESS0 register. MDIO user interrupt for USERACCESS0 is disabled if the corresponding bit is 0. Writing a 0 to this register has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.

22.4.10 MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)

The MDIO user command complete interrupt mask clear register (USERINTMASKCLEAR) is shown in Figure 22-32 and described in Table 22-42.

Figure 22-32. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)



LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear; -*n* = value after reset

Table 22-42. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		MDIO user command complete interrupt mask clear for USERINTMASKED[1]. Setting the bit to 1 will disable further user command complete interrupts for USERACCESS1. Writing a 0 to this register has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is enabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is disabled.
0	USERACCESS0		MDIO user command complete interrupt mask clear for USERINTMASKED[0]. Setting the bit to 1 will disable further user command complete interrupts for USERACCESS0. Writing a 0 to this register has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.

22.4.11 MDIO User Access Register 0 (USERACCESS0)

The MDIO user access register 0 (USERACCESS0) is shown in Figure 22-33 and described in Table 22-43.

Figure 22-33. MDIO User Access Register 0 (USERACCESS0)

31	30	29	28 26	25 21	20 16
GO	WRITE	ACK	Reserved	REGADR	PHYADR
R/WS-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 0
DATA
R/W-0

LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set; -*n* = value after reset

Table 22-43. MDIO User Access Register 0 (USERACCESS0) Field Descriptions

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to USERACCESS0 are blocked when the GO bit is 1.
30	WRITE		Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read.
		0	The user command is a read operation.
		1	The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

22.4.12 MDIO User PHY Select Register 0 (USERPHYSEL0)

The MDIO user PHY select register 0 (USERPHYSEL0) is shown in Figure 22-34 and described in Table 22-44.

Figure 22-34. MDIO User PHY Select Register 0 (USERPHYSEL0)

31	16
Reserved	
R-0	

15	8	7	6	5	4	0
Reserved		LINKSEL	LINKINTENB	Rsvd	PHYADRMON	
R-0		R/W-0	R/W-0	R-0	R/W-0	

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

Table 22-44. MDIO User PHY Select Register 0 (USERPHYSEL0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	LINKSEL		Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device.
		0	The link status is determined by the MDIO state machine.
		1	Not supported.
6	LINKINTENB		Link change interrupt enable. Set to 1 to enable link change status interrupts for PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is cleared to 0.
		0	Link change interrupts are disabled.
		1	Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled.
5	Reserved	0	Reserved
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

22.4.13 MDIO User Access Register 1 (USERACCESS1)

The MDIO user access register 1 (USERACCESS1) is shown in Figure 22-35 and described in Table 22-45.

Figure 22-35. MDIO User Access Register 1 (USERACCESS1)

31	30	29	28 26	25 21	20 16
GO	WRITE	ACK	Reserved	REGADR	PHYADR
R/WS-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 0
DATA
R/W-0

LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set; -*n* = value after reset

Table 22-45. MDIO User Access Register 1 (USERACCESS1) Field Descriptions

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to USERACCESS0 are blocked when the GO bit is 1.
30	WRITE		Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read.
		0	The user command is a read operation.
		1	The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

22.4.14 MDIO User PHY Select Register 1 (USERPHYSEL1)

The MDIO user PHY select register 1 (USERPHYSEL1) is shown in Figure 22-36 and described in Table 22-46.

Figure 22-36. MDIO User PHY Select Register 1 (USERPHYSEL1)

31	16
Reserved	
R-0	

15	8	7	6	5	4	0
Reserved		LINKSEL	LINKINTENB	Rsvd	PHYADRMON	
R-0		R/W-0	R/W-0	R-0	R/W-0	

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

Table 22-46. MDIO User PHY Select Register 1 (USERPHYSEL1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	LINKSEL		Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device.
		0	The link status is determined by the MDIO state machine.
		1	Not supported.
6	LINKINTENB		Link change interrupt enable. Set to 1 to enable link change status interrupts for the PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is cleared to 0.
		0	Link change interrupts are disabled.
		1	Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled.
5	Reserved	0	PHY address whose link status is to be monitored.
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

22.5 EMAC Module Registers

Table 22-47 lists the memory-mapped registers for the EMAC. The base address of these registers is 0x5C01 0000.

Table 22-47. Ethernet Media Access Controller (EMAC) Registers

Offset	Acronym	Register Description	Section
0h	TXREVID	Transmit Revision ID Register	Section 22.5.1
4h	TXCONTROL	Transmit Control Register	Section 22.5.2
8h	TXTEARDOWN	Transmit Teardown Register	Section 22.5.3
10h	RXREVID	Receive Revision ID Register	Section 22.5.4
14h	RXCONTROL	Receive Control Register	Section 22.5.5
18h	RXTEARDOWN	Receive Teardown Register	Section 22.5.6
80h	TXINTSTATRAW	Transmit Interrupt Status (Unmasked) Register	Section 22.5.7
84h	TXINTSTATMASKED	Transmit Interrupt Status (Masked) Register	Section 22.5.8
88h	TXINTMASKSET	Transmit Interrupt Mask Set Register	Section 22.5.9
8Ch	TXINTMASKCLEAR	Transmit Interrupt Clear Register	Section 22.5.10
90h	MACINVECTOR	MAC Input Vector Register	Section 22.5.11
94h	MACEOIVECTOR	MAC End Of Interrupt Vector Register	Section 22.5.12
A0h	RXINTSTATRAW	Receive Interrupt Status (Unmasked) Register	Section 22.5.13
A4h	RXINTSTATMASKED	Receive Interrupt Status (Masked) Register	Section 22.5.14
A8h	RXINTMASKSET	Receive Interrupt Mask Set Register	Section 22.5.15
ACh	RXINTMASKCLEAR	Receive Interrupt Mask Clear Register	Section 22.5.16
B0h	MACINTSTATRAW	MAC Interrupt Status (Unmasked) Register	Section 22.5.17
B4h	MACINTSTATMASKED	MAC Interrupt Status (Masked) Register	Section 22.5.18
B8h	MACINTMASKSET	MAC Interrupt Mask Set Register	Section 22.5.19
BCh	MACINTMASKCLEAR	MAC Interrupt Mask Clear Register	Section 22.5.20
100h	RXMBPENABLE	Receive Multicast/Broadcast/Promiscuous Channel Enable Register	Section 22.5.21
104h	RXUNICASTSET	Receive Unicast Enable Set Register	Section 22.5.22
108h	RXUNICASTCLEAR	Receive Unicast Clear Register	Section 22.5.23
10Ch	RXMAXLEN	Receive Maximum Length Register	Section 22.5.24
110h	RXBUFFEROFFSET	Receive Buffer Offset Register	Section 22.5.25
114h	RXFILTERLOWTHRESH	Receive Filter Low Priority Frame Threshold Register	Section 22.5.26
120h	RX0FLOWTHRESH	Receive Channel 0 Flow Control Threshold Register	Section 22.5.27
124h	RX1FLOWTHRESH	Receive Channel 1 Flow Control Threshold Register	Section 22.5.27
128h	RX2FLOWTHRESH	Receive Channel 2 Flow Control Threshold Register	Section 22.5.27
12Ch	RX3FLOWTHRESH	Receive Channel 3 Flow Control Threshold Register	Section 22.5.27
130h	RX4FLOWTHRESH	Receive Channel 4 Flow Control Threshold Register	Section 22.5.27
134h	RX5FLOWTHRESH	Receive Channel 5 Flow Control Threshold Register	Section 22.5.27
138h	RX6FLOWTHRESH	Receive Channel 6 Flow Control Threshold Register	Section 22.5.27
13Ch	RX7FLOWTHRESH	Receive Channel 7 Flow Control Threshold Register	Section 22.5.27
140h	RX0FREEBUFFER	Receive Channel 0 Free Buffer Count Register	Section 22.5.28
144h	RX1FREEBUFFER	Receive Channel 1 Free Buffer Count Register	Section 22.5.28
148h	RX2FREEBUFFER	Receive Channel 2 Free Buffer Count Register	Section 22.5.28
14Ch	RX3FREEBUFFER	Receive Channel 3 Free Buffer Count Register	Section 22.5.28
150h	RX4FREEBUFFER	Receive Channel 4 Free Buffer Count Register	Section 22.5.28
154h	RX5FREEBUFFER	Receive Channel 5 Free Buffer Count Register	Section 22.5.28
158h	RX6FREEBUFFER	Receive Channel 6 Free Buffer Count Register	Section 22.5.28
15Ch	RX7FREEBUFFER	Receive Channel 7 Free Buffer Count Register	Section 22.5.28
160h	MACCONTROL	MAC Control Register	Section 22.5.29
164h	MACSTATUS	MAC Status Register	Section 22.5.30

Table 22-47. Ethernet Media Access Controller (EMAC) Registers (continued)

Offset	Acronym	Register Description	Section
168h	EMCONTROL	Emulation Control Register	Section 22.5.31
16Ch	FIFOCONTROL	FIFO Control Register	Section 22.5.32
170h	MACCONFIG	MAC Configuration Register	Section 22.5.33
174h	SOFTRESET	Soft Reset Register	Section 22.5.34
1D0h	MACSRCADDRLO	MAC Source Address Low Bytes Register	Section 22.5.35
1D4h	MACSRCADDRHI	MAC Source Address High Bytes Register	Section 22.5.36
1D8h	MACHASH1	MAC Hash Address Register 1	Section 22.5.37
1DCh	MACHASH2	MAC Hash Address Register 2	Section 22.5.38
1E0h	BOFFTEST	Back Off Test Register	Section 22.5.39
1E4h	TPACETEST	Transmit Pacing Algorithm Test Register	Section 22.5.40
1E8h	RXPAUSE	Receive Pause Timer Register	Section 22.5.41
1ECh	TXPAUSE	Transmit Pause Timer Register	Section 22.5.42
500h	MACADDRLO	MAC Address Low Bytes Register, Used in Receive Address Matching	Section 22.5.43
504h	MACADDRHI	MAC Address High Bytes Register, Used in Receive Address Matching	Section 22.5.44
508h	MACINDEX	MAC Index Register	Section 22.5.45
600h	TX0HDP	Transmit Channel 0 DMA Head Descriptor Pointer Register	Section 22.5.46
604h	TX1HDP	Transmit Channel 1 DMA Head Descriptor Pointer Register	Section 22.5.46
608h	TX2HDP	Transmit Channel 2 DMA Head Descriptor Pointer Register	Section 22.5.46
60Ch	TX3HDP	Transmit Channel 3 DMA Head Descriptor Pointer Register	Section 22.5.46
610h	TX4HDP	Transmit Channel 4 DMA Head Descriptor Pointer Register	Section 22.5.46
614h	TX5HDP	Transmit Channel 5 DMA Head Descriptor Pointer Register	Section 22.5.46
618h	TX6HDP	Transmit Channel 6 DMA Head Descriptor Pointer Register	Section 22.5.46
61Ch	TX7HDP	Transmit Channel 7 DMA Head Descriptor Pointer Register	Section 22.5.46
620h	RX0HDP	Receive Channel 0 DMA Head Descriptor Pointer Register	Section 22.5.47
624h	RX1HDP	Receive Channel 1 DMA Head Descriptor Pointer Register	Section 22.5.47
628h	RX2HDP	Receive Channel 2 DMA Head Descriptor Pointer Register	Section 22.5.47
62Ch	RX3HDP	Receive Channel 3 DMA Head Descriptor Pointer Register	Section 22.5.47
630h	RX4HDP	Receive Channel 4 DMA Head Descriptor Pointer Register	Section 22.5.47
634h	RX5HDP	Receive Channel 5 DMA Head Descriptor Pointer Register	Section 22.5.47
638h	RX6HDP	Receive Channel 6 DMA Head Descriptor Pointer Register	Section 22.5.47
63Ch	RX7HDP	Receive Channel 7 DMA Head Descriptor Pointer Register	Section 22.5.47
640h	TX0CP	Transmit Channel 0 Completion Pointer Register	Section 22.5.48
644h	TX1CP	Transmit Channel 1 Completion Pointer Register	Section 22.5.48
648h	TX2CP	Transmit Channel 2 Completion Pointer Register	Section 22.5.48
64Ch	TX3CP	Transmit Channel 3 Completion Pointer Register	Section 22.5.48
650h	TX4CP	Transmit Channel 4 Completion Pointer Register	Section 22.5.48
654h	TX5CP	Transmit Channel 5 Completion Pointer Register	Section 22.5.48
658h	TX6CP	Transmit Channel 6 Completion Pointer Register	Section 22.5.48
65Ch	TX7CP	Transmit Channel 7 Completion Pointer Register	Section 22.5.48
660h	RX0CP	Receive Channel 0 Completion Pointer Register	Section 22.5.49
664h	RX1CP	Receive Channel 1 Completion Pointer Register	Section 22.5.49
668h	RX2CP	Receive Channel 2 Completion Pointer Register	Section 22.5.49
66Ch	RX3CP	Receive Channel 3 Completion Pointer Register	Section 22.5.49
670h	RX4CP	Receive Channel 4 Completion Pointer Register	Section 22.5.49
674h	RX5CP	Receive Channel 5 Completion Pointer Register	Section 22.5.49
678h	RX6CP	Receive Channel 6 Completion Pointer Register	Section 22.5.49
67Ch	RX7CP	Receive Channel 7 Completion Pointer Register	Section 22.5.49

Table 22-47. Ethernet Media Access Controller (EMAC) Registers (continued)

Offset	Acronym	Register Description	Section
300 - 3FCh	RX FIFO Processor Test Access	Only accessible when the MEMTEST bit in the MacControl register is set (32-bit access only).	
400 - 4FCh	TX FIFO Processor Test Access	Only accessible when the MEMTEST bit in the MacControl register is set (32-bit access only).	
700h - 77Ch	Stateram Test Access	Processor read and write access to head descriptor pointers and interrupt acknowledge registers.	
		Network Statistics Registers	
200h	RXGOODFRAMES	Good Receive Frames Register	Section 22.5.50.1
204h	RXBCASTFRAMES	Broadcast Receive Frames Register	Section 22.5.50.2
208h	RXMCASTFRAMES	Multicast Receive Frames Register	Section 22.5.50.3
20Ch	RXPAUSEFRAMES	Pause Receive Frames Register	Section 22.5.50.4
210h	RXCRCERRORS	Receive CRC Errors Register	Section 22.5.50.5
214h	RXALIGNCODEERRORS	Receive Alignment/Code Errors Register	Section 22.5.50.6
218h	RXOVERSIZED	Receive Oversized Frames Register	Section 22.5.50.7
21Ch	RXJABBER	Receive Jabber Frames Register	Section 22.5.50.8
220h	RXUNDERSIZED	Receive Undersized Frames Register	Section 22.5.50.9
224h	RXFRAGMENTS	Receive Frame Fragments Register	Section 22.5.50.10
228h	RXFILTERED	Filtered Receive Frames Register	Section 22.5.50.11
22Ch	RXQOSFILTERED	Receive QOS Filtered Frames Register	Section 22.5.50.12
230h	RXOCTETS	Receive Octet Frames Register	Section 22.5.50.13
234h	TXGOODFRAMES	Good Transmit Frames Register	Section 22.5.50.14
238h	TXBCASTFRAMES	Broadcast Transmit Frames Register	Section 22.5.50.15
23Ch	TXMCASTFRAMES	Multicast Transmit Frames Register	Section 22.5.50.16
240h	TXPAUSEFRAMES	Pause Transmit Frames Register	Section 22.5.50.17
244h	TXDEFERRED	Deferred Transmit Frames Register	Section 22.5.50.18
248h	TXCOLLISION	Transmit Collision Frames Register	Section 22.5.50.19
24Ch	TXSINGLECOLL	Transmit Single Collision Frames Register	Section 22.5.50.20
250h	TXMULTICOLL	Transmit Multiple Collision Frames Register	Section 22.5.50.21
254h	TXEXCESSIVECOLL	Transmit Excessive Collision Frames Register	Section 22.5.50.22
258h	TXLATECOLL	Transmit Late Collision Frames Register	Section 22.5.50.23
25Ch	TXUNDERRUN	Transmit Underrun Error Register	Section 22.5.50.24
260h	TXCARRIERSENSE	Transmit Carrier Sense Errors Register	Section 22.5.50.25
264h	TXOCTETS	Transmit Octet Frames Register	Section 22.5.50.26
268h	FRAME64	Transmit and Receive 64 Octet Frames Register	Section 22.5.50.27
26Ch	FRAME65T127	Transmit and Receive 65 to 127 Octet Frames Register	Section 22.5.50.28

Table 22-47. Ethernet Media Access Controller (EMAC) Registers (continued)

Offset	Acronym	Register Description	Section
270h	FRAME128T255	Transmit and Receive 128 to 255 Octet Frames Register	Section 22.5.50.29
274h	FRAME256T511	Transmit and Receive 256 to 511 Octet Frames Register	Section 22.5.50.30
278h	FRAME512T1023	Transmit and Receive 512 to 1023 Octet Frames Register	Section 22.5.50.31
27Ch	FRAME1024TUP	Transmit and Receive 1024 to RXMAXLEN Octet Frames Register	Section 22.5.50.32
280h	NETOCTETS	Network Octet Frames Register	Section 22.5.50.33
284h	RXSOFOVERRUNS	Receive FIFO or DMA Start of Frame Overruns Register	Section 22.5.50.34
288h	RXMOFOVERRUNS	Receive FIFO or DMA Middle of Frame Overruns Register	Section 22.5.50.35
28Ch	RXDMAOVERRUNS	Receive DMA Overruns Register	Section 22.5.50.36

22.5.1 Transmit Revision ID Register (TXREVID)

The Transmit Revision ID Register (TXREVID) is shown in Figure 22-37 and described in Table 22-48.

Figure 22-37. Transmit Revision ID Register (TXREVID)

31	16
TXREV	
R-4EC0h	

15	0
TXREV	
R-020Dh	

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

Table 22-48. Transmit Revision ID Register (TXREVID) Field Descriptions

Bit	Field	Value	Description
31-0	TXREV		Transmit module revision
		4EC0 020Dh	Current transmit revision value

22.5.2 Transmit Control Register (TXCONTROL)

The transmit control register (TXCONTROL) is shown in Figure 22-38 and described in Table 22-49.

Figure 22-38. Transmit Control Register (TXCONTROL)

31	16
Reserved	
R-0	

15	1	0
Reserved		TXEN
R-0		R/W-0

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

Table 22-49. Transmit Control Register (TXCONTROL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	TXEN		Transmit enable
		0	Transmit is disabled.
		1	Transmit is enabled.

22.5.3 Transmit Teardown Register (TXTEARDOWN)

The transmit teardown register (TXTEARDOWN) is shown in Figure 22-39 and described in Table 22-50.

Figure 22-39. Transmit Teardown Register (TXTEARDOWN)

31	30	16
TXTDNRDY	Reserved	
R-0	R-0	

15	3	2	0
Reserved		TXTDNCH	
R-0		R/W-0	

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

Table 22-50. Transmit Teardown Register (TXTEARDOWN) Field Descriptions

Bit	Field	Value	Description
31	TXTDNRDY	0	Tx Teardown Ready – read as zero, but is always assumed to be one (unused).
30-3	Reserved	0	Reserved
2-0	TXTDNCH	0-7h	Transmit teardown channel. The transmit channel teardown is commanded by writing the encoded value of the transmit channel to be torn down. The teardown register is read as 0.
		0	Teardown transmit channel 0
		1h	Teardown transmit channel 1
		2h	Teardown transmit channel 2
		3h	Teardown transmit channel 3
		4h	Teardown transmit channel 4
		5h	Teardown transmit channel 5
		6h	Teardown transmit channel 6
		7h	Teardown transmit channel 7

22.5.4 Receive Revision ID Register (RXREVID)

The Receive Revision ID Register (RXREVID) is shown in Figure 22-40 and described in Table 22-51.

Figure 22-40. Receive Revision ID Register (RXREVID)

31	16
RXREV	
R-4EC0h	

15	0
RXREV	
R-020Dh	

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

Table 22-51. Receive Revision ID Register (RXREVID) Field Descriptions

Bit	Field	Value	Description
31-0	RXREV		Receive module revision
		4EC0 020Dh	Current receive revision value

22.5.5 Receive Control Register (RXCONTROL)

The receive control register (RXCONTROL) is shown in Figure 22-41 and described in Table 22-52.

Figure 22-41. Receive Control Register (RXCONTROL)

31	16
Reserved	
R-0	

15	1	0
Reserved		RXEN
R-0		R/W-0

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

Table 22-52. Receive Control Register (RXCONTROL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RXEN		Receive enable
		0	Receive is disabled.
		1	Receive is enabled.

22.5.6 Receive Teardown Register (RXTEARDOWN)

The receive teardown register (RXTEARDOWN) is shown in Figure 22-42 and described in Table 22-53.

Figure 22-42. Receive Teardown Register (RXTEARDOWN)

31	30	16
RXTNRDY	Reserved	
R/W-0	R-0	

15	3	2	0
Reserved		RXTDNCH	
R-0		R/W-0	

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

Table 22-53. Receive Teardown Register (RXTEARDOWN) Field Descriptions

Bit	Field	Value	Description
31	RXTNRDY		Teardown Ready-read as zero, but is always assumed as one.
		0	
		1	
30-3	Reserved	0	Reserved
2-0	RXTDNCH	0-7h	Receive teardown channel. The receive channel teardown is commanded by writing the encoded value of the receive channel to be torn down. The teardown register is read as 0.
		0	Teardown receive channel 0
		1h	Teardown receive channel 1
		2h	Teardown receive channel 2
		3h	Teardown receive channel 3
		4h	Teardown receive channel 4
		5h	Teardown receive channel 5
		6h	Teardown receive channel 6
		7h	Teardown receive channel 7

22.5.7 Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)

The transmit interrupt status (unmasked) register (TXINTSTATRAW) is shown in Figure 22-43 and described in Table 22-54.

Figure 22-43. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
TX7PEND	TX6PEND	TX5PEND	TX4PEND	TX3PEND	TX2PEND	TX1PEND	TX0PEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

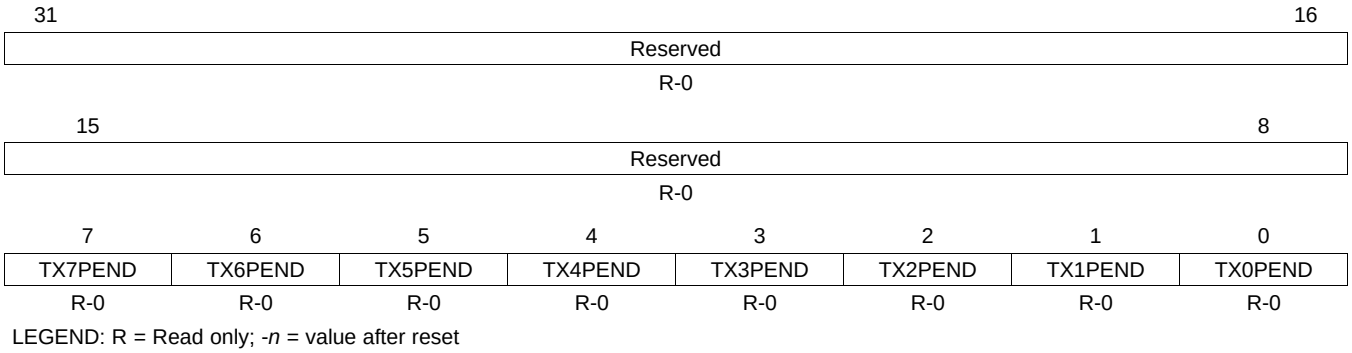
Table 22-54. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND raw interrupt read (before mask)
6	TX6PEND	0-1	TX6PEND raw interrupt read (before mask)
5	TX5PEND	0-1	TX5PEND raw interrupt read (before mask)
4	TX4PEND	0-1	TX4PEND raw interrupt read (before mask)
3	TX3PEND	0-1	TX3PEND raw interrupt read (before mask)
2	TX2PEND	0-1	TX2PEND raw interrupt read (before mask)
1	TX1PEND	0-1	TX1PEND raw interrupt read (before mask)
0	TX0PEND	0-1	TX0PEND raw interrupt read (before mask)

22.5.8 Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)

The transmit interrupt status (masked) register (TXINTSTATMASKED) is shown in Figure 22-44 and described in Table 22-55.

Figure 22-44. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)



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

Table 22-55. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND masked interrupt read
6	TX6PEND	0-1	TX6PEND masked interrupt read
5	TX5PEND	0-1	TX5PEND masked interrupt read
4	TX4PEND	0-1	TX4PEND masked interrupt read
3	TX3PEND	0-1	TX3PEND masked interrupt read
2	TX2PEND	0-1	TX2PEND masked interrupt read
1	TX1PEND	0-1	TX1PEND masked interrupt read
0	TX0PEND	0-1	TX0PEND masked interrupt read

22.5.9 Transmit Interrupt Mask Set Register (TXINTMASKSET)

The transmit interrupt mask set register (TXINTMASKSET) is shown in Figure 22-45 and described in Table 22-56.

Figure 22-45. Transmit Interrupt Mask Set Register (TXINTMASKSET)

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
TX7PULSEMASK	TX6PULSEMASK	TX5PULSEMASK	TX4PULSEMASK	TX3PULSEMASK	TX2PULSEMASK	TX1PULSEMASK	TX0PULSEMASK
R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0
15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
TX7PENDMASK	TX6PENDMASK	TX5PENDMASK	TX4PENDMASK	TX3PENDMASK	TX2PENDMASK	TX1PENDMASK	TX0PENDMASK
R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0

LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 22-56. Transmit Interrupt Mask Set Register (TXINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-24	RESERVED	0	Reserved
23	TX7PULSEMASK	0-1	Transmit channel 7 Pulse mask set bit. Write 1 to enable interrupt.
22	TX6PULSEMASK	0-1	Transmit channel 6 Pulse mask set bit. Write 1 to enable interrupt.
21	TX5PULSEMASK	0-1	Transmit channel 5 Pulse mask set bit. Write 1 to enable interrupt.
20	TX4PULSEMASK	0-1	Transmit channel 4 Pulse mask set bit. Write 1 to enable interrupt.
19	TX3PULSEMASK	0-1	Transmit channel 3 Pulse mask set bit. Write 1 to enable interrupt.
18	TX2PULSEMASK	0-1	Transmit channel 2 Pulse mask set bit. Write 1 to enable interrupt.
17	TX1PULSEMASK	0-1	Transmit channel 1 Pulse mask set bit. Write 1 to enable interrupt.
16	TX0PULSEMASK	0-1	Transmit channel 0 Pulse mask set bit. Write 1 to enable interrupt.
15-8	RESERVED	0	Reserved
7	TX7PENDMASK	0-1	Transmit channel 7 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
6	TX6PENDMASK	0-1	Transmit channel 6 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
5	TX5PENDMASK	0-1	Transmit channel 5 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
4	TX4PENDMASK	0-1	Transmit channel 4 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
3	TX3PENDMASK	0-1	Transmit channel 3 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
2	TX2PENDMASK	0-1	Transmit channel 2 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
1	TX1PENDMASK	0-1	Transmit channel 1 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	TX0PENDMASK	0-1	Transmit channel 0 pend interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

22.5.10 Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)

The transmit interrupt mask clear register (TXINTMASKCLEAR) is shown in Figure 22-46 and described in Table 22-57.

Figure 22-46. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
TX7PULSEMASK	TX6PULSEMASK	TX5PULSEMASK	TX4PULSEMASK	TX3PULSEMASK	TX2PULSEMASK	TX1PULSEMASK	TX0PULSEMASK
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0
15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
TX7PENDMASK	TX6PENDMASK	TX5PENDMASK	TX4PENDMASK	TX3PENDMASK	TX2PENDMASK	TX1PENDMASK	TX0PENDMASK
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 22-57. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-24	RESERVED	0	Reserved
23	TX7PULSEMASK	0-1	Transmit channel 7 Pulse mask set bit. Write 1 to disable interrupt.
22	TX6PULSEMASK	0-1	Transmit channel 6 Pulse mask set bit. Write 1 to disable interrupt.
21	TX5PULSEMASK	0-1	Transmit channel 5 Pulse mask set bit. Write 1 to disable interrupt.
20	TX4PULSEMASK	0-1	Transmit channel 4 Pulse mask set bit. Write 1 to disable interrupt.
19	TX3PULSEMASK	0-1	Transmit channel 3 Pulse mask set bit. Write 1 to disable interrupt.
18	TX2PULSEMASK	0-1	Transmit channel 2 Pulse mask set bit. Write 1 to disable interrupt.
17	TX1PULSEMASK	0-1	Transmit channel 1 Pulse mask set bit. Write 1 to disable interrupt.
16	TX0PULSEMASK	0-1	Transmit channel 0 Pulse mask set bit. Write 1 to disable interrupt.
15-8	RESERVED	0	Reserved
7	TX7PENDMASK	0-1	Transmit channel 7 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
6	TX6PENDMASK	0-1	Transmit channel 6 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
5	TX5PENDMASK	0-1	Transmit channel 5 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
4	TX4PENDMASK	0-1	Transmit channel 4 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
3	TX3PENDMASK	0-1	Transmit channel 3 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
2	TX2PENDMASK	0-1	Transmit channel 2 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
1	TX1PENDMASK	0-1	Transmit channel 1 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	TX0PENDMASK	0-1	Transmit channel 0 pend interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

22.5.11 MAC Input Vector Register (MACINVECTOR)

The MAC input vector register (MACINVECTOR) is shown in Figure 22-47 and described in Table 22-58.

Figure 22-47. MAC Input Vector Register (MACINVECTOR)

31	28	27	26	25	24	23	16
Reserved		STATPEND	HOSTPEND	LINKINT0	USERINT0	TXPEND	
R-0		R-0	R-0	R-0	R-0	R-0	
15					8	7	0
RXTHRESHPEND						RXPEND	
R-0						R-0	

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

Table 22-58. MAC Input Vector Register (MACINVECTOR) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	STATPEND	0-1	EMAC module statistics interrupt (STATPEND) pending status bit
26	HOSTPEND	0-1	EMAC module host error interrupt (HOSTPEND) pending status bit
25	LINKINT0	0-1	MDIO module USERPHYSEL0 (LINKINT0) status bit
24	USERINT0	0-1	MDIO module USERACCESS0 (USERINT0) status bit
23-16	TXPEND	0-FFh	Transmit channels 0-7 interrupt (TX*n*PEND) pending status. Bit 16 is TX0PEND.
15-8	RXTHRESHPEND	0-FFh	Receive channels 0-7 interrupt (RX*n*THRESHPEND) pending status. Bit 8 is RX0THRESHPEND.
7-0	RXPEND	0-FFh	Receive channels 0-7 interrupt (RX*n*PEND) pending status bit. Bit 0 is RX0PEND.

22.5.12 MAC End Of Interrupt Vector Register (MACEOIVECTOR)

The MAC end of interrupt vector register (MACEOIVECTOR) is shown in Figure 22-48 and described in Table 22-59.

Figure 22-48. MAC End Of Interrupt Vector Register (MACEOIVECTOR)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		INTVECT	
R-0		R/W-0	

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

Table 22-59. MAC End Of Interrupt Vector Register (MACEOIVECTOR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	INTVECT	0-1Fh	Acknowledge Interrupts
		0h	Acknowledge C0RXTHRESH Interrupt
		1h	Acknowledge C0RX Interrupt
		2h	Acknowledge C0TX Interrupt
		3h	Acknowledge C0MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		4h	Acknowledge C1RXTHRESH Interrupt
		5h	Acknowledge C1RX Interrupt
		6h	Acknowledge C1TX Interrupt
		7h	Acknowledge C1MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		8h	Acknowledge C2RXTHRESH Interrupt
		9h	Acknowledge C2RX Interrupt
		Ah	Acknowledge C2TX Interrupt
		Bh	Acknowledge C2MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		Ch-1Fh	Reserved

22.5.13 Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)

The receive interrupt status (unmasked) register (RXINTSTATRAW) is shown in Figure 22-49 and described in Table 22-60.

Figure 22-49. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)

31							16
Reserved							
R-0							
15	14	13	12	11	10	9	8
RX7THRESHPEND	RX6THRESHPEND	RX5THRESHPEND	RX4THRESHPEND	RX3THRESHPEND	RX2THRESHPEND	RX1THRESHPEND	RX0THRESHPEND
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
7	6	5	4	3	2	1	0
RX7PEND	RX6PEND	RX5PEND	RX4PEND	RX3PEND	RX2PEND	RX1PEND	RX0PEND
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 22-60. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHPEND	0-1	RX7THRESHPEND raw interrupt read (before mask)
14	RX6THRESHPEND	0-1	RX6THRESHPEND raw interrupt read (before mask)
13	RX5THRESHPEND	0-1	RX5THRESHPEND raw interrupt read (before mask)
12	RX4THRESHPEND	0-1	RX4THRESHPEND raw interrupt read (before mask)
11	RX3THRESHPEND	0-1	RX3THRESHPEND raw interrupt read (before mask)
10	RX2THRESHPEND	0-1	RX2THRESHPEND raw interrupt read (before mask)
9	RX1THRESHPEND	0-1	RX1THRESHPEND raw interrupt read (before mask)
8	RX0THRESHPEND	0-1	RX0THRESHPEND raw interrupt read (before mask)
7	RX7PEND	0-1	RX7PEND raw interrupt read (before mask)
6	RX6PEND	0-1	RX6PEND raw interrupt read (before mask)
5	RX5PEND	0-1	RX5PEND raw interrupt read (before mask)
4	RX4PEND	0-1	RX4PEND raw interrupt read (before mask)
3	RX3PEND	0-1	RX3PEND raw interrupt read (before mask)
2	RX2PEND	0-1	RX2PEND raw interrupt read (before mask)
1	RX1PEND	0-1	RX1PEND raw interrupt read (before mask)
0	RX0PEND	0-1	RX0PEND raw interrupt read (before mask)

22.5.14 Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)

The receive interrupt status (masked) register (RXINTSTATMASKED) is shown in Figure 22-50 and described in Table 22-61.

Figure 22-50. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
RX7THRESHPEND	RX6THRESHPEND	RX5THRESHPEND	RX4THRESHPEND	RX3THRESHPEND	RX2THRESHPEND	RX1THRESHPEND	RX0THRESHPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RX7PEND	RX6PEND	RX5PEND	RX4PEND	RX3PEND	RX2PEND	RX1PEND	RX0PEND
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 22-61. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHPEND	0-1	RX7THRESHPEND masked interrupt read
14	RX6THRESHPEND	0-1	RX6THRESHPEND masked interrupt read
13	RX5THRESHPEND	0-1	RX5THRESHPEND masked interrupt read
12	RX4THRESHPEND	0-1	RX4THRESHPEND masked interrupt read
11	RX3THRESHPEND	0-1	RX3THRESHPEND masked interrupt read
10	RX2THRESHPEND	0-1	RX2THRESHPEND masked interrupt read
9	RX1THRESHPEND	0-1	RX1THRESHPEND masked interrupt read
8	RX0THRESHPEND	0-1	RX0THRESHPEND masked interrupt read
7	RX7PEND	0-1	RX7PEND masked interrupt read
6	RX6PEND	0-1	RX6PEND masked interrupt read
5	RX5PEND	0-1	RX5PEND masked interrupt read
4	RX4PEND	0-1	RX4PEND masked interrupt read
3	RX3PEND	0-1	RX3PEND masked interrupt read
2	RX2PEND	0-1	RX2PEND masked interrupt read
1	RX1PEND	0-1	RX1PEND masked interrupt read
0	RX0PEND	0-1	RX0PEND masked interrupt read

22.5.15 Receive Interrupt Mask Set Register (RXINTMASKSET)

The receive interrupt mask set register (RXINTMASKSET) is shown in Figure 22-51 and described in Table 22-62.

Figure 22-51. Receive Interrupt Mask Set Register (RXINTMASKSET)

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
RX7PULSEMASK	RX6PULSEMASK	RX5PULSEMASK	RX4PULSEMASK	RX3PULSEMASK	RX2PULSEMASK	RX1PULSEMASK	RX0PULSEMASK
R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0
15	14	13	12	11	10	9	8
RX7PENDTHRESHMASK	RX6PENDTHRESHMASK	RX5PENDTHRESHMASK	RX4PENDTHRESHMASK	RX3PENDTHRESHMASK	RX2PENDTHRESHMASK	RX1PENDTHRESHMASK	RX0PENDTHRESHMASK
R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0
7	6	5	4	3	2	1	0
RX7PENDMASK	RX6PENDMASK	RX5PENDMASK	RX4PENDMASK	RX3PENDMASK	RX2PENDMASK	RX1PENDMASK	RX0PENDMASK
R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0

LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 22-62. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23	RX7PULSEMASK	0-1	Receive channel 7 Pulse mask set bit. Write 1 to enable interrupt.
22	RX6PULSEMASK	0-1	Receive channel 6 Pulse mask set bit. Write 1 to enable interrupt.
21	RX5PULSEMASK	0-1	Receive channel 5 Pulse mask set bit. Write 1 to enable interrupt.
20	RX4PULSEMASK	0-1	Receive channel 4 Pulse mask set bit. Write 1 to enable interrupt.
19	RX3PULSEMASK	0-1	Receive channel 3 Pulse mask set bit. Write 1 to enable interrupt.
18	RX2PULSEMASK	0-1	Receive channel 2 Pulse mask set bit. Write 1 to enable interrupt.
17	RX1PULSEMASK	0-1	Receive channel 1 Pulse mask set bit. Write 1 to enable interrupt.
16	RX0PULSEMASK	0-1	Receive channel 0 Pulse mask set bit. Write 1 to enable interrupt.
15	RX7PENDTHRESHMASK	0-1	Receive channel 7 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
14	RX6PENDTHRESHMASK	0-1	Receive channel 6 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
13	RX5PENDTHRESHMASK	0-1	Receive channel 5 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
12	RX4PENDTHRESHMASK	0-1	Receive channel 4 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
11	RX3PENDTHRESHMASK	0-1	Receive channel 3 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
10	RX2PENDTHRESHMASK	0-1	Receive channel 2 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
9	RX1PENDTHRESHMASK	0-1	Receive channel 1 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
8	RX0PENDTHRESHMASK	0-1	Receive channel 0 pend threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
7	RX7PENDMASK	0-1	Receive channel 7 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
6	RX6PENDMASK	0-1	Receive channel 6 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
5	RX5PENDMASK	0-1	Receive channel 5 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
4	RX4PENDMASK	0-1	Receive channel 4 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
3	RX3PENDMASK	0-1	Receive channel 3 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.

Table 22-62. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions (continued)

Bit	Field	Value	Description
2	RX2PENDMASK	0-1	Receive channel 2 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
1	RX1PENDMASK	0-1	Receive channel 1 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
0	RX0PENDMASK	0-1	Receive channel 0 pendmask set bit. Write 1 to enable interrupt; a write of 0 has no effect.

22.5.16 Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)

The receive interrupt mask clear register (RXINTMASKCLEAR) is shown in Figure 22-52 and described in Table 22-63.

Figure 22-52. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
RX7PULSEMASK	RX6PULSEMASK	RX5PULSEMASK	RX4PULSEMASK	RX3PULSEMASK	RX2PULSEMASK	RX1PULSEMASK	RX0PULSEMASK
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0
15	14	13	12	11	10	9	8
RX7PENDTHRESHMASK	RX6PENDTHRESHMASK	RX5PENDTHRESHMASK	RX4PENDTHRESHMASK	RX3PENDTHRESHMASK	RX2PENDTHRESHMASK	RX1PENDTHRESHMASK	RX0PENDTHRESHMASK
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0
7	6	5	4	3	2	1	0
RX7PENDMASK	RX6PENDMASK	RX5PENDMASK	RX4PENDMASK	RX3PENDMASK	RX2PENDMASK	RX1PENDMASK	RX0PENDMASK
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 22-63. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved.
23	RX7PULSEMASK	0-1	Receive channel 7 pulse mask set bit. Write 1 to disable interrupt.
22	RX6PULSEMASK	0-1	Receive channel 6 pulse mask set bit. Write 1 to disable interrupt.
21	RX5PULSEMASK	0-1	Receive channel 5 pulse mask set bit. Write 1 to disable interrupt.
20	RX4PULSEMASK	0-1	Receive channel 4 pulse mask set bit. Write 1 to disable interrupt.
19	RX3PULSEMASK	0-1	Receive channel 3 pulse mask set bit. Write 1 to disable interrupt.
18	RX2PULSEMASK	0-1	Receive channel 2 pulse mask set bit. Write 1 to disable interrupt.
17	RX1PULSEMASK	0-1	Receive channel 1 pulse mask set bit. Write 1 to disable interrupt.
16	RX0PULSEMASK	0-1	Receive channel 0 pulse mask set bit. Write 1 to disable interrupt.
15	RX7PENDTHRESHMASK	0-1	Receive channel 7 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
14	RX6PENDTHRESHMASK	0-1	Receive channel 6 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
13	RX5PENDTHRESHMASK	0-1	Receive channel 5 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
12	RX4PENDTHRESHMASK	0-1	Receive channel 4 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
11	RX3PENDTHRESHMASK	0-1	Receive channel 3 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
10	RX2PENDTHRESHMASK	0-1	Receive channel 2 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
9	RX1PENDTHRESHMASK	0-1	Receive channel 1 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.

Table 22-63. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions (continued)

Bit	Field	Value	Description
8	RX0PENDTHRESHMASK	0-1	Receive channel 0 pend threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
7	RX7PENDMASK	0-1	Receive channel 7 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
6	RX6PENDMASK	0-1	Receive channel 6 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
5	RX5PENDMASK	0-1	Receive channel 5 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
4	RX4PENDMASK	0-1	Receive channel 4 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
3	RX3PENDMASK	0-1	Receive channel 3 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
2	RX2PENDMASK	0-1	Receive channel 2 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
1	RX1PENDMASK	0-1	Receive channel 1 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
0	RX0PENDMASK	0-1	Receive channel 0 pendmask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.

22.5.17 MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)

The MAC interrupt status (unmasked) register (MACINTSTATRAW) is shown in Figure 22-53 and described in Table 22-64.

Figure 22-53. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTPEND	STATPEND
R-0		R-0	R-0

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

Table 22-64. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); raw interrupt read (before mask).
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); raw interrupt read (before mask).

22.5.18 MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

The MAC interrupt status (masked) register (MACINTSTATMASKED) is shown in Figure 22-54 and described in Table 22-65.

Figure 22-54. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTPEND	STATPEND
R-0		R-0	R-0

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

Table 22-65. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); masked interrupt read.
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); masked interrupt read.

22.5.19 MAC Interrupt Mask Set Register (MACINTMASKSET)

The MAC interrupt mask set register (MACINTMASKSET) is shown in Figure 22-55 and described in Table 22-66.

Figure 22-55. MAC Interrupt Mask Set Register (MACINTMASKSET)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTMASK	STATMASK
R-0		R/WS-0	R/WS-0

LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 22-66. MAC Interrupt Mask Set Register (MACINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

22.5.20 MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)

The MAC interrupt mask clear register (MACINTMASKCLEAR) is shown in Figure 22-56 and described in Table 22-67.

Figure 22-56. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTMASK	STATMASK
R-0		R/WC-0	R/WC-0

LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

Table 22-67. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

22.5.21 Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)

The receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is shown in Figure 22-57 and described in Table 22-68.

Figure 22-57. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)

31	30	29	28	27	25	24
Reserved	RXPASSCRC	RXQOSEN	RXNOCHAIN	Reserved		RXCMFEN
R-0	R/W-0	R/W-0	R/W-0	R-0		R/W-0

23	22	21	20	19	18	16
RXCSFEN	RXCEFEN	RXCAFEN	Reserved		RXPROMCH	
R/W-0	R/W-0	R/W-0	R-0		R/W-0	

15	14	13	12	11	10	8
Reserved		RXBROADEN	Reserved		RXBROADCH	
R-0		R/W-0	R-0		R/W-0	

7	6	5	4	3	2	0
Reserved		RXMULTEN	Reserved		RXMULTCH	
R-0		R/W-0	R-0		R/W-0	

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

Table 22-68. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	RXPASSCRC		Pass receive CRC enable bit
		0	Received CRC is discarded for all channels and is not included in the buffer descriptor packet length field.
		1	Received CRC is transferred to memory for all channels and is included in the buffer descriptor packet length.
29	RXQOSEN		Receive quality of service enable bit
		0	Receive QOS is disabled.
		1	Receive QOS is enabled.
28	RXNOCHAIN		Receive no buffer chaining bit
		0	Received frames can span multiple buffers.
		1	The Receive DMA controller transfers each frame into a single buffer, regardless of the frame or buffer size. All remaining frame data after the first buffer is discarded. The buffer descriptor buffer length field will contain the entire frame byte count (up to 65535 bytes).
27-25	Reserved	0	Reserved
24	RXCMFEN		Receive copy MAC control frames enable bit. Enables MAC control frames to be transferred to memory. MAC control frames are normally acted upon (if enabled), but not copied to memory. MAC control frames that are pause frames will be acted upon if enabled in MACCONTROL, regardless of the value of RXCMFEN. Frames transferred to memory due to RXCMFEN will have the CONTROL bit set in their EOP buffer descriptor.
		0	MAC control frames are filtered (but acted upon if enabled).
		1	MAC control frames are transferred to memory.
23	RXCSFEN		Receive copy short frames enable bit. Enables frames or fragments shorter than 64 bytes to be copied to memory. Frames transferred to memory due to RXCSFEN will have the FRAGMENT or UNDERSIZE bit set in their EOP buffer descriptor. Fragments are short frames that contain CRC / align / code errors and undersized are short frames without errors.
		0	Short frames are filtered.
		1	Short frames are transferred to memory.

Table 22-68. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
22	RXCEFEN		Receive copy error frames enable bit. Enables frames containing errors to be transferred to memory. The appropriate error bit will be set in the frame EOP buffer descriptor.
		0	Frames containing errors are filtered.
		1	Frames containing errors are transferred to memory.
21	RXCAFEN		Receive copy all frames enable bit. Enables frames that do not address match (includes multicast frames that do not hash match) to be transferred to the promiscuous channel selected by RXPROMCH bits. Such frames will be marked with the NOMATCH bit in their EOP buffer descriptor.
		0	Frames that do not address match are filtered.
		1	Frames that do not address match are transferred to the promiscuous channel selected by RXPROMCH bits.
20-19	Reserved	0	Reserved
18-16	RXPROMCH	0-7h	Receive promiscuous channel select
		0	Select channel 0 to receive promiscuous frames
		1h	Select channel 1 to receive promiscuous frames
		2h	Select channel 2 to receive promiscuous frames
		3h	Select channel 3 to receive promiscuous frames
		4h	Select channel 4 to receive promiscuous frames
		5h	Select channel 5 to receive promiscuous frames
		6h	Select channel 6 to receive promiscuous frames
		7h	Select channel 7 to receive promiscuous frames
15-14	Reserved	0	Reserved
13	RXBROADEN		Receive broadcast enable. Enable received broadcast frames to be copied to the channel selected by RXBROADCH bits.
		0	Broadcast frames are filtered.
		1	Broadcast frames are copied to the channel selected by RXBROADCH bits.
12-11	Reserved	0	Reserved
10-8	RXBROADCH	0-7h	Receive broadcast channel select
		0	Select channel 0 to receive broadcast frames
		1h	Select channel 1 to receive broadcast frames
		2h	Select channel 2 to receive broadcast frames
		3h	Select channel 3 to receive broadcast frames
		4h	Select channel 4 to receive broadcast frames
		5h	Select channel 5 to receive broadcast frames
		6h	Select channel 6 to receive broadcast frames
		7h	Select channel 7 to receive broadcast frames
7-6	Reserved	0	Reserved
5	RXMULTEN		RX multicast enable. Enable received hash matching multicast frames to be copied to the channel selected by RXMULTCH bits.
		0	Multicast frames are filtered.
		1	Multicast frames are copied to the channel selected by RXMULTCH bits.
4-3	Reserved	0	Reserved

Table 22-68. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	RXMULTCH	0-7h	Receive multicast channel select
		0	Select channel 0 to receive multicast frames
		1h	Select channel 1 to receive multicast frames
		2h	Select channel 2 to receive multicast frames
		3h	Select channel 3 to receive multicast frames
		4h	Select channel 4 to receive multicast frames
		5h	Select channel 5 to receive multicast frames
		6h	Select channel 6 to receive multicast frames
		7h	Select channel 7 to receive multicast frames

22.5.22 Receive Unicast Enable Set Register (RXUNICASTSET)

The receive unicast enable set register (RXUNICASTSET) is shown in Figure 22-58 and described in Table 22-69.

Figure 22-58. Receive Unicast Enable Set Register (RXUNICASTSET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0	R/WS-0

LEGEND: R = Read only; R/W = Read/Write; WS = Write 1 to set, write of 0 has no effect; -*n* = value after reset

Table 22-69. Receive Unicast Enable Set Register (RXUNICASTSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
6	RXCH6EN	0-1	Receive channel 6 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
5	RXCH5EN	0-1	Receive channel 5 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
4	RXCH4EN	0-1	Receive channel 4 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
3	RXCH3EN	0-1	Receive channel 3 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
2	RXCH2EN	0-1	Receive channel 2 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
1	RXCH1EN	0-1	Receive channel 1 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
0	RXCH0EN	0-1	Receive channel 0 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.

22.5.23 Receive Unicast Clear Register (RXUNICASTCLEAR)

The receive unicast clear register (RXUNICASTCLEAR) is shown in Figure 22-59 and described in Table 22-70.

Figure 22-59. Receive Unicast Clear Register (RXUNICASTCLEAR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R-0															

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

LEGEND: R = Read only; R/W = Read/Write; WC = Write 1 to clear, write of 0 has no effect; -*n* = value after reset

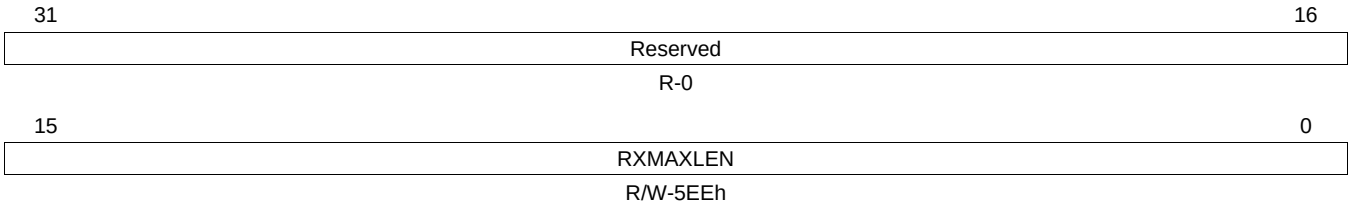
Table 22-70. Receive Unicast Clear Register (RXUNICASTCLEAR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
6	RXCH6EN	0-1	Receive channel 6 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
5	RXCH5EN	0-1	Receive channel 5 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
4	RXCH4EN	0-1	Receive channel 4 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
3	RXCH3EN	0-1	Receive channel 3 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
2	RXCH2EN	0-1	Receive channel 2 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
1	RXCH1EN	0-1	Receive channel 1 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
0	RXCH0EN	0-1	Receive channel 0 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.

22.5.24 Receive Maximum Length Register (RXMAXLEN)

The receive maximum length register (RXMAXLEN) is shown in Figure 22-60 and described in Table 22-71.

Figure 22-60. Receive Maximum Length Register (RXMAXLEN)



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

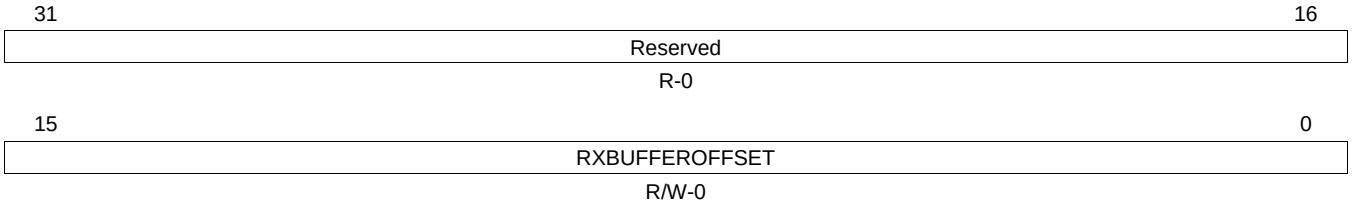
Table 22-71. Receive Maximum Length Register (RXMAXLEN) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXMAXLEN	0-FFFFh	Receive maximum frame length. These bits determine the maximum length of a received frame. The reset value is 5EEh (1518). Frames with byte counts greater than RXMAXLEN are long frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment error are jabber frames.

22.5.25 Receive Buffer Offset Register (RXBUFFEROFFSET)

The receive buffer offset register (RXBUFFEROFFSET) is shown in Figure 22-61 and described in Table 22-72.

Figure 22-61. Receive Buffer Offset Register (RXBUFFEROFFSET)



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

Table 22-72. Receive Buffer Offset Register (RXBUFFEROFFSET) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXBUFFEROFFSET	0-FFFFh	Receive buffer offset value. These bits are written by the EMAC into each frame SOP buffer descriptor Buffer Offset field. The frame data begins after the RXBUFFEROFFSET value of bytes. A value of 0 indicates that there are no unused bytes at the beginning of the data, and that valid data begins on the first byte of the buffer. A value of Fh (15) indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. This value is used for all channels.

22.5.26 Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) is shown in Figure 22-62 and described in Table 22-73.

Figure 22-62. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RXFILTERTHRESH	
R-0		R/W-0	

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

Table 22-73. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RXFILTERTHRESH	0-FFh	Receive filter low threshold. These bits contain the free buffer count threshold value for filtering low priority incoming frames. This field should remain 0, if no filtering is desired.

22.5.27 Receive Channel Flow Control Threshold Registers (RX0FLOWTHRESH-RX7FLOWTHRESH)

The receive channel 0-7 flow control threshold register (RX*n*FLOWTHRESH) is shown in Figure 22-63 and described in Table 22-74.

Figure 22-63. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RX*n*FLOWTHRESH	
R-0		R/W-0	

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

Table 22-74. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RX*n*FLOWTHRESH	0-FFh	Receive flow threshold. These bits contain the threshold value for issuing flow control on incoming frames for channel *n* (when enabled).

22.5.28 Receive Channel Free Buffer Count Registers (RX0FREEBUFFER-RX7FREEBUFFER)

The receive channel 0-7 free buffer count register (RX*n*FREEBUFFER) is shown in Figure 22-64 and described in Table 22-75.

Figure 22-64. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER)

31	16
Reserved	
R-0	

15	0
RX*n*FREEBUF	
WI-0	

LEGEND: R = Read only; WI = Write to increment; -*n* = value after reset

Table 22-75. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RX*n*FREEBUF	0-FFh	Receive free buffer count. These bits contain the count of free buffers available. The RXFILTERTHRESH value is compared with this field to determine if low priority frames should be filtered. The RX*n*FLOWTHRESH value is compared with this field to determine if receive flow control should be issued against incoming packets (if enabled). This is a write-to-increment field. This field rolls over to 0 on overflow. If hardware flow control or QOS is used, the host must initialize this field to the number of available buffers (one register per channel). The EMAC decrements the associated channel register for each received frame by the number of buffers in the received frame. The host must write this field with the number of buffers that have been freed due to host processing.

22.5.29 MAC Control Register (MACCONTROL)

The MAC control register (MACCONTROL) is shown in Figure 22-65 and described in Table 22-76.

Figure 22-65. MAC Control Register (MACCONTROL)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved					MACEXTEN	GIGFORCE	MACIFCTL_B
R-0					R/W-0	R/W-0	R/W-0

15	14	13	12	11	10	9	8
MACIFCTL_A	RXOFFLENBLOCK	RXOWNERSHIP	Reserved	CMDIDLE	TXSHORTGAPEN	TXPTYPE	MEMTEST
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
Reserved	TXPACE	GMIIEN	TXFLOWEN	RXBUFFERFLOWEN	Reserved	LOOPBACK	FULLDUPLEX
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

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

Table 22-76. MAC Control Register (MACCONTROL) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved.
18	MACEXTEN	0-1	External Enable. Enables the full duplex and gigabit mode to be selected from the EXTFULLDUPLEXSEL and EXTGIGSEL input signals and not from the full duplex and gig bits contained in this register. This register bit is also output on the MAC_EXT_EN signal.
17	GIGFORCE	0-1	Gigabit Mode Force. This bit is used to force the EMAC into gigabit mode if the input GMII_MTCLK has been stopped by the PHY. For RMII interface the value should be zero.
16	MACIFCTL_B	0-1	Interface Control B. Intended as a general purpose output bit to be used to control external gaskets associated with the GMII (GMII to RGMII etc). For RMII interface this bit is unused.
15	MACIFCTL_A		Interface Control A – Intended as a general purpose output bit to be used to control external gaskets associated with the GMII (GMII to RGMII etc). For GMII interface this bit is unused. For RMII Interface, it is used to select transmit and receive speed.
		0	Operate RMII interface in 10 Mbps speed mode.
		1	Operate RMII interface in 100 Mbps speed mode.
14	RXOFFLENBLOCK		Receive offset / length word write block.
		0	Do not block the DMA writes to the receive buffer descriptor offset / buffer length word.
		1	Block all EMAC DMA controller receive buffer descriptor offset / buffer length words during packet processing. When this bit is set, the EMAC will never write the third word to any receive buffer descriptor.
13	RXOWNERSHIP		Receive ownership write bit value.
		0	The EMAC writes the Receive ownership bit to 0 at the end of packet processing.
		1	The EMAC writes the Receive ownership bit to 1 at the end of packet processing. If you do not use the ownership mechanism, you can set this mode to preclude the necessity of software having to set this bit each time the buffer descriptor is used.
12	Reserved	0	Reserved
11	CMDIDLE		Command Idle bit
		0	Idle is not commanded.
		1	Idle is commanded (read IDLE in the MACSTATUS register).
10	TXSHORTGAPEN		Transmit Short Gap Enable
		0	Transmit with a short IPG is disabled. Normal 96-bit time IPG is inserted between packets.
		1	Transmit with a short IPG is enabled. Shorter 88-bit time IPG is inserted between packets.

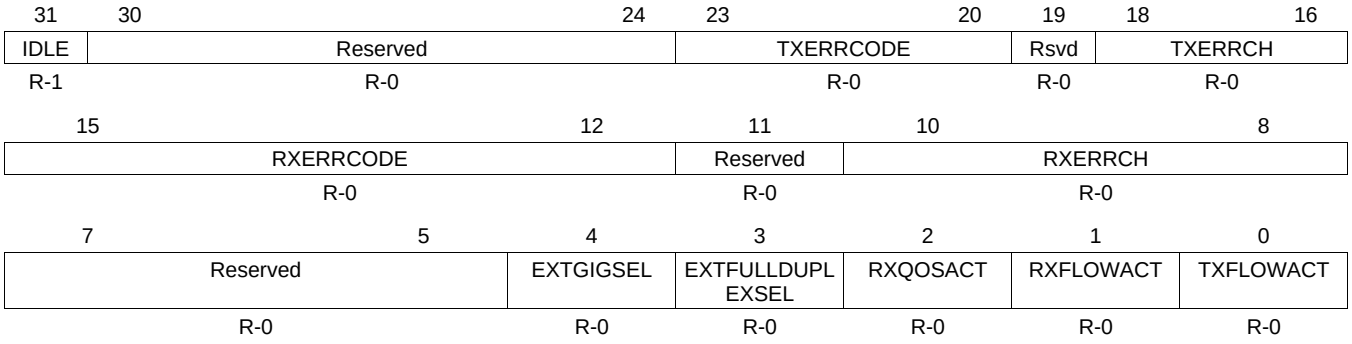
Table 22-76. MAC Control Register (MACCONTROL) Field Descriptions (continued)

Bit	Field	Value	Description
9	TXPTYPE		Transmit queue priority type
		0	The queue uses a round-robin scheme to select the next channel for transmission.
		1	The queue uses a fixed-priority (channel 7 highest priority) scheme to select the next channel for transmission.
8	MEMTEST		FIFO RAM processor read/write enable (ram test mode)
		0	No host access to RX/TX FIFO RAM.
		1	Host read/write access to RX/TX FIFO RAMS.
7	Reserved	0	Reserved
6	TXPACE		Transmit pacing enable bit
		0	Transmit pacing is disabled.
		1	Transmit pacing is enabled.
5	GMIIEN		GMII enable bit
		0	GMII RX and TX are held in reset.
		1	GMII RX and TX are enabled for receive and transmit.
4	TXFLOWEN		Transmit flow control enable bit. This bit determines if incoming pause frames are acted upon in full-duplex mode. Incoming pause frames are not acted upon in half-duplex mode, regardless of this bit setting. The RXMBPENABLE bits determine whether or not received pause frames are transferred to memory.
		0	Transmit flow control is disabled. Full-duplex mode: incoming pause frames are not acted upon.
		1	Transmit flow control is enabled. Full-duplex mode: incoming pause frames are acted upon.
3	RXBUFFERFLOWEN		Receive buffer flow control enable bit
		0	Receive flow control is disabled. Half-duplex mode: no flow control generated collisions are sent. Full-duplex mode: no outgoing pause frames are sent.
		1	Receive flow control is enabled. Half-duplex mode: collisions are initiated when receive buffer flow control is triggered. Full-duplex mode: outgoing pause frames are sent when receive flow control is triggered.
2	Reserved	0	Reserved
1	LOOPBACK		Loopback mode. The loopback mode forces internal full-duplex mode regardless of the FULLDUPLEX bit. The loopback bit should be changed only when GMIIEN bit is deasserted.
		0	Loopback mode is disabled.
		1	Loopback mode is enabled.
0	FULLDUPLEX		Full duplex mode.
		0	Half-duplex mode is enabled.
		1	Full-duplex mode is enabled.

22.5.30 MAC Status Register (MACSTATUS)

The MAC status register (MACSTATUS) is shown in Figure 22-66 and described in Table 22-77.

Figure 22-66. MAC Status Register (MACSTATUS)



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

Table 22-77. MAC Status Register (MACSTATUS) Field Descriptions

Bit	Field	Value	Description
31	IDLE		EMAC idle bit. This bit is cleared to 0 at reset; one clock after reset, it goes to 1.
		0	The EMAC is not idle.
		1	The EMAC is in the idle state.
30-24	Reserved	0	Reserved
23-20	TXERRCODE	0-Fh	Transmit host error code. These bits indicate that EMAC detected transmit DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover. A 0 packet length is an error, but it is not detected.
		0	No error
		1h	SOP error; the buffer is the first buffer in a packet, but the SOP bit is not set in software.
		2h	Ownership bit not set in SOP buffer
		3h	Zero next buffer descriptor pointer without EOP
		4h	Zero buffer pointer
		5h	Zero buffer length
		6h	Packet length error (sum of buffers is less than packet length)
		7h-Fh	Reserved
19	Reserved	0	Reserved
18-16	TXERRCH	0-7h	Transmit host error channel. These bits indicate which transmit channel the host error occurred on. This field is cleared to 0 on a host read.
		0	The host error occurred on transmit channel 0
		1h	The host error occurred on transmit channel 1
		2h	The host error occurred on transmit channel 2
		3h	The host error occurred on transmit channel 3
		4h	The host error occurred on transmit channel 4
		5h	The host error occurred on transmit channel 5
		6h	The host error occurred on transmit channel 6
		7h	The host error occurred on transmit channel 7

Table 22-77. MAC Status Register (MACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
15-12	RXERRCODE	0-Fh	Receive host error code. These bits indicate that EMAC detected receive DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover.
		0	No error
		1h	Reserved
		2h	Ownership bit not set in SOP buffer
		3h	Reserved
		4h	Zero buffer pointer
		5h-Fh	Reserved
11	Reserved	0	Reserved
10-8	RXERRCH	0-7h	Receive host error channel. These bits indicate which receive channel the host error occurred on. This field is cleared to 0 on a host read.
		0	The host error occurred on receive channel 0
		1h	The host error occurred on receive channel 1
		2h	The host error occurred on receive channel 2
		3h	The host error occurred on receive channel 3
		4h	The host error occurred on receive channel 4
		5h	The host error occurred on receive channel 5
		6h	The host error occurred on receive channel 6
		7h	The host error occurred on receive channel 7
7-5	Reserved	0	Reserved
4	EXTGIGSEL	0-1	External GIG – This is the value of the EXTGIGSEL input.
3	EXTFULLDUPLEXSEL	0-1	External Fullduplex – This is the value of the EXTFULLDUPLEXSEL input.
2	RXQOSACT		Receive Quality of Service (QOS) active bit. When asserted, indicates that receive quality of service is enabled and that at least one channel freebuffer count (RX*n*FREEBUFFER) is less than or equal to the RXFILTERLOWTHRESH value.
		0	Receive quality of service is disabled.
		1	Receive quality of service is enabled.
1	RXFLOWACT		Receive flow control active bit. When asserted, at least one channel freebuffer count (RX*n*FREEBUFFER) is less than or equal to the channel's corresponding RX*n*FILTERTHRESH value.
		0	Receive flow control is inactive.
		1	Receive flow control is active.
0	TXFLOWACT		Transmit flow control active bit. When asserted, this bit indicates that the pause time period is being observed for a received pause frame. No new transmissions will begin while this bit is asserted, except for the transmission of pause frames. Any transmission in progress when this bit is asserted will complete.
		0	Transmit flow control is inactive.
		1	Transmit flow control is active.

22.5.31 Emulation Control Register (EMCONTROL)

The emulation control register (EMCONTROL) is shown in Figure 22-67 and described in Table 22-78.

Figure 22-67. Emulation Control Register (EMCONTROL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

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

Table 22-78. Emulation Control Register (EMCONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Emulation soft bit. This bit is used in conjunction with FREE bit to determine the emulation suspend mode. This bit has no effect if FREE = 1.
		0	Soft mode is disabled. EMAC stops immediately during emulation halt.
		1	Soft mode is enabled. During emulation halt, EMAC stops after completion of current operation.
0	FREE		Emulation free bit. This bit is used in conjunction with SOFT bit to determine the emulation suspend mode.
		0	Free-running mode is disabled. During emulation halt, SOFT bit determines operation of EMAC.
		1	Free-running mode is enabled. During emulation halt, EMAC continues to operate.

22.5.32 FIFO Control Register (FIFOCONTROL)

The FIFO control register (FIFOCONTROL) is shown in Figure 22-68 and described in Table 22-79.

Figure 22-68. FIFO Control Register (FIFOCONTROL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		TXCELLTHRESH	
R-0		R/W-2h	

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

Table 22-79. FIFO Control Register (FIFOCONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	TXCELLTHRESH	0-3h	Transmit FIFO cell threshold. Indicates the number of 64-byte packet cells required to be in the transmit FIFO before the packet transfer is initiated. Packets with fewer cells will be initiated when the complete packet is contained in the FIFO. The default value is 2, but 3 is also valid. 0 and 1 are not valid values.
		0-1h	Not a valid value.
		2h	Two 64-byte packet cells required to be in the transmit FIFO.
		3h	Three 64-byte packet cells required to be in the transmit FIFO.

22.5.33 MAC Configuration Register (MACCONFIG)

The MAC configuration register (MACCONFIG) is shown in Figure 22-69 and described in Table 22-80.

Figure 22-69. MAC Configuration Register (MACCONFIG)

31	24	23	16
TXCELLDEPTH		RXCELLDEPTH	
R-3h		R-3h	

15	8	7	0
ADDRESSTYPE		MACCFIG	
R-2h		R-2h	

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

Table 22-80. MAC Configuration Register (MACCONFIG) Field Descriptions

Bit	Field	Value	Description
31-24	TXCELLDEPTH	3h	Transmit cell depth. These bits indicate the number of cells in the transmit FIFO.
23-16	RXCELLDEPTH	3h	Receive cell depth. These bits indicate the number of cells in the receive FIFO.
15-8	ADDRESSTYPE	2h	Address type
7-0	MACCFIG	2h	MAC configuration value

22.5.34 Soft Reset Register (SOFTRESET)

The soft reset register (SOFTRESET) is shown in Figure 22-70 and described in Table 22-81.

Figure 22-70. Soft Reset Register (SOFTRESET)

31	16
Reserved	
R-0	

15	1	0
Reserved		SOFTRESET
R-0		R/W-0

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

Table 22-81. Soft Reset Register (SOFTRESET) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SOFTRESET		Software reset. Writing a 1 to this bit causes the EMAC logic to be reset. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the Configuration bus. After writing a 1 to this bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet complete. If a 0 is read, then a reset has complete.
		0	Software reset complete status - read.
		1	Software reset is applied - write.

22.5.35 MAC Source Address Low Bytes Register (MACSRCADDRLO)

The MAC source address low bytes register (MACSRCADDRLO) is shown in Figure 22-71 and described in Table 22-82.

Figure 22-71. MAC Source Address Low Bytes Register (MACSRCADDRLO)

31	16
Reserved	
R-0	

15	8	7	0
MACSRCADDR0		MACSRCADDR1	
R/W-0		R/W-0	

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

Table 22-82. MAC Source Address Low Bytes Register (MACSRCADDRLO) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	MACSRCADDR0	0-FFh	MAC source address lower 7-0 bits (byte 0)
7-0	MACSRCADDR1	0-FFh	MAC source address bits 15-8 (byte 1)

22.5.36 MAC Source Address High Bytes Register (MACSRCADDRHI)

The MAC source address high bytes register (MACSRCADDRHI) is shown in Figure 22-72 and described in Table 22-83.

Figure 22-72. MAC Source Address High Bytes Register (MACSRCADDRHI)

31	24	23	16
MACSRCADDR2		MACSRCADDR3	
R/W-0		R/W-0	

15	8	7	0
MACSRCADDR4		MACSRCADDR5	
R/W-0		R/W-0	

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

Table 22-83. MAC Source Address High Bytes Register (MACSRCADDRHI) Field Descriptions

Bit	Field	Value	Description
31-24	MACSRCADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACSRCADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACSRCADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACSRCADDR5	0-FFh	MAC source address bits 47-40 (byte 5)

22.5.37 MAC Hash Address Register 1 (MACHASH1)

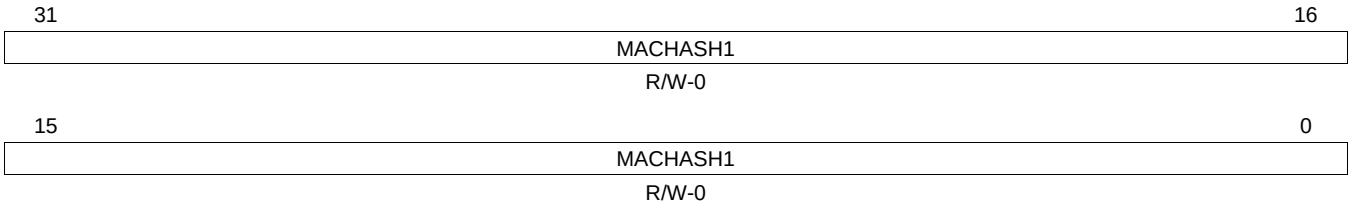
The MAC hash registers allow group addressed frames to be accepted on the basis of a hash function of the address. The hash function creates a 6-bit data value (Hash_fun) from the 48-bit destination address (DA) as follows:

```
Hash_fun(0)=DA(0) XOR DA(6) XOR DA(12) XOR DA(18) XOR DA(24) XOR DA(30) XOR DA(36) XOR DA(42);

Hash_fun(1)=DA(1) XOR DA(7) XOR DA(13) XOR DA(19) XOR DA(25) XOR DA(31) XOR DA(37) XOR DA(43);

Hash_fun(2)=DA(2) XOR DA(8) XOR DA(14) XOR DA(20) XOR DA(26) XOR DA(32) XOR DA(38) XOR DA(44);

Hash_fun(3)=DA(3) XOR DA(9) XOR DA(15) XOR DA(21) XOR DA(27) XOR DA(33) XOR DA(39) XOR DA(45);

Hash_fun(4)=DA(4) XOR DA(10) XOR DA(16) XOR DA(22) XOR DA(28) XOR DA(34) XOR DA(40) XOR DA(46);

Hash_fun(5)=DA(5) XOR DA(11) XOR DA(17) XOR DA(23) XOR DA(29) XOR DA(35) XOR DA(41) XOR DA(47);
```

This function is used as an offset into a 64-bit hash table stored in MACHASH1 and MACHASH2 that indicates whether a particular address should be accepted or not.

The MAC hash address register 1 (MACHASH1) is shown in Figure 22-73 and described in Table 22-84.

Figure 22-73. MAC Hash Address Register 1 (MACHASH1)



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

Table 22-84. MAC Hash Address Register 1 (MACHASH1) Field Descriptions

Bit	Field	Value	Description
31-0	MACHASH1	0-FFFF FFFFh	Least-significant 32 bits of the hash table corresponding to hash values 0 to 31. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

22.5.38 MAC Hash Address Register 2 (MACHASH2)

The MAC hash address register 2 (MACHASH2) is shown in Figure 22-74 and described in Table 22-85.

Figure 22-74. MAC Hash Address Register 2 (MACHASH2)

31	16
MACHASH2	
R/W-0	

15	0
MACHASH2	
R/W-0	

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

Table 22-85. MAC Hash Address Register 2 (MACHASH2) Field Descriptions

Bit	Field	Value	Description
31-0	MACHASH2	0-FFFF FFFFh	Most-significant 32 bits of the hash table corresponding to hash values 32 to 63. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

22.5.39 Back Off Test Register (BOFFTEST)

The back off test register (BOFFTEST) is shown in Figure 22-75 and described in Table 22-86.

Figure 22-75. Back Off Random Number Generator Test Register (BOFFTEST)

31	26	25	16
Reserved		RNDNUM	
R-0		R-0	

15	12	11	10	9	0
COLLCOUNT		Reserved		TXBACKOFF	
R-0		R-0		R-0	

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

Table 22-86. Back Off Test Register (BOFFTEST) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	RNDNUM	0-3FFh	Backoff random number generator. This field allows the Backoff Random Number Generator to be read. Reading this field returns the generator's current value. The value is reset to 0 and begins counting on the clock after the deassertion of reset.
15-12	COLLCOUNT	0-Fh	Collision count. These bits indicate the number of collisions the current frame has experienced.
11-10	Reserved	0	Reserved
9-0	TXBACKOFF	0-3FFh	Backoff count. This field allows the current value of the backoff counter to be observed for test purposes. This field is loaded automatically according to the backoff algorithm, and is decremented by one for each slot time after the collision.

22.5.40 Transmit Pacing Algorithm Test Register (TPACETEST)

The transmit pacing algorithm test register (TPACETEST) is shown in Figure 22-76 and described in Table 22-87.

Figure 22-76. Transmit Pacing Algorithm Test Register (TPACETEST)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		PACEVAL	
R-0		R-0	

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

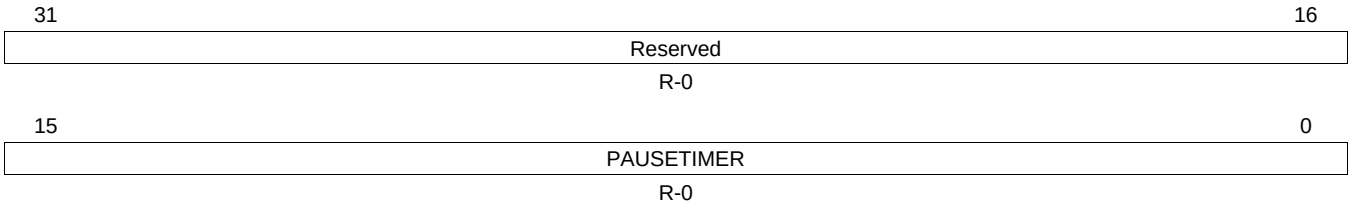
Table 22-87. Transmit Pacing Algorithm Test Register (TPACETEST) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PACEVAL	0-1Fh	Pacing register current value. A nonzero value in this field indicates that transmit pacing is active. A transmit frame collision or deferral causes PACEVAL to be loaded with 1Fh (31); good frame transmissions (with no collisions or deferrals) cause PACEVAL to be decremented down to 0. When PACEVAL is nonzero, the transmitter delays four Inter Packet Gaps between new frame transmissions after each successfully transmitted frame that had no deferrals or collisions. If a transmit frame is deferred or suffers a collision, the IPG time is not stretched to four times the normal value. Transmit pacing helps reduce capture effects, which improves overall network bandwidth.

22.5.41 Receive Pause Timer Register (RXPAUSE)

The receive pause timer register (RXPAUSE) is shown in Figure 22-77 and described in Table 22-88.

Figure 22-77. Receive Pause Timer Register (RXPAUSE)



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

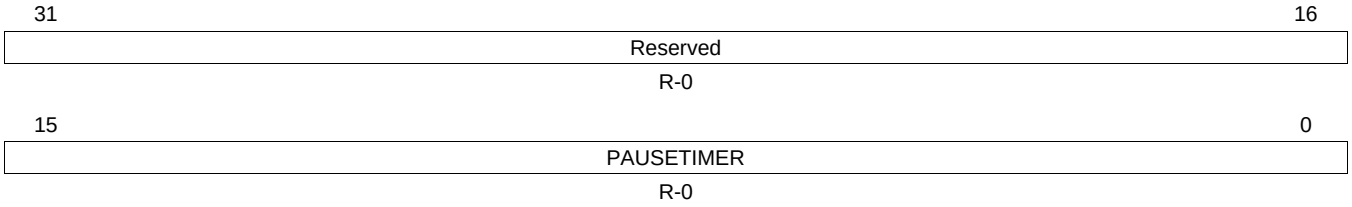
Table 22-88. Receive Pause Timer Register (RXPAUSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Receive pause timer value. These bits allow the contents of the receive pause timer to be observed. The receive pause timer is loaded with FF00h when the EMAC sends an outgoing pause frame (with pause time of FFFFh). The receive pause timer is decremented at slot time intervals. If the receive pause timer decrements to 0, then another outgoing pause frame is sent and the load/decrement process is repeated.

22.5.42 Transmit Pause Timer Register (TXPAUSE)

The transmit pause timer register (TXPAUSE) is shown in Figure 22-78 and described in Table 22-89.

Figure 22-78. Transmit Pause Timer Register (TXPAUSE)



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

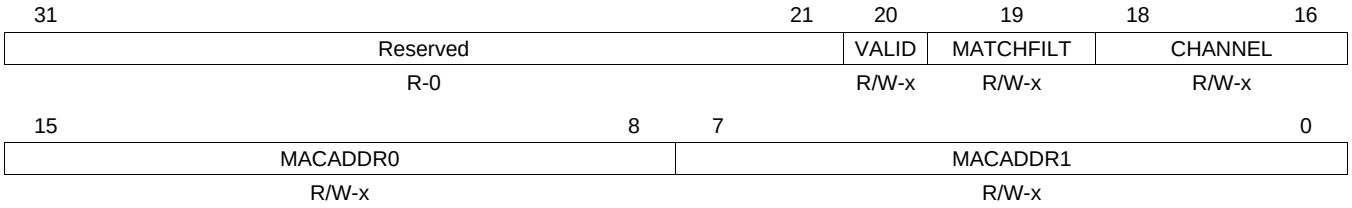
Table 22-89. Transmit Pause Timer Register (TXPAUSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Transmit pause timer value. These bits allow the contents of the transmit pause timer to be observed. The transmit pause timer is loaded by a received (incoming) pause frame, and then decremented at slot time intervals down to 0, at which time EMAC transmit frames are again enabled.

22.5.43 MAC Address Low Bytes Register (MACADDRLO)

The MAC address low bytes register used in address matching (MACADDRLO), is shown in Figure 22-79 and described in Table 22-90.

Figure 22-79. MAC Address Low Bytes Register (MACADDRLO)



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

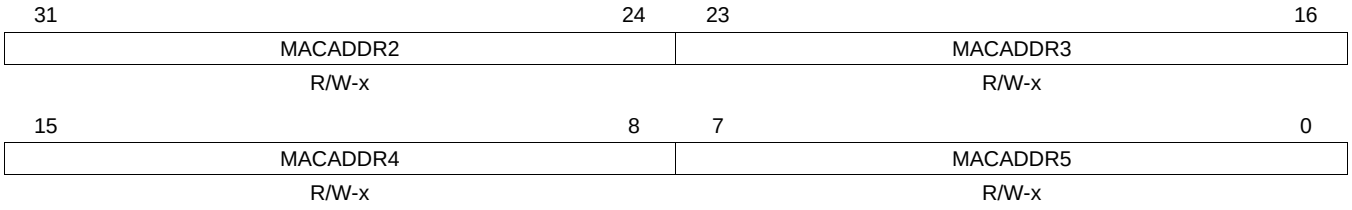
Table 22-90. MAC Address Low Bytes Register (MACADDRLO) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	VALID		Address valid bit. This bit should be cleared to zero for unused address channels
		0	Address is not valid and will not be used for matching or filtering incoming packets
		1	Address is valid and will be used for matching or filtering incoming packets
19	MATCHFILT		Match or filter bit
		0	The address will be used (if the VALID bit is set) to filter incoming packet addresses
		1	The address will be used (if the VALID bit is set) to match incoming packet addresses
18-16	CHANNEL	0-7h	Channel select. Determines which receive channel a valid address match will be transferred to. The channel is a don't care if MATCHFILT is cleared to 0.
15-8	MACADDR0	0-FFh	MAC address lower 7-0 bits (byte 0)
7-0	MACADDR1	0-FFh	MAC address bits 15-8 (byte 1)

22.5.44 MAC Address High Bytes Register (MACADDRHI)

The MAC address high bytes register (MACADDRHI) is shown in Figure 22-80 and described in Table 22-91.

Figure 22-80. MAC Address High Bytes Register (MACADDRHI)



LEGEND: R/W = Read/Write; -x = value is indeterminate after reset

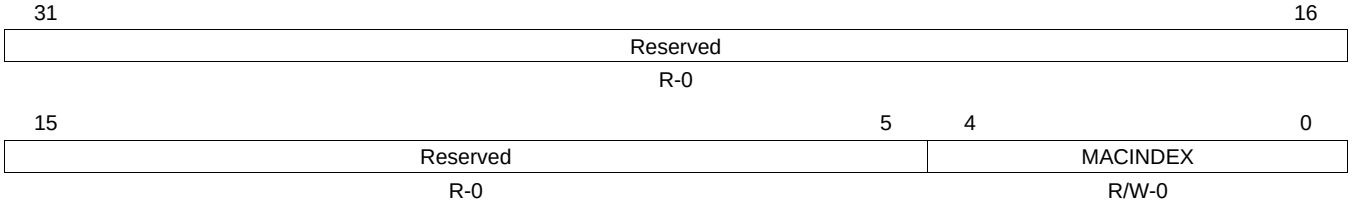
Table 22-91. MAC Address High Bytes Register (MACADDRHI) Field Descriptions

Bit	Field	Value	Description
31-24	MACADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACADDR5	0-FFh	MAC source address bits 47-40 (byte 5). Bit 40 is the group bit. It is forced to 0 and read as 0. Therefore, only unicast addresses are represented in the address table.

22.5.45 MAC Index Register (MACINDEX)

The MAC index register (MACINDEX) is shown in Figure 22-81 and described in Table 22-92.

Figure 22-81. MAC Index Register (MACINDEX)



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

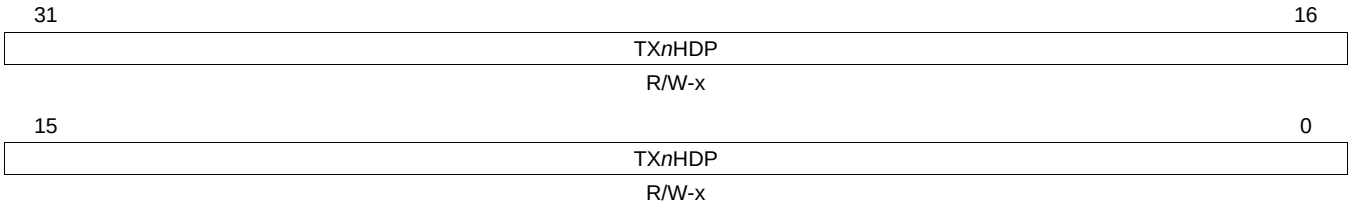
Table 22-92. MAC Index Register (MACINDEX) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	MACINDEX	0-1Fh	Mac Address Index. The 53-bits of each address location are written with three consecutive 32-bit VBUS accesses. The host must write the index into the ram in the MACINDEX(4:0) register, followed by the upper 32-bits of address (which are stored in a holding register), followed by the lower 16- bits of address (with control bits). The 53-bit indexed ram location is written when the low location is written.

22.5.46 Transmit Channel DMA Head Descriptor Pointer Registers (TX0HDP-TX7HDP)

The transmit channel 0-7 DMA head descriptor pointer register (TX*n*HDP) is shown in Figure 22-82 and described in Table 22-93.

Figure 22-82. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP)



LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

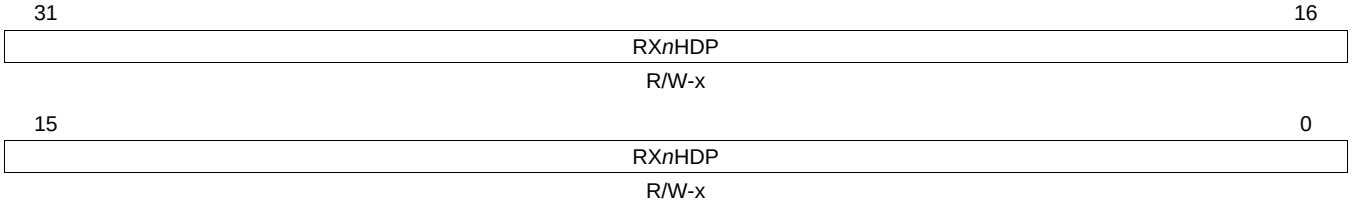
Table 22-93. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP) Field Descriptions

Bit	Field	Value	Description
31-0	TX*n*HDP	0-FFFF FFFFh	Transmit channel *n* DMA Head Descriptor pointer. Writing a transmit DMA buffer descriptor address to a head pointer location initiates transmit DMA operations in the queue for the selected channel. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

22.5.47 Receive Channel DMA Head Descriptor Pointer Registers (RX0HDP-RX7HDP)

The receive channel 0-7 DMA head descriptor pointer register (RX*n*HDP) is shown in Figure 22-83 and described in Table 22-94.

Figure 22-83. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP)



LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

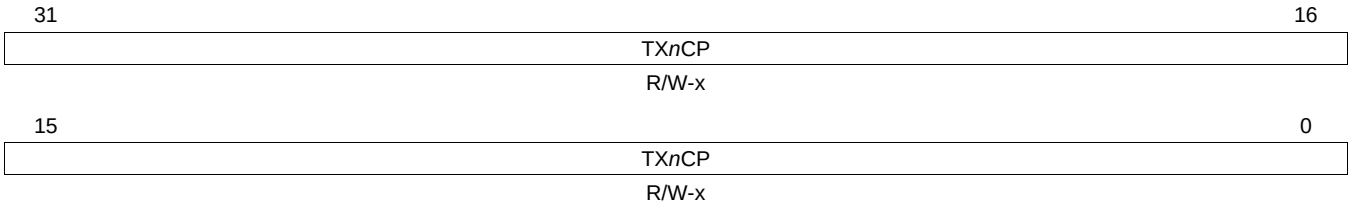
Table 22-94. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP) Field Descriptions

Bit	Field	Value	Description
31-0	RX*n*HDP	0-FFFF FFFFh	Receive channel *n* DMA Head Descriptor pointer. Writing a receive DMA buffer descriptor address to this location allows receive DMA operations in the selected channel when a channel frame is received. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

22.5.48 Transmit Channel Completion Pointer Registers (TX0CP-TX7CP)

The transmit channel 0-7 completion pointer register (TX*n*CP) is shown in Figure 22-84 and described in Table 22-95.

Figure 22-84. Transmit Channel *n* Completion Pointer Register (TX*n*CP)



LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

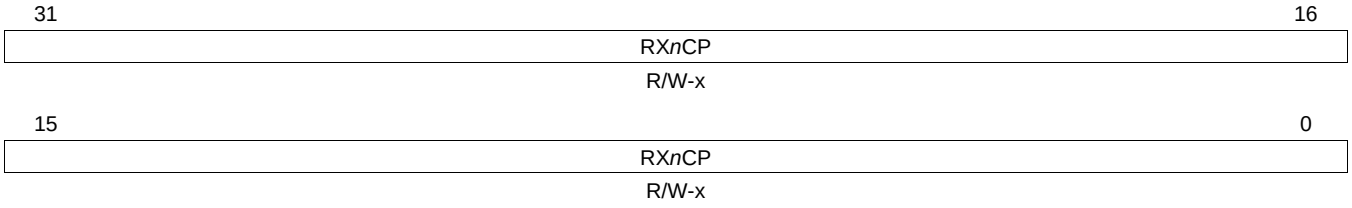
Table 22-95. Transmit Channel *n* Completion Pointer Register (TX*n*CP) Field Descriptions

Bit	Field	Value	Description
31-0	TX*n*CP	0-FFFF FFFFh	Transmit channel *n* completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

22.5.49 Receive Channel Completion Pointer Registers (RX0CP-RX7CP)

The receive channel 0-7 completion pointer register (RX*n*CP) is shown in Figure 22-85 and described in Table 22-96.

Figure 22-85. Receive Channel *n* Completion Pointer Register (RX*n*CP)



LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

Table 22-96. Receive Channel *n* Completion Pointer Register (RX*n*CP) Field Descriptions

Bit	Field	Value	Description
31-0	RX*n*CP	0-FFFF FFFFh	Receive channel *n* completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

NOTE: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPGMAC dma controller (port). The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted. The value written is not actually stored in the location. The interrupt is deasserted if the two values are equal.

22.5.50 Network Statistics Registers

The EMAC has a set of statistics that record events associated with frame traffic. The statistics values are cleared to zero 38 clocks after the rising edge of VBUSP_RST_N. When the GMIIEN bit in the MACCONTROL register is set, all statistics registers (see Figure 22-86) are write-to-decrement. The value written is subtracted from the register value with the result stored in the register. If a value greater than the statistics value is written, then zero is written to the register (writing FFFF FFFFh clears a statistics location). When the GMIIEN bit is cleared, all statistics registers are read/write (normal write direct, so writing 0000 0000h clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STATPEND) is issued, if enabled, when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from FFFF FFFFh to 0000 0000h.

Figure 22-86. Statistics Register

31	16
COUNT	
R/WD-0	

15	0
COUNT	
R/WD-0	

LEGEND: R/W = Read/Write; WD = Write to decrement; -*n* = value after reset

22.5.50.1 Good Receive Frames Register (RXGOODFRAMES)

The total number of good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See **Section 22.2.5.5** for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.2 Broadcast Receive Frames Register (RXBCASTFRAMES)

The total number of good broadcast frames received on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for address FF-FF-FF-FF-FF-FFh only
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.3 Multicast Receive Frames Register (RXMCASTFRAMES)

The total number of good multicast frames received on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.4 Pause Receive Frames Register (RXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames received by the EMAC (whether acted upon or not). A pause frame is defined as having all of the following:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08h and the opcode 0001h
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error
- Pause-frames had been enabled on the EMAC (TXFLOWEN bit is set in MACCONTROL).

The EMAC could have been in either half-duplex or full-duplex mode. See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.5 Receive CRC Errors Register (RXCRCERRORS)

The total number of frames received on the EMAC that experienced a CRC error. A frame with CRC errors is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no alignment or code error
- Had a CRC error. A CRC error is defined as having all of the following:
  - A frame containing an even number of nibbles
  - Fails the frame check sequence test

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.6 Receive Alignment/Code Errors Register (RXALIGNCODEERRORS)

The total number of frames received on the EMAC that experienced an alignment error or code error. Such a frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had either an alignment error or a code error
  - An alignment error is defined as having all of the following:
    - A frame containing an odd number of nibbles
    - Fails the frame check sequence test, if the final nibble is ignored
  - A code error is defined as a frame that has been discarded because the EMACs MII_RXER pin is driven with a one for at least one bit-time's duration at any point during the frame's reception.

Overruns have no effect on this statistic.

CRC alignment or code errors can be calculated by summing receive alignment errors, receive code errors, and receive CRC errors.

22.5.50.7 Receive Oversized Frames Register (RXOVERSIZED)

The total number of oversized frames received on the EMAC. An oversized frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN in bytes
- Had no CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.8 Receive Jabber Frames Register (RXJABBER)

The total number of jabber frames received on the EMAC. A jabber frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN bytes long
- Had a CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.9 Receive Undersized Frames Register (RXUNDERSIZED)

The total number of undersized frames received on the EMAC. An undersized frame is defined as having all of the following:

- Was any data frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes long
- Had no CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.10 Receive Frame Fragments Register (RXFRAGMENTS)

The total number of frame fragments received on the EMAC. A frame fragment is defined as having all of the following:

- Any data frame (address matching does not matter)
- Was less than 64 bytes long
- Had a CRC error, alignment error, or code error
- Was not the result of a collision caused by half duplex, collision based flow control

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.11 Filtered Receive Frames Register (RXFILTERED)

The total number of frames received on the EMAC that the EMAC address matching process indicated should be discarded. Such a frame is defined as having all of the following:

- Was any data frame (not MAC control frame) destined for any unicast, broadcast, or multicast address
- Did not experience any CRC error, alignment error, code error
- The address matching process decided that the frame should be discarded (filtered) because it did not match the unicast, broadcast, or multicast address, and it did not match due to promiscuous mode.

To determine the number of receive frames discarded by the EMAC for any reason, sum the following statistics (promiscuous mode disabled):

- Receive fragments
- Receive undersized frames
- Receive CRC errors
- Receive alignment/code errors
- Receive jabbers
- Receive overruns
- Receive filtered frames

This may not be an exact count because the receive overruns statistic is independent of the other statistics, so if an overrun occurs at the same time as one of the other discard reasons, then the above sum double-counts that frame.

22.5.50.12 Receive QOS Filtered Frames Register (RXQOSFILTERED)

The total number of frames received on the EMAC that were filtered due to receive quality of service (QOS) filtering. Such a frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- The frame destination channel flow control threshold register (RX*n*FLOWTHRESH) value was greater than or equal to the channel's corresponding free buffer register (RX*n*FREEBUFFER) value
- Was of length 64 to RXMAXLEN
- RXQOSEN bit is set in RXMBPENABLE
- Had no CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.13 Receive Octet Frames Register (RXOCTETS)

The total number of bytes in all good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 22.2.5.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

22.5.50.14 Good Transmit Frames Register (TXGOODFRAMES)

The total number of good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

22.5.50.15 Broadcast Transmit Frames Register (TXBCASTFRAMES)

The total number of good broadcast frames transmitted on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame destined for address FF-FF-FF-FF-FF-FFh only
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

22.5.50.16 Multicast Transmit Frames Register (TXMCASTFRAMES)

The total number of good multicast frames transmitted on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

22.5.50.17 Pause Transmit Frames Register (TXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames transmitted by the EMAC. Pause frames cannot underrun or contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect on this statistic. Pause frames sent by software are not included in this count. Since pause frames are only transmitted in full-duplex mode, carrier loss and collisions have no effect on this statistic.

Transmitted pause frames are always 64-byte multicast frames so appear in the multicast transmit frames register and 64 octet frames register statistics.

22.5.50.18 Deferred Transmit Frames Register (TXDEFERRED)

The total number of frames transmitted on the EMAC that first experienced deferment. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect on this statistic.

22.5.50.19 Transmit Collision Frames Register (TXCOLLISION)

The total number of times that the EMAC experienced a collision. Collisions occur under two circumstances:

- When a transmit data or MAC control frame has all of the following:
  - Was destined for any unicast, broadcast, or multicast address
  - Was any size
  - Had no carrier loss and no underrun
  - Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic increments on each occasion if a frame experiences multiple collisions (and increments on late collisions).
- When the EMAC is in half-duplex mode, flow control is active, and a frame reception begins.

CRC errors have no effect on this statistic.

22.5.50.20 Transmit Single Collision Frames Register (TXSINGLECOLL)

The total number of frames transmitted on the EMAC that experienced exactly one collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect on this statistic.

22.5.50.21 Transmit Multiple Collision Frames Register (TXMULTICOLL)

The total number of frames transmitted on the EMAC that experienced multiple collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect on this statistic.

22.5.50.22 Transmit Excessive Collision Frames Register (TXEXCESSIVECOLL)

The total number of frames when transmission was abandoned due to excessive collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect on this statistic.

22.5.50.23 Transmit Late Collision Frames Register (TXLATECOLL)

The total number of frames when transmission was abandoned due to a late collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions that had previously required the transmission to be reattempted. The late collisions statistic dominates over the single, multiple, and excessive collisions statistics. If a late collision occurs, the frame is not counted in any of these other three statistics.

CRC errors, carrier loss, and underrun have no effect on this statistic.

22.5.50.24 Transmit Underrun Error Register (TXUNDERRUN)

The number of frames sent by the EMAC that experienced FIFO underrun. Late collisions, CRC errors, carrier loss, and underrun have no effect on this statistic.

22.5.50.25 Transmit Carrier Sense Errors Register (TXCARRIERSENSE)

The total number of frames on the EMAC that experienced carrier loss. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted)

CRC errors and underrun have no effect on this statistic.

22.5.50.26 Transmit Octet Frames Register (TXOCTETS)

The total number of bytes in all good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

22.5.50.27 Transmit and Receive 64 Octet Frames Register (FRAME64)

The total number of 64-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was exactly 64-bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame is recorded in this statistic).

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

22.5.50.28 Transmit and Receive 65 to 127 Octet Frames Register (FRAME65T127)

The total number of 65-byte to 127-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 65-bytes to 127-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

22.5.50.29 Transmit and Receive 128 to 255 Octet Frames Register (FRAME128T255)

The total number of 128-byte to 255-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 128-bytes to 255-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

22.5.50.30 Transmit and Receive 256 to 511 Octet Frames Register (FRAME256T511)

The total number of 256-byte to 511-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 256-bytes to 511-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

22.5.50.31 Transmit and Receive 512 to 1023 Octet Frames Register (FRAME512T1023)

The total number of 512-byte to 1023-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 512-bytes to 1023-bytes long

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

22.5.50.32 Transmit and Receive 1024 to RXMAXLEN Octet Frames Register (FRAME1024TUP)

The total number of 1024-byte to RXMAXLEN-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 1024-bytes to RXMAXLEN-bytes long

CRC/alignment/code errors, underruns, and overruns do not affect frame recording in this statistic.

22.5.50.33 Network Octet Frames Register (NETOCTETS)

The total number of bytes of frame data received and transmitted on the EMAC. Each frame counted has all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address (address match does not matter)
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced (multiple retries are counted each time)
- Every byte received if the EMAC is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence is not counted to prevent double-counting).

Error conditions such as alignment errors, CRC errors, code errors, overruns, and underruns do not affect the recording of bytes in this statistic. The objective of this statistic is to give a reasonable indication of Ethernet utilization.

22.5.50.34 Receive FIFO or DMA Start of Frame Overruns Register (RXSOFOVERRUNS)

The total number of frames received on the EMAC that had either a FIFO or DMA start of frame (SOF) overrun. An SOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available at the start of the frame).

CRC errors, alignment errors, and code errors have no effect on this statistic.

22.5.50.35 Receive FIFO or DMA Middle of Frame Overruns Register (RXMOFOVERRUNS)

The total number of frames received on the EMAC that had either a FIFO or DMA middle of frame (MOF) overrun. An MOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available after the frame was successfully started - no SOF overrun).

CRC errors, alignment errors, and code errors have no effect on this statistic.

22.5.50.36 Receive DMA Overruns Register (RXDMAOVERRUNS)

The total number of frames received on the EMAC that had either a DMA start of frame (SOF) overrun or a DMA middle of frame (MOF) overrun. A receive DMA overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the DMA buffer resources to receive it (zero head descriptor pointer at the start or during the middle of the frame reception).

CRC errors, alignment errors, and code errors have no effect on this statistic.

22.6 EMAC/MDIO Glossary

Broadcast MAC Address— A special Ethernet MAC address used to send data to all Ethernet devices on the local network. The broadcast address is FFh-FFh-FFh-FFh-FFh-FFh. The LSB of the first byte is odd, qualifying it as a group address; however, its value is reserved for broadcast. It is classified separately by the EMAC.

Descriptor (Packet Buffer Descriptor)— A small memory structure that describes a larger block of memory in terms of size, location, and state. Descriptors are used by the EMAC and application to describe the memory buffers that hold Ethernet data.

Device — In this document, device refers to the processor.

Ethernet MAC Address (MAC Address)— A unique 6-byte address that identifies an Ethernet device on the network. In an Ethernet packet, a MAC address is used twice, first to identify the packet's destination, and second to identify the packet's sender or source. An Ethernet MAC address is normally specified in hexadecimal, using dashes to separate bytes. For example, 08h-00h-28h-32h-17h-42h.

The first three bytes normally designate the manufacturer of the device. However, when the first byte of the address is odd (LSB is 1), the address is a group address (broadcast or multicast). The second bit specifies whether the address is globally or locally administrated (not considered in this document).

Ethernet Packet (Packet)— An Ethernet packet is the collection of bytes that represents the data portion of a single Ethernet frame on the wire.

Full Duplex— Full-duplex operation allows simultaneous communication between a pair of stations using point-to-point media (dedicated channel). Full-duplex operation does not require that transmitters defer, nor do they monitor or react to receive activity, as there is no contention for a shared medium in this mode. Full-duplex mode can only be used when all of the following are true:

- The physical medium is capable of supporting simultaneous transmission and reception without interference.
- There are exactly two stations connected with a full duplex point-to-point link. As there is no contention for use of a shared medium, the multiple access (that is, CSMA/CD) algorithms are unnecessary.
- Both stations on the LAN are capable of, and have been configured to use, full-duplex operation.

The most common configuration envisioned for full-duplex operation consists of a central bridge (also known as a switch) with a dedicated LAN connecting each bridge port to a single device.

Full-duplex operation constitutes a proper subset of the MAC functionality required for half-duplex operation.

Half Duplex— In half-duplex mode, the CSMA/CD media access method is the means by which two or more stations share a common transmission medium. To transmit, a station waits (defers) for a quiet period on the medium, that is, no other station is transmitting. It then sends the intended message in bit-serial form. If, after initiating a transmission, the message collides with that of another station, then each transmitting station intentionally transmits for an additional predefined period to ensure propagation of the collision throughout the system. The station remains silent for a random amount of time (backoff) before attempting to transmit again.

Host— The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port (EMAC) interrupts. In this document, host refers to the device.

Jabber— A condition wherein a station transmits for a period of time longer than the maximum permissible packet length, usually due to a fault condition.

Link— The transmission path between any two instances of generic cabling.

Multicast MAC Address— A class of MAC address that sends a packet to potentially more than one recipient. A group address is specified by setting the LSB of the first MAC address byte to 1. Thus, 01h-02h-03h-04h-05h-06h is a valid multicast address. Typically, an Ethernet MAC looks for only certain multicast addresses on a network to reduce traffic load. The multicast address list of acceptable packets is specified by the application.

Physical Layer and Media Notation— To identify different Ethernet technologies, a simple, three-field, type notation is used. The Physical Layer type used by the Ethernet is specified by these fields:

<data rate in Mb/s><medium type><maximum segment length (×100m)>

The definitions for the technologies mentioned in this document are in Table 22-97.

Table 22-97. Physical Layer Definitions

Term	Definition
10Base-T	IEEE 802.3 Physical Layer specification for a 10 Mb/s CSMA/CD local area network over two pairs of twisted-pair telephone wire.
100Base-T	IEEE 802.3 Physical Layer specification for a 100 Mb/s CSMA/CD local area network over two pairs of Category 5 unshielded twisted-pair (UTP) or shielded twisted-pair (STP) wire.
Twisted pair	A cable element that consists of two insulated conductors twisted together in a regular fashion to form a balanced transmission line.

Port— Ethernet device.

Promiscuous Mode— EMAC receives frames that do not match its address.

High-End CAN Controller (HECC)

This document contains a general description of the controller area network (CAN). The CAN uses established protocol to communicate serially with other controllers in harsh environments. This document describes the Standard CAN controller (SCC) and the High-End CAN controller (HECC) modules. Unless otherwise stated in the text, all information is related to both the SCC and HECC.

23.1 CAN Overview

The CAN controller is available in two different implementations that are both fully compliant with the CAN protocol, version 2.0B. The two different CAN controller versions use the same CAN protocol kernel module to perform the basic CAN protocol tasks. Only the message controller differs between the two CAN controller versions.

Key features of the CAN module include:

- Common CAN protocol kernel (CPK) to perform protocol tasks
- Standard CAN controller (SCC) for standard CAN applications
  - Sixteen message-object acceptance-filtering
- High-end CAN controller (HECC) for complex applications
  - Thirty-two message objects full-mask acceptance-filtering

Table 23-1. SCC and HECC Features Overview

Feature	SCC	HECC
Number of message objects	16 Receive/Transmit	32 Receive/Transmit
Number of receive identifier masks	3	32
CAN, version 2.0B compliant	X	X
Low-power mode	X	X
Programmable wake-up on bus activity	X	X
Programmable interrupt scheme	X	X
Automatic reply to a remote request	X	X
Automatic retransmission in case of error	X	X
Protect against reception of new message	X	X
32-bit time stamp		X
Local network time counter		X
Programmable priority register for each message		X
Programmable transmission and reception time-out		X

23.1.1 CAN Protocol Processor Features

The CAN protocol kernel (CPK) performs the basic CAN protocol tasks according to CAN protocol specification 2.0B. The CPK is the basic module of all CAN controller implementations. The CPK provides the following features:

- Full implementation of CAN protocol, version 2.0B
  - Standard and extended identifiers
  - Data and remote frames
- Bus speed of up to 1 Mbps

- Programmable prescaler from 1 to 256
- Programmable bit time according to the CAN specification
- Programmable sampling mode
  - One or three samples used to determine the received bit value
- Selectable edge of receive bit flow for synchronization
  - Both edges or falling edge only
- Automatic retransmission of a frame in case of loss of arbitration
- Low-power mode
- Bus failure diagnostic
  - Bus on/off
  - Error passive/active
  - Bus error warning
  - Bus stuck dominant
  - Frame error report: cyclic redundancy check (CRC), stuff, form, bit, and acknowledgment errors
  - Readable error counters
- Self-test mode
  - Operate in a loop-back mode (receiving its own message)
  - Dummy acknowledge

23.1.2 Standard CAN Controller (SCC) Features

The SCC provides the following features:

- All the CAN protocol kernel (CPK) features
  - Full implementation of CAN protocol, version 2.0B
- Sixteen message objects, each with the following properties:
  - Configurable as receive or transmit
  - Configurable with standard or extended identifier
  - Protects against reception of new message
  - Supports data and remote frame
  - Composed of 0 to 8 bytes of data
  - Employs a programmable interrupt scheme with two interrupt levels
  - Has a message priority based on object number
- Two programmable local identifier masks for objects 0-2 and 3-5
  - Configurable as standard or extended message identifier
  - Has an acceptance mask register for identifier extension bit
- One global programmable identifier mask for objects 6-15
  - Configurable as standard or extended message identifier
  - Has an acceptance mask register for identifier extension bit
- Low-power mode
- Programmable wake-up on bus activity
- Automatic reply to a remote request message
- Automatic retransmission of a frame in case of loss of arbitration or error

23.1.3 High-End CAN Controller (HECC) Features

The HECC provides the following features:

- All the CAN protocol kernel (CPK) features

  - Full implementation of CAN protocol, version 2.0B
- Thirty-two message objects, each with the following properties:
  - Configurable as receive or transmit
  - Configurable with standard or extended identifier
  - Has a programmable receive mask
  - Supports data and remote frame
  - Composed of 0 to 8 bytes of data
  - Uses a 32-bit time stamp on receive and transmit message
  - Protects against reception of new message
  - Holds the dynamically programmable priority of transmit message
  - Employs a programmable interrupt scheme with two interrupt levels
  - Employs a programmable alarm on transmission or reception time-out
- Low-power mode
- Programmable wake-up on bus activity
- Automatic reply to a remote request message
- Automatic retransmission of a frame in case of loss of arbitration or error
- 32-bit local network time counter synchronized by a specific message (communication in conjunction with mailbox 16)
- SCC-compatible mode
  - Upward-compatible software
  - Software written for the SCC can run without any changes on the HECC
  - SCC-compatible mode is automatically activated after RESET

23.2 CAN Network and Module Overview

The controller area network (CAN) uses a serial multimaster communication protocol that efficiently supports distributed real-time control, with a very high level of security, and a communication rate of up to 1 Mbps. The CAN bus is ideal for applications operating in noisy and harsh environments, such as in the automotive and other industrial fields that require reliable communication or multiplexed wiring.

Prioritized messages of up to eight bytes in data length can be sent on a multimaster serial bus using an arbitration protocol and an error-detection mechanism for a high level of data integrity.

23.2.1 CAN Protocol Overview

The CAN protocol supports four different frame types for communication:

- Data frames that carry data from a transmitter node to the receiver nodes
- Remote frames that are transmitted by a node to request the transmission of a data frame with the same identifier
- Error frames that are transmitted by any node on a bus-error detection
- Overload frames that provide an extra delay between the preceding and the succeeding data frames or remote frames.

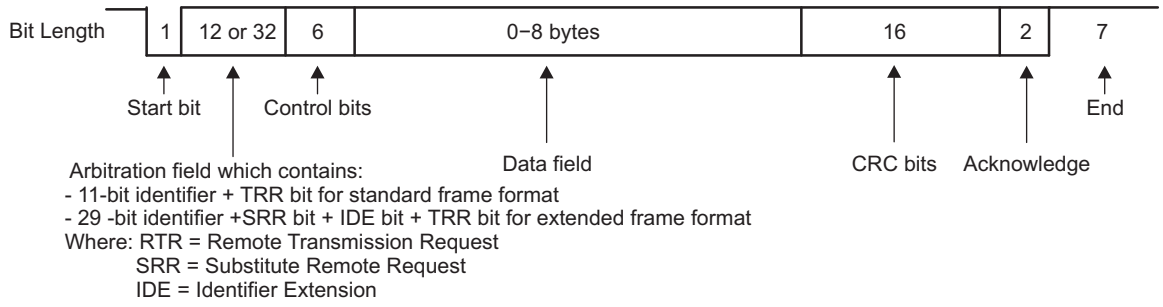
In addition, CAN specification version 2.0B defines two different formats that differ in the length of the identifier field: standard frames with an 11-bit identifier and extended frames with 29-bit identifier.

CAN standard data frames contain from 44 to 108 bits and CAN extended data frames contain 64 to 128 bits. Furthermore, up to 23 stuff bits can be inserted in a standard data frame, and up to 28 stuff bits in an extended data frame, depending on the data-stream coding. The overall maximum data frame length is then 131 bits for a standard frame and 156 bits for an extended frame.

Bit fields within the data frame, shown in Figure 23-1, identify:

- Start of the frame
- Arbitration field containing the identifier and the type of message being sent
- Control field containing the number of data bytes
- Up to 8 bytes of data
- Cyclic redundancy check (CRC)
- Acknowledgment
- End-of-frame bits

Figure 23-1. CAN Data Frame



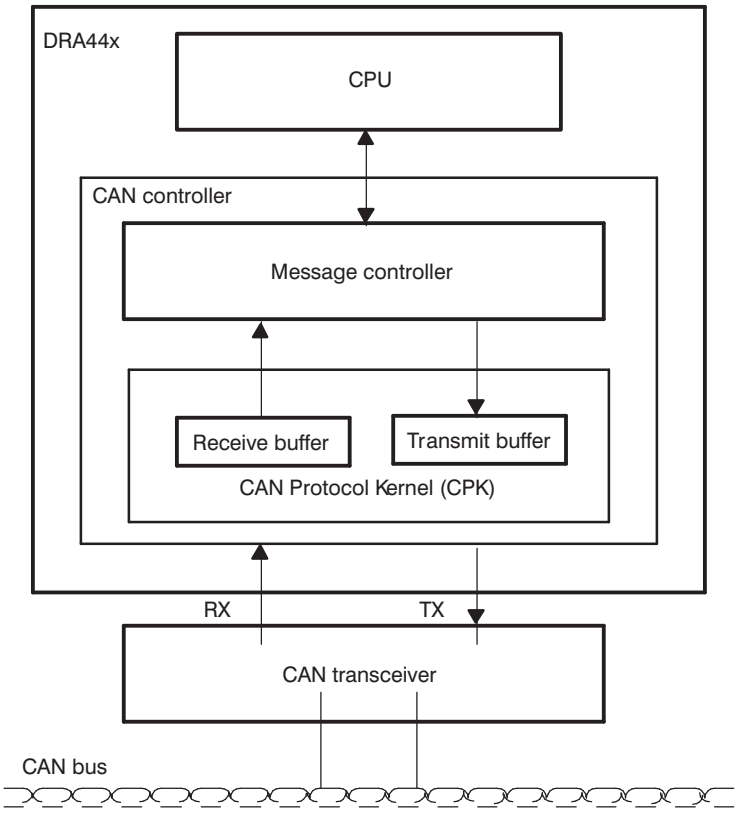
Note: Unless otherwise noted, numbers specify the number of bits in the field.

23.2.2 CAN Controller Overview

The CAN controllers provide the CPU with full functionality of the CAN protocol, version 2.0B. The CAN controller minimizes the CPU load in communication overhead and enhances the CAN standard by providing additional features.

The architecture of both the SCC and the HECC controllers is composed of a CAN protocol kernel (CPK) and a message controller (Figure 23-2).

Figure 23-2. Architecture of the SCC and HECC CAN Controllers



Two functions of the CAN protocol kernel (CPK) are to decode all messages received on the CAN bus according to the CAN protocol and to transfer these messages into a receive buffer. Another CPK function is to transmit messages on the CAN bus according to the CAN protocol.

The message controller of a CAN controller is responsible for determining if any message received by the CPK must be preserved for the CPU use or be discarded. At the initialization phase, the CPU specifies to the message controller all message identifiers used by the application. The message controller is also responsible for sending the next message to transmit to the CPK according to the message's priority.

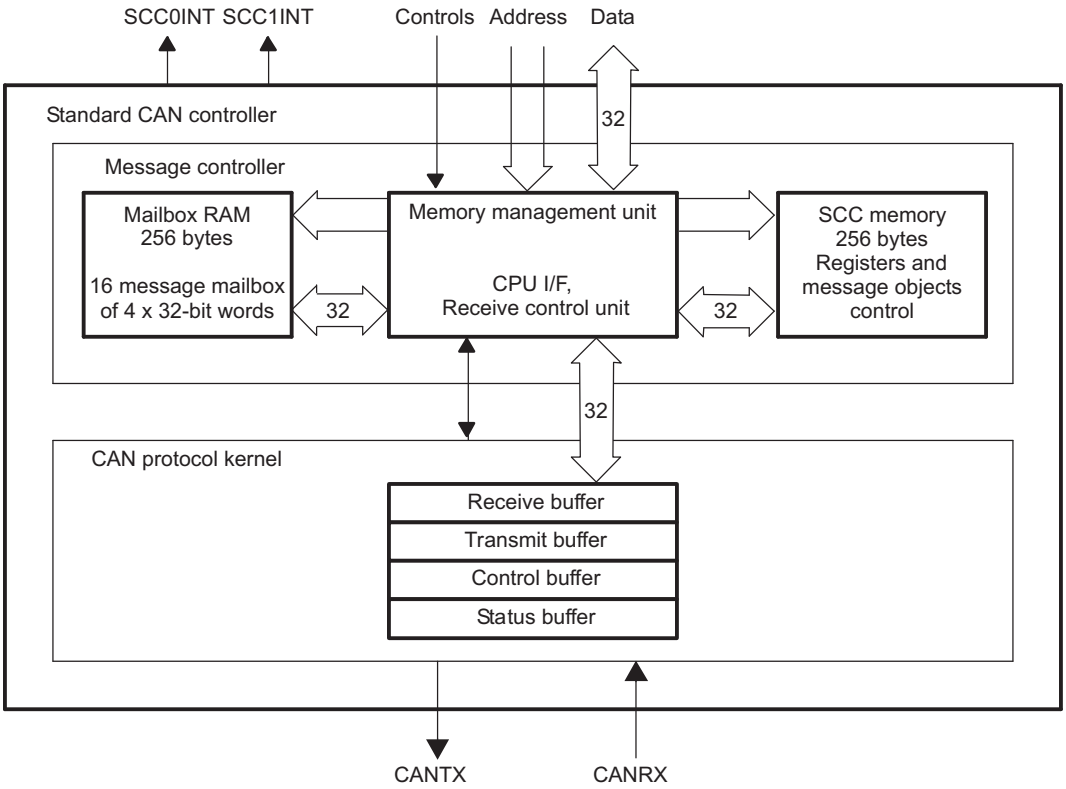
The SCC and the HECC differ only by their message controller - the CPK always offers the same services. The message management of the message controller is not part of the CAN protocol specification.

23.3 Standard CAN Controller (SCC) Overview

The SCC is a standard CAN controller with an internal 32-bit architecture, upwardly compatible with HECC. The SCC consists of:

- The CAN protocol kernel (CPK)
- The message controller:
  - The memory management unit (MMU) including the CPU interface and the receive control unit (acceptance filtering)
  - 256 bytes of mailbox RAM enabling the storage of 16 messages
  - 256 bytes of space register containing the global and the local identifier masks.

Figure 23-3. SCC Functional Block Diagram



After the reception of a valid message by the CAN protocol kernel (CPK), the receive control unit of the message controller determines if the message must be stored into one of the 16 message objects of the mailbox RAM. The receive control unit checks the state, the identifier, and the mask of all message objects to determine the appropriate mailbox location. The received message is stored into the first mailbox passing the acceptance filtering. If the receive control unit could not find any mailbox to store the received message, the message is discarded.

A message is composed of an identifier of 11 or 29 bits (contained in the arbitration field), a control field, and up to 8 bytes of data.

When a message must be transmitted, the message controller transfers the message into the transmit buffer of the CPK in order to start the message transmission at the next bus-idle state. When more than one message must be transmitted, the message with the highest priority is transferred into the CPK by the message controller.

The memory registers contain the global identifier mask and the two dedicated identifier masks for the message objects 0-2 and 3-5. After the reception of a valid identifier, the receive control unit checks the state and the identifier of all objects to determine if the received message must be stored into one of the message object's buffers.

To initiate a data transfer, the transmission-request bit has to be set in the corresponding control register. The entire transmission procedure and possible error handling is then done without any CPU involvement. If a mailbox has been configured to receive messages, the CPU easily reads its data registers using CPU read instructions. The mailbox may be configured to interrupt the CPU after every successful message transmission or reception.

23.3.1 SCC Memory Map

The SCC module has two different offset addresses mapped in memory. The first offset address, *SCC_Offset*, is used to access the control and status registers, and the acceptance masks of the message objects. This memory range can only be accessed 32-bits wide. The second offset address, *Mailbox_RAM_Offset,* is used to access the message mailboxes. This memory range can be accessed 8-bits, 16-bits, and 32-bits wide. Each of these two memory blocks uses 256 bytes of address space.

The message storage is implemented by RAM that is addressed by the CAN controller or the CPU. The CPU controls the CAN controller by modifying the various mailboxes in the RAM or the additional registers. The contents of the various storage elements are used to perform acceptance filtering, message transmission, and interrupt handling.

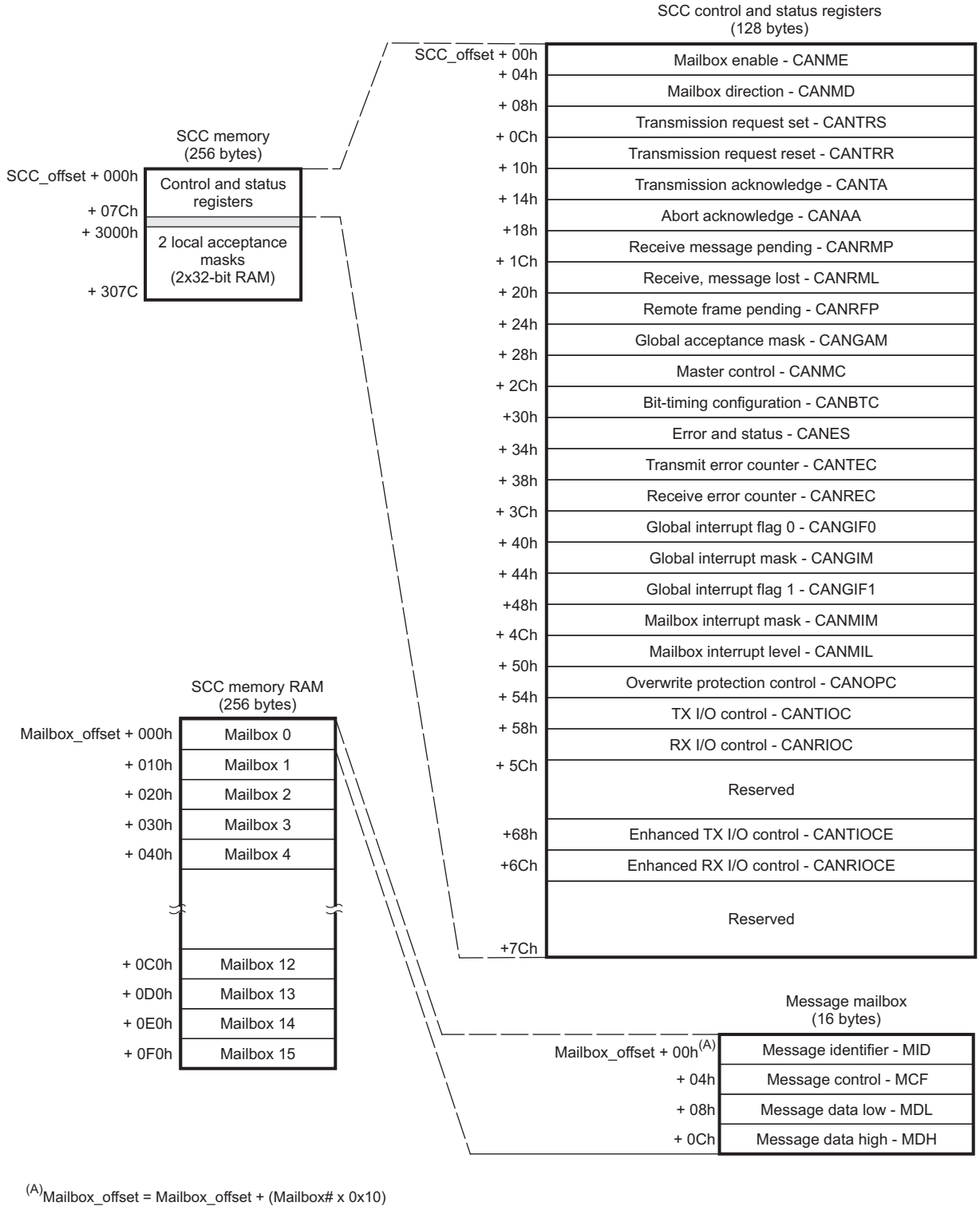
The mailbox module in the SCC provides 16 message mailboxes of 8-byte data lengths, a 29-bit identifier, and several control bits. Each mailbox can be configured to either transmit or receive data.

Unused Message Mailboxes

NOTE: All RAM areas are byte-writable and may be used as normal memory. In this case, it must be ensured that no CAN function uses the RAM area. This assurance is reached by disabling the corresponding mailbox or by disabling the corresponding functions.

The SCC and HECC registers, listed in Section 23.10, are used by the CPU to configure and control the CAN controller and the message objects

Figure 23-4. SCC Memory Map



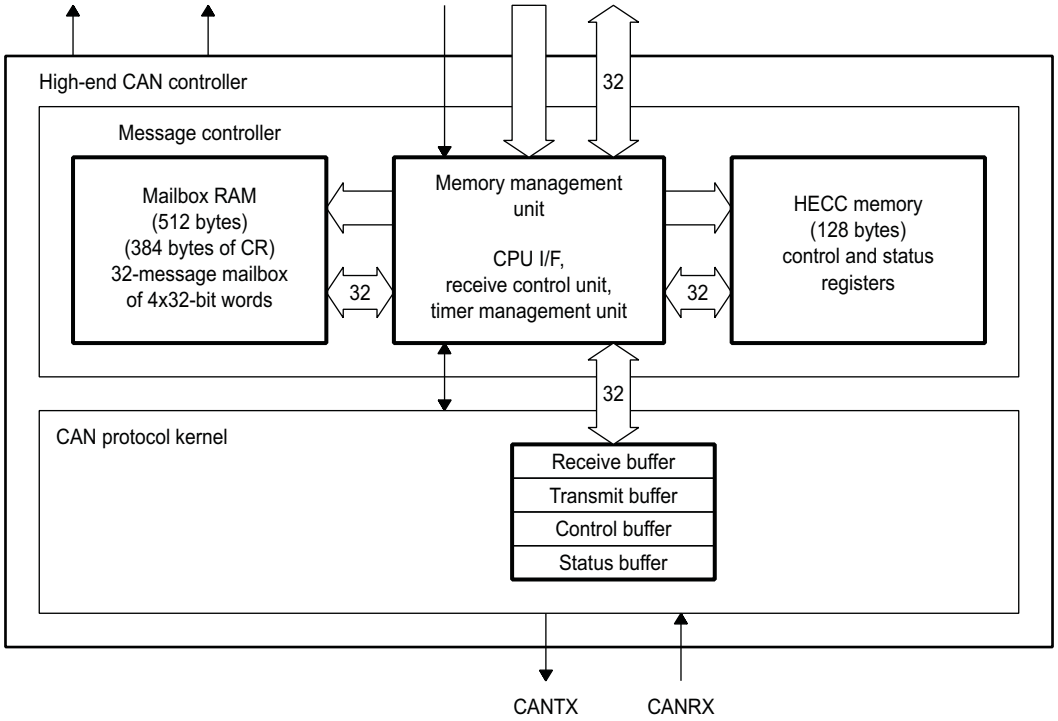
(A)Mailbox_offset = Mailbox_offset + (Mailbox# x 0x10)

23.4 High-End CAN Controller (HECC) Overview

The HECC is a new-generation, Texas Instruments, advanced CAN controller with an internal 32-bit architecture. The HECC, shown in Figure 23-5, consists of:

- The CAN protocol kernel (CPK)
- The message controller:
  - The memory management unit (MMU), including the CPU interface and the receive control unit (acceptance filtering), and the timer management unit
  - 512 bytes of mailbox RAM enabling the storage of 32 messages
  - 128 bytes of memory comprising the registers and the message objects control
  - 384 bytes of control RAM comprising the message objects control registers

Figure 23-5. HECC Functional Block Diagram



After the reception of a valid message by the CAN protocol kernel (CPK), the receive control unit of the message controller determines if the received message must be stored into one of the 32 message objects of the mailbox RAM. The receive control unit checks the state, the identifier, and the mask of all message objects to determine the appropriate mailbox location. The received message is stored into the first mailbox passing the acceptance filtering. If the receive control unit could not find any mailbox to store the received message, the message is discarded.

A message is composed of an identifier of 11 or 29 bits, a control field, and up to 8 bytes of data.

When a message must be transmitted, the message controller transfers the message into the transmit buffer of the CPK in order to start the message transmission at the next bus-idle state. When more than one message must be transmitted, the message with the highest priority (defined by the message-object-priority register) that is ready to be transmitted is transferred into the CPK by the message controller.

The timer management unit comprises a local network time counter and applies a time stamp to all messages received or transmitted. The timer management unit controls all message reception and transmission, and generates an alarm when a message has not been received or transmitted during an allowed period of time (time-out).

To initiate a data transfer, the transmission request bit has to be set in the corresponding control register. The entire transmission procedure and possible error handling are then performed without any CPU involvement. If a mailbox has been configured to receive messages, the CPU easily reads its data registers using CPU read instructions. The mailbox may be configured to interrupt the CPU after every successful message transmission or reception.

23.4.1 SCC-Compatible Mode

The HECC can be used in SCC mode. In this mode, all functions specific to the HECC are not available and the HECC behaves exactly as the SCC. This mode is selected by default in order to allow any application software written for the SCC to run on the HECC without any modification. When using the HECC in SCC-compatible mode, you must refer to the SCC specification only. The SCC-compatible mode is selected with the SCM bit in the master control register (CANMC).

23.4.2 HECC Memory Map

The HECC module has two different offset addresses mapped in memory. The first offset address, *HECC_Offset*, is used to access the control register, the status register, the acceptance mask, the time stamp, and the time-out of the message objects. The access to the control and status registers (memory range: *HECC_Offset* ... *HECC_Offset* + *07Ch*) is limited to 32-bit-wide accesses. The local acceptance masks, the time stamp registers, and the time-out registers can be accessed 8-bits, 16-bits, and 32-bits wide. The second offset address, *Mailbox_RAM_Offset,* is used to access the mailboxes. This memory range can be accessed 8-bits, 16-bits, and 32-bits wide. Each of these two memory blocks uses 512 bytes of address space.

The message storage is implemented by RAM that can be addressed by the CAN controller or the CPU. The CPU controls the CAN controller by modifying the various mailboxes in the RAM or the additional registers. The contents of the various storage elements are used to perform the functions of the acceptance filtering, message transmission, and interrupt handling.

The mailbox module in the HECC provides 32 message mailboxes of 8-byte data lengths, a 29-bit identifier, and several control bits. Each mailbox can be configured as either transmit or receive. In the HECC, each mailbox has an individual acceptance mask.

Unused Message Mailboxes

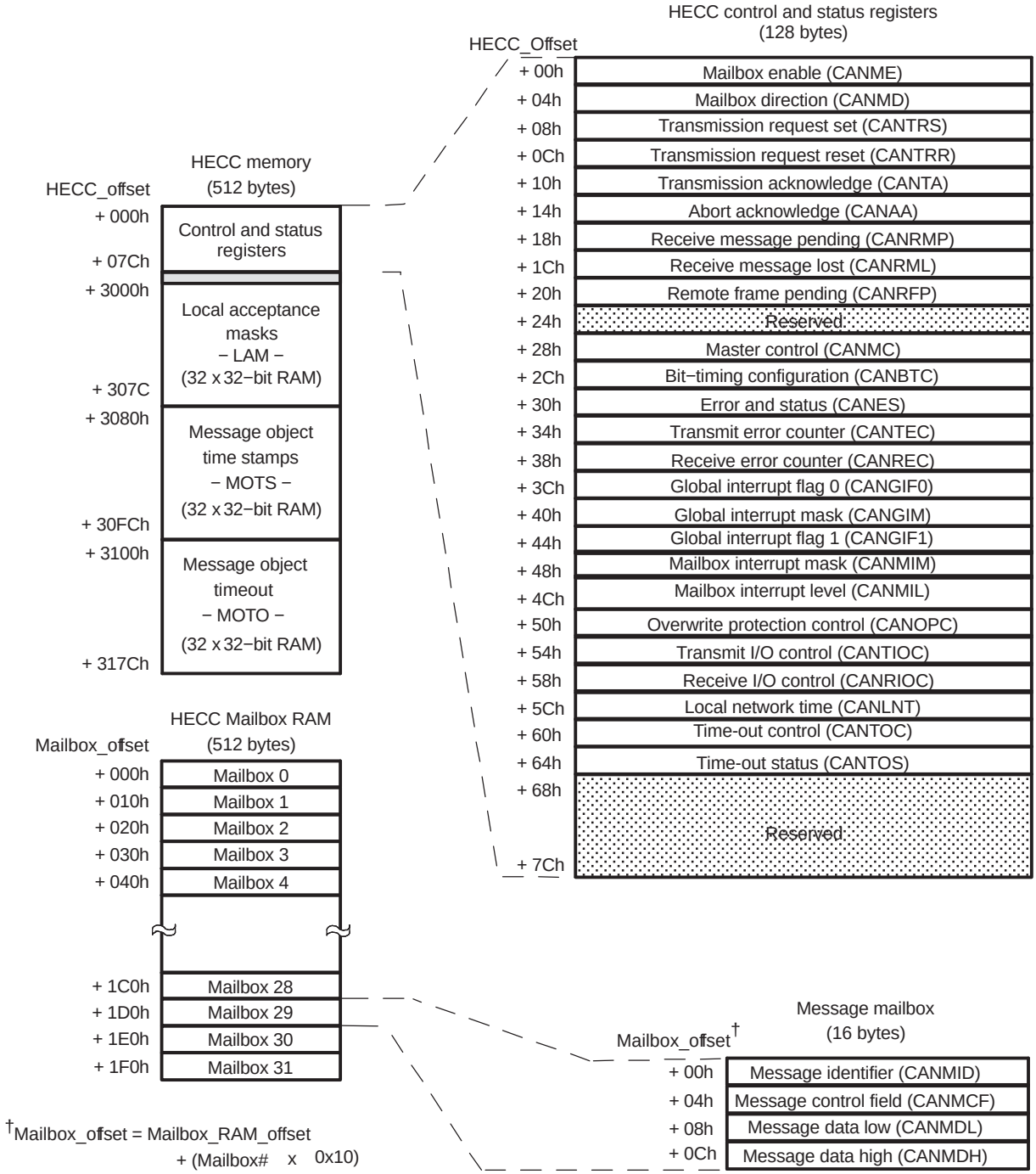
NOTE: All RAM areas are byte-writable and may be used as normal memory. In this case, it must be ensured that no CAN function uses the RAM area. This assurance is reached by disabling the corresponding mailbox or by disabling the corresponding functions.

HECC Used in SCC Mode

When the HECC is used in SCC mode, refer to the SCC memory map.

The SCC and HECC registers, listed in Section 23.10, are used by the CPU to configure and control the CAN controller and the message objects.

Figure 23-6. HECC Memory Map



23.5 Message Objects

23.5.1 SCC Message Objects

The message controller of the SCC handles 16 different mailboxes. Each mailbox is configured to either transmit or receive. In the SCC, mailboxes 0 to 2, 3 to 5, and 6 to 15 share the same acceptance masks.

An SCC mailbox consists of 20 bytes of RAM distributed, as shown in Table 23-2, as:

- A message mailbox RAM comprising
  - The 29-bit message identifier
  - The message control register
  - 8 bytes of message data
- A 29-bit acceptance mask

Furthermore, corresponding control and status bits located in the SCC registers allow control of the message objects.

Table 23-2. SCC Message Object Description

Offset Address[1]	Mnemonic	Name	Section
Mailbox_RAM_Offset + (Mailbox# + 10h) + 00h	CANMID	Message identifier	Section 23.10.4.1
Mailbox_RAM_Offset + (Mailbox# + 10h) + 04h	CANMCF	Message control field	Section 23.10.4.2
Mailbox_RAM_Offset + (Mailbox# + 10h) + 08h	CANMDL	Message data low word	Section 23.10.4.3
Mailbox_RAM_Offset + (Mailbox# + 10h) + 0Ch	CANMDH	Message data high word	Section 23.10.4.3
SCC_Offset + 3000h " for objects 0 to 2	CANLAM0	Local acceptance mask	Section 23.10.4.4
SCC_Offset + 300Ch " for objects 3 to 5	CANLAM3	Local acceptance mask	Section 23.10.4.4
SCC_Offset + 24h " for objects 6 to 15	CANGAM	Global acceptance mask	Section 23.10.1.10

(1) The actual addresses of the message objects are device-specific. See the device-specific datasheet to verify the SCC module memory offset and RAM offset.

Unused Message Mailboxes

NOTE: Message mailboxes not used by the application for CAN message (disabled in the CANME register) may be used as general memory by the CPU.

23.5.2 HECC Message Objects

The message controller of the HECC handles 32 different mailboxes. Each mailbox is configured to either transmit or receive. In the HECC, each mailbox has an individual acceptance mask.

A HECC mailbox consists of 28 bytes of RAM distributed, as shown in Table 23-3, as:

- A message mailbox RAM comprising
  - The 29-bit message identifier
  - The message control register
  - 8 bytes of message data
- A 29-bit acceptance mask
  - A 32-bit time stamp
  - A 32-bit time-out

Furthermore, corresponding control and status bits located in the HECC registers allow control of the message objects.

Table 23-3. HECC Message Object Description

Offset/Address[1]	Mnemonic	Name	Section
Mailbox_RAM_Offset + (Mailbox# + 10h) + 00h	CANMID	Message identifier	Section 23.10.4.1
Mailbox_RAM_Offset + (Mailbox# + 10h) + 04h	CANMCF	Message control field	Section 23.10.4.2
Mailbox_RAM_Offset + (Mailbox# + 10h) + 08h	CANMDL	Message data low word	Section 23.10.4.3
Mailbox_RAM_Offset + (Mailbox# + 10h) + 0Ch	CANMDH	Message data high word	Section 23.10.4.3
HECC_Offset + (Mailbox# x 4) + 3000h	CANLAM	Local acceptance mask	Section 23.10.4.4
HECC_Offset + (Mailbox# x 4) + 3080h	CANMOTS	Message object time stamp	Section 23.10.2.2
HECC_Offset + (Mailbox# x 4) + 3100h	CANMOTO	Message object time-out	Section 23.10.3.1

(1) The actual addresses of the message objects are device-specific. See the device-specific datasheet to verify the HECC module memory offset and RAM offset.

HECC Used in SCC Mode

NOTE: When the HECC is used in SCC mode, you must refer to the SCC message objects description (see Table 23-2).

Unused Message Mailboxes

NOTE: Message mailboxes not used by the application for CAN message (disabled in the CANME register) may be used as general memory by the CPU.

23.5.3 CAN Message Mailbox

The message mailboxes are the RAM area where the CAN messages are actually stored after they were received or before they are transmitted. The CPU may use the RAM area of the message mailboxes that are not used for storing messages as normal memory. This RAM area, unlike the register area, can be accessed by byte. Each mailbox contains:

- The message identifier
  - 29 bits for extended identifier
  - 11 bits for standard identifier
- Identifier extension (IDE) bit in the message identifier register (CANMID)
- Acceptance mask enable (AME) bit in the message identifier register (CANMID)
- Auto answer mode (AAM) bit in the message identifier register (CANMID)
- Remote transmission request (RTR) bit in the message control field register (CANMCF)
- Data length code (DLC) bits in the message control field register (CANMCF)
- Up to eight bytes for the data field
- On the HECC, transmit priority level (TPL) bits in the message control field register (CANMCF).

Each of the mailboxes can be configured as one of four message object types (see Table 23-4). Transmit and receive message objects are used for data exchange between one sender and multiple receivers (1-to-*n* communication link); whereas, request and reply message objects are used to set up a one-to-one communication link.

Table 23-4. Message Object Types

Message Object Types	MD*n* bit in CANMD	AAM bit in CANMID	RTR bit in CANMCF
Transmit message object	0	0	0
Receive message object	1	0	0
Request message object	1	0	1
Reply message object	0	1	0

23.5.3.1 Transmit Mailbox

The CPU stores the data to be transmitted in a mailbox configured as transmit mailbox. After writing the data and the identifier into the RAM, the message is sent if the corresponding TRS*n* bit in the transmit request set register (CANTRS) is set.

If more than one mailbox is configured as a transmit mailbox and more than one corresponding TRS*n* bit is set, the messages are sent one after another in descending order beginning with the mailbox with the highest priority.

In the SCC, the priority of the mailbox transmission depends on the mailbox number. The highest mailbox number (15) comprises the highest transmit priority.

In the HECC, the priority of the mailbox transmission depends on the setting of the TPL bits in the message control field register (CANMCF). The mailbox with the highest value in the TPL bits is transmitted first. Only when two mailboxes have the same value in the TPL bits is the higher numbered mailbox transmitted first. See Section 23.10.4.2 for more information about CANMCF.

If a transmission fails due to a loss of arbitration or an error, the message transmission is re-attempted. Before re-attempting the transmission, the CAN module checks if other transmissions are requested and then transmits the mailbox with the highest priority.

23.5.3.2 Receive Mailbox

The identifier of each incoming message is compared to the identifiers held in the receive mailboxes using the appropriate mask. When equality is detected, the received identifier, the control bits, and the data bytes are written into the matching RAM location. At the same time, the corresponding RMP*n* bit in the receive message pending register (CANRMP) is set and a receive interrupt is generated if enabled. If no match is detected, the message is not stored.

When a message is received, the message controller starts looking for a matching mailbox at the mailbox with the highest mailbox priority. Mailbox 15 of the SCC and of the HECC in SCC-compatible mode has the highest receive priority; mailbox 31 has the highest receive priority of the HECC in HECC mode.

NOTE: While operating in loop back mode:

- mailbox 0 receives data even if it is not programmed to receive data, or
- the last used reception mailbox, whether in loop back mode or normal operation mode, receives data even if none of the mailboxes is programmed to receive data.

The RMP*n* bit has to be reset by the CPU after reading the data. If a second message has been received for this mailbox and the RMP*n* bit is already set, the corresponding RML*n* bit in the receive message lost register (CANRML) is set. In this case, the stored message is overwritten with the new data if the corresponding OPC*n* bit in the overwrite protection control register (CANOPC) is cleared; otherwise, the next mailboxes are checked.

23.5.3.3 Handling of Remote Frames

If a remote frame is received (the incoming message has the RTR bit in the message control field register, CANMCF, set to 1), the CAN module compares the identifier to all identifiers of the mailboxes using the appropriate masks starting at the highest mailbox number in descending order.

- In the case of a matching identifier (with the message object configured as a send mailbox and the AAM bit in the message identifier register, CANMID, set to 1 in this message object), this message object is marked to be sent (corresponding TRS*n* bit in the transmit request set register (CANTRS) is set).
- In the case of a matching identifier (with the message object configured as a send mailbox and the AAM bit in CANMID cleared to 0 in this message object), this message is not received.

  After finding a matching identifier in a send mailbox, no further compare is done.
- In the case of a matching identifier and the message object configured as a receive mailbox, this message is handled like a data frame and the corresponding RMP*n* bit in the receive message pending register (CANRMP) is set. The CPU then has to decide how to handle this situation.

If the CPU wants to change the data in a message object that is configured as a remote frame mailbox (the AAM bit in CANMID is set to 1), the CPU has to set the MBNR bits and the CDR bit in the master control register (CANMC). The CPU may then perform the access and clear the CDR bit to inform the CAN module that the access is finished. Until the CDR bit is cleared, the transmission of this mailbox is not performed by the CAN module. Since CANTRS is not affected by the CDR bit, a pending transmission is started after the CDR bit is cleared; thus, the newest data is sent.

In order to change the identifier in that mailbox, the message object first must be disabled (the ME*n* bit in the mailbox enable register, CANME, is cleared to 0).

If the CPU wants to request data from another node, it may configure the message object as a receive mailbox and set the corresponding TRS*n* bit in CANTRS. In this case, the CAN module sends a remote frame request and receives the data frame in the same mailbox that sent the request. Therefore, only one mailbox is necessary to do a remote request. Note that the CPU must set the RTR bit in CANMCF to enable a remote frame transmission.

The behavior of the message object *n* is configured with the MD*n* bit in CANMD, the AAM bit in CANMID, and the RTR bit in CANMCF. Table 23-4 shows how to configure a message object according to the desired behavior.

To summarize, a message object can be configured with four different behaviors:

1. Transmit message object: only able to transmit messages.
2. Receive message object: only able to receive messages.
3. Request message object: able to transmit a remote request frame and to wait for the corresponding data frame.
4. Reply message object: able to transmit a data frame whenever a remote request frame is received for

the corresponding identifier.

Remote Transmission Request Bit

NOTE: When a remote transmission request is successfully transmitted with a message object configured in request mode, the TA*n* bit in the transmission acknowledge register (CANTA) is not set and no interrupt is generated. When the remote reply message is received, the behavior of the message object is the same as a message object configured in receive mode.

23.5.3.4 CPU Message Mailbox Access

Write accesses to the identifier is only accomplished when the mailbox is disabled (the ME*n* bit in the mailbox enable register, CANME, is cleared to 0). During access to the data field, it is critical that the data does not change while the CAN module is reading it; hence, a write access to the data field is disabled for a receive mailbox.

For send mailboxes, an access is usually denied if the TRS*n* bit in CANTRS or the TRR*n* bit in CANTRR is set. In these cases, an interrupt may be asserted. A way to access those mailboxes is to set the CDR bit in CANMC before accessing the mailbox data.

After the CPU access is finished, the CPU must clear the CDR bit in CANMC by writing a 0 to the CDR bit. The CAN module checks the CDR bit before and after reading the mailbox. If the CDR bit is set during those checks, the CAN module does not transmit the message but continues to look for other transmit requests. The setting of the CDR bit also stops the write-denied interrupt (WDI) from being asserted.

23.5.4 CAN Acceptance Filter

In the SCC and in the HECC there are two different implementations of the handling of the masks for the mailboxes. If the SCM bit is set in the master control register (CANMC), SCC-compatible mode is enabled and the HECC behaves like the SCC.

The identifier of the incoming message is first compared to the message identifier of the mailbox (that is stored in the mailbox). Then, the appropriate acceptance mask is used to mask out the bits of the identifier that should not be compared.

23.5.4.1 SCC Acceptance Filtering

In the SCC (or in the HECC in SCC-compatible mode), the global acceptance mask register (CANGAM) is used for mailboxes 6 to 15. An incoming message is stored in the highest numbered mailbox with a matching identifier. If there is no matching identifier in mailboxes 15 to 6, the incoming message is compared to the identifier stored in mailboxes 5 to 3 and then 2 to 0.

The mailboxes 5 to 3 use the local acceptance mask register CANLAM3 of the SCC registers. The mailboxes 2 to 0 use the local acceptance mask register CANLAM0 of the SCC registers. See Section 23.10.4.4.

To modify the two local acceptance mask registers of the SCC (CANLAM0 and CANLAM3), the CAN module must be set in the initialization mode. See Section 23.6.

23.5.4.2 HECC Acceptance Filtering

Each of the 32 mailboxes of the HECC has its own local acceptance mask register, CANLAM0 to CANLAM31. There is no global acceptance mask register in the HECC.

23.6 CAN Module Initialization

The CAN module must be initialized before being used. Initialization is only possible if the CAN module is in initialization mode, see Figure 23-7. Programming the CCR bit in the master control register (CANMC) to 1 sets the initialization mode. The initialization is performed only when the CCE bit in the error and status register (CANES) is set to 1. Afterwards, the configuration registers may be written.

In order to modify the two local acceptance mask registers of the SCC (CANLAM0 and CANLAM3), the CAN module also must be set in the initialization mode. The global acceptance mask register (CANGAM) can be modified in normal mode and initialization mode. The module is activated again by programming the CCR bit in CANMC to 0. After a hardware reset, the initialization mode is active.

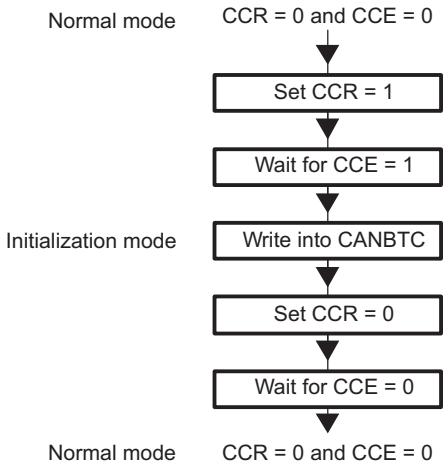
Bit-Timing Configuration Register (CANBTC) With Zero Value

NOTE: If the bit-timing configuration register (CANBTC) is programmed to 0, or left with the initial value, the CAN module never leaves the initialization mode; that is, the CCE bit in CANES remains set to 1 when clearing the CCR bit in CANMC.

Enter/Exit Initialization Mode

The transition between initialization mode and normal mode, and conversely, is performed in synchronization with the CAN network. That is, the CAN controller waits until it detects a bus idle sequence (11 recessive bits) before it changes the mode. In the event of a stuck-to-dominant bus error, the CAN controller cannot detect a bus-idle condition and is unable to perform a mode transition.

Figure 23-7. Configuration Sequence

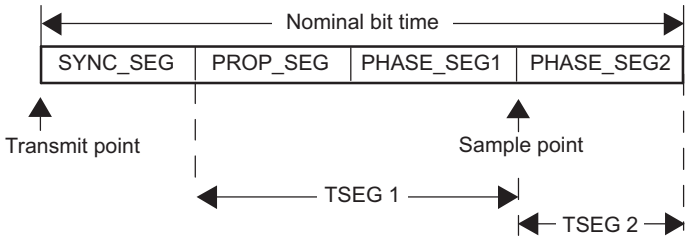


23.6.1 CAN Bit-Timing Configuration

As shown in Figure 23-8, the CAN protocol specification partitions the nominal bit time into four different time segments:

- SYNC_SEG: this part of bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment. This segment is always 1 TIME QUANTUM (TQ).
- PROP_SEG: this part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal propagation time on the bus line, the input comparator delay, and the output driver delay. This segment is programmable from 1 to 8 TIME QUANTA (TQ).
- PHASE_SEG1: this phase is used to compensate for positive edge phase error. This segment is programmable from 1 to 8 TIME QUANTA (TQ) and can be lengthened by resynchronization.
- PHASE_SEG2: this phase is used to compensate for negative edge phase error. This segment is programmable from 2 to 8 TIME QUANTA (TQ) and can be shortened by resynchronization.

Figure 23-8. Partition of the Bit Time



All controllers on a CAN bus must have the same bit rate and bit length. At different clock frequencies of the individual controllers, the bit rate has to be adjusted by the time segments.

In the SCC/HECC modules, the length of a bit on the CAN bus is determined by the parameters in the bit-timing configuration register (CANBTC): TSEG1, TSEG2, and BRP (*BRPrescaler* is the binary value of BRP + 1).

TSEG1 combines the two time segments PROP_SEG and PHASE_SEG1 as defined by the CAN protocol. TSEG2 defines the length of the time segment PHASE_SEG2.

The following bit-timing rules have to be fulfilled when determining the bit segment values:

- $TSEG1_{CALC(min)} \geq TSEG2_{CALC}$
- IPT ≤ $TSEG1_{CALC} \leq$ 16 TQ (IPT = Information Processing Time)
- IPT ≤ $TSEG2_{CALC} \leq$ 8 TQ
- IPT = $3/BRP_{CALC}$ (the resulting IPT has to be rounded up to the next integer value)

NOTE: For the special case of baud rate prescaler value BRP_{CALC} = 1, the Information Processing Time (IPT) is equal to three time quanta. This is not compliant to the ISO 11898 Standard, where the Information Processing Time is defined to be less than or equal to two time quanta. Thus the usage of this mode (BRP_{CALC} = 1) is not allowed.

- 1 TQ ≤ $SJW_{CALC} \leq$ min[4 TQ, $TSEG2_{CALC}$] (SJW = Synchronization Jump Width)
- To utilize three-time sampling mode $BRP_{CALC} \geq 5$ has to be selected

23.6.2 CAN Bit Rate Calculation

The bit rate is calculated as follows (in bits per second):

$$Bitrate = \frac{InternalBus\ CLK}{BRPrescaler \times BitTime}$$

Where *BitTime* is the number of time quanta (TQ) per bit. *Internal Bus CLK* is the CAN module system clock frequency. *BRPrescaler* is the BRP bits in the bit-timing configuration register (CANBTC) + 1.

BRPrescaler = BRP + 1

BitTime (TQ) = TSEG1 + TSEG2 + 1

Example:

With *Internal Bus CLK* = 26 MHz and *BRPrescaler* = 4, if a bit rate of 0.8125 Mbits/s is required, the bit-timing parameters is programmed as follows:

$$TQ = \frac{InternalBus\ CLK}{BRPrescaler \times Bitrate} = \frac{26\,MHz}{4 \times 0.8125\,Mbits/s} = 8$$

TQ = TSEG1 + TSEG2 + 1 = 4 + 3 + 1 = 8

TSEG1 = 4 TQ and TSEG2 = 3 TQ, so TQ = 8.

With this setting, a threefold sampling of the bus is not possible; thus the SAM bit in the bit-timing configuration register (CANBTC) must be cleared to 0. Since the SJW bits in CANBTC are not allowed to be greater than TSEG2 (that is set to 3 TQ), SJW = 3. (See Section 23.10.1.12 for more information about CANBTC.)

BTC = 0000021Ah

The value programmed into CANBTC.SJW has to be decremented by 1, because of the 2-bit size of the binary SJW value:

SJWCANBTC = SJW - 1

23.7 CAN Interrupts

There are two different types of interrupts. One type of interrupt is a mailbox-related interrupt; for example, the receive-message-pending interrupt or the abort-acknowledge interrupt. The other type of interrupt is a system interrupt that handles errors or system-related interrupt sources; for example, the error-passive interrupt or the wake-up interrupt. See Figure 23-9.

The following events may initiate one of the two interrupts:

- Mailbox interrupts
  - Message reception interrupt: a message was received
  - Message transmission interrupt: a message was transmitted successfully
  - Abort-acknowledge interrupt: a sent transmission was aborted
  - Receive-message-lost interrupt: an old message was overwritten by a new message
  - Message alarm interrupt (HECC only): one of the messages was not transmitted or received within a predefined time frame
- System interrupts
  - Write-denied interrupt: the CPU tried to write to a mailbox but was not allowed to
  - Wake-up interrupt: this interrupt is generated after a wake up
  - Bus-off interrupt: the CAN module enters the bus-off state
  - Error-passive interrupt: the CAN module enters the error-passive mode
  - Warning level interrupt: one or both error counters are greater than or equal to 96
  - Time counter overflow interrupt (HECC only): the local network time stamp counter had an overflow

Figure 23-9. SCC Interrupts Scheme Block Diagram

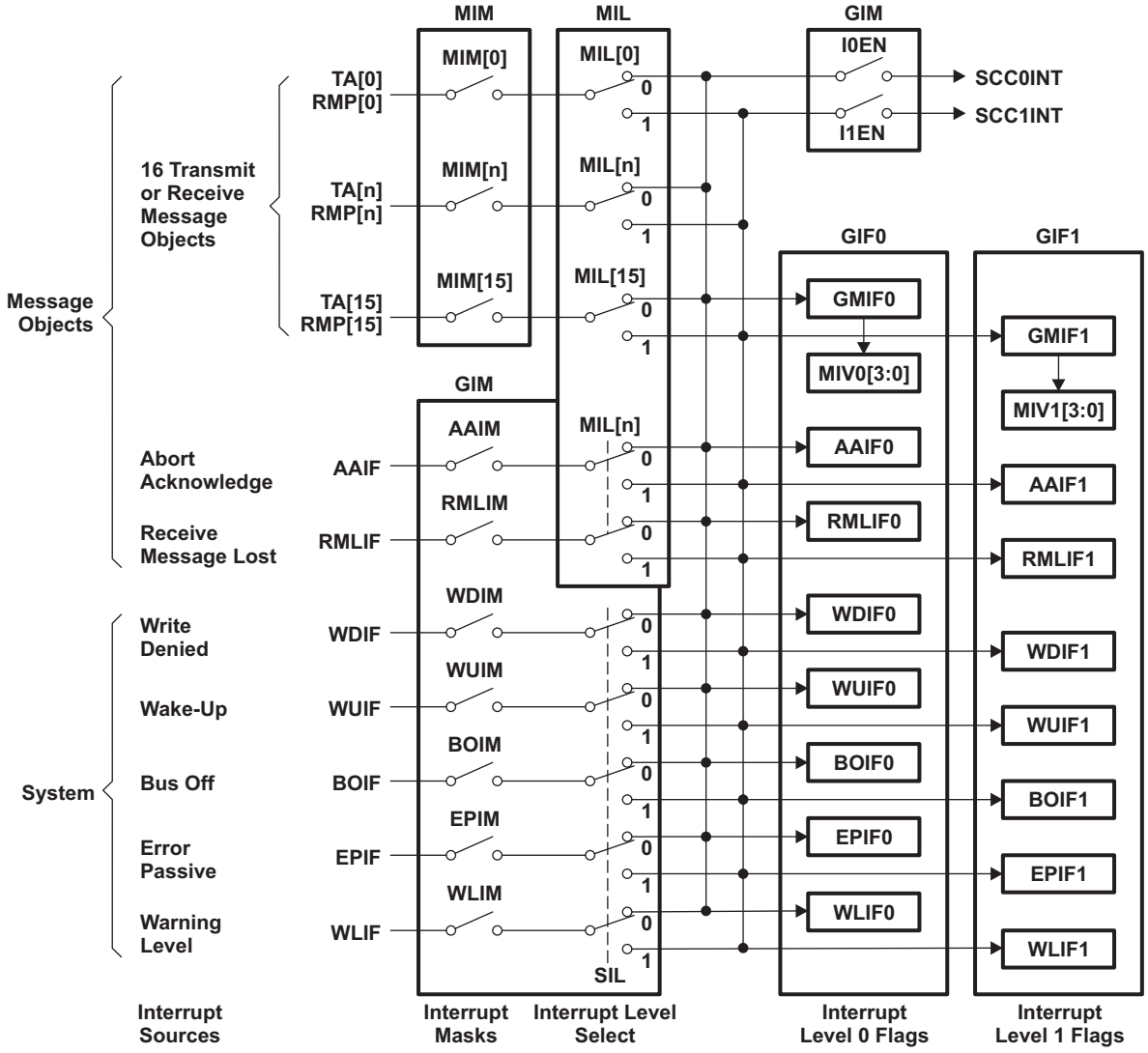
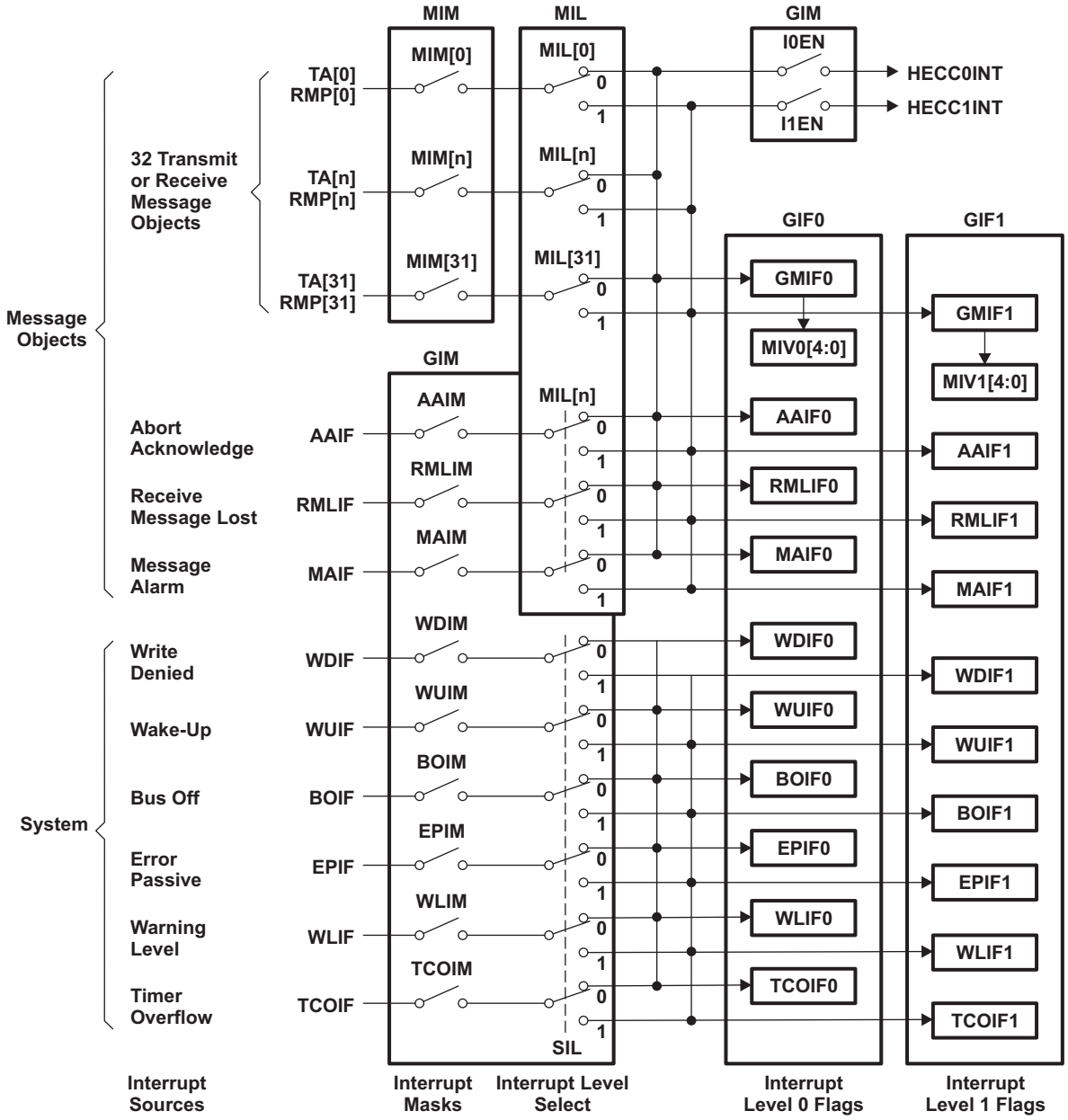


Figure 23-10. HECC Interrupts Scheme Diagram



23.7.1 Interrupts Scheme

The GMIF0/GMIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1) is set depending on the setting of the MIL*n* bit, in the mailbox interrupt level register (CANMIL), that corresponds to the mailbox originating that interrupt. If the MIL*n* bit is set, the corresponding message object interrupt flag sets the GMIF1 bit in CANGIF1; otherwise, it sets the GMIF0 bit in CANGIF0.

If all interrupt flags are cleared and a new interrupt flag is set, the CAN module interrupt output line (SCC0INT/HECC0INT or SCC1INT/HECC1INT) is activated if the corresponding interrupt mask bit is set. The interrupt line stays active until the interrupt flag is cleared by the CPU by writing a 1 to the appropriate bit.

The GMIF0 or GMIF1 bit must be cleared by writing a 1 to the appropriate bit in the transmission acknowledge register (CANTA) or in the receive message pending register (CANRMP) (depending on mailbox configuration) and cannot be cleared in CANGIF0/CANGIF1.

After clearing one or more interrupt flags, and one or more interrupt flags are still pending, a new interrupt is generated. The interrupt flags are cleared by writing a 1 to the corresponding bit location. If the GMIF0 or GMIF1 bit is set, the interrupt vector bit, MIV0 in CANGIF0 or MIV1 in CANGIF1, indicates the mailbox number of the mailbox that caused the setting of the GMIF0/GMIF1. It always displays the highest mailbox interrupt vector assigned to that interrupt line.

23.7.2 Message Object Interrupt

Each of the 32 mailboxes in the HECC or the 16 mailboxes in the SCC may initiate an interrupt on one of the two interrupt output lines 1 or 0. These interrupts are receive or transmit interrupts depending on the mailbox configuration.

There is one interrupt mask (MIM*n*) bit and one interrupt level (MIL*n*) bit dedicated to each mailbox. To generate a mailbox interrupt upon a receive/transmit event, the corresponding MIM*n* bit in the mailbox interrupt mask register (CANMIM) has to be set. If a CAN message is received in a receive mailbox (the RMP*n* bit in the receive message pending register, CANRMP, is set to 1) or is transmitted from a transmit mailbox (the TA*n* bit in the transmission acknowledge register, CANTA, is set to 1), an interrupt is asserted. If a mailbox is configured as a remote request mailbox (the MD*n* bit in the mailbox direction register, CANMD, is set to 1 and the RTR bit in the message control field register, MCF, is set to 1), an interrupt occurs upon reception of the reply frame. A remote reply mailbox generates an interrupt upon successful transmission of the reply frame (the MD*n* bit in CANMD is cleared to 0 and the AAM bit in the message identifier register, MID, is set to 1).

Setting the RMP*n* bit in CANRMP or the TA*n* bit in CANTA also sets the GMIF0/GMIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1), if the corresponding interrupt mask bit is set. The GMIF0/GMIF1 bit then generates an interrupt and the corresponding mailbox vector (mailbox number) is read from the MIV0/MIV1 bits in CANGIF0/CANGIF1. If more than one mailbox interrupts are pending, the actual value of the MIV0/MIV1 bits reflects the highest priority interrupt vector. The interrupt generated depends on the setting of the MIL*n* bit in the mailbox interrupt level register (CANMIL).

The AA*n* bit in the abort acknowledge register (CANAA) and the AAIF0/AAIF1 bit in CANGIF0/CANGIF1 are set when a transmit message is aborted by setting the TRR*n* bit in the transmit request reset register (CANTRR). An interrupt is asserted upon transmission abortion, if the AAIM bit in the global interrupt mask register (CANGIM) is set. Clearing the AA*n* bit(s) does not reset the AAIF0/AAIF1 bit. The interrupt bit has to be cleared separately. The interrupt line for the abort acknowledge interrupt is selected in accordance with the MIL*n* bit in CANMIL of the concerned mailbox.

A lost receive message is notified by setting the RML*n* bit in the receive message lost register (CANRML) and the RMLIF0/RMLIF1 bit in CANGIF0/CANGIF1. If an interrupt shall be generated upon the lost receive message event, the RMLIM bit in CANGIM has to be set. Clearing the RML*n* bit does not reset the RMLIF0/RMLIF1 bit. The interrupt bit has to be cleared separately. The interrupt line for the receive message lost interrupt is selected in accordance with the MIL*n* bit in CANMIL of the concerned mailbox.

Each mailbox of the HECC (in HECC mode only) is linked to a message object time-out register (CANMOTO). If a time-out event occurs (the TOS*n* bit in the time-out status register, CANTOS, is set to 1), a message alarm interrupt is asserted to one of the two interrupt lines if the MAIM bit in CANGIM is set. The interrupt line for the message alarm interrupt is selected in accordance with the MIL*n* bit in CANMIL of the concerned mailbox. Clearing the TOS*n* bit does not reset the MAIF0/MAIF1 bit in CANGIF0/CANGIF1.

23.8 CAN Power-Down Mode

There are two different power-down modes: the global power-down mode, when all clocks are stopped by the CPU; and the local power-down mode, when only the CAN module internal clock is deactivated by the CAN module itself. Therefore, it is possible to have a local power down, where only the clock of the CAN module logic is disabled, or a global power down when the clock for the complete device is disabled.

23.8.1 Local Power Down

The local power-down mode is requested by writing a 1 to the PDR bit in the master control register (CANMC). When the CAN module enters the local power-down mode, the power-down mode acknowledge (PDA) bit in the error and status register (CANES) is set to 1.

During local power-down mode, the clock of the CAN module is turned off and only the wake-up logic is still active. Since the clock is enabled for every access on an Internal Bus request, the contents of any register can be read-back even during power down.

The module leaves the local power-down mode when the PDR bit in CANMC is cleared or if any bus activity is detected on the CAN bus line (if the wake-up-on bus activity is enabled).

The automatic wake-up-on bus activity is enabled or disabled with the WUBA bit in CANMC. If there is any activity on the CAN bus line, the module begins its power-up sequence. The module waits until it detects 11 consecutive recessive bits on the CANRX pin and then it goes bus-active.

CAUTION

First Message Received During Power-Down Mode is Lost
The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, is lost.

After leaving the sleep mode, the PDR and PDA bits in CANMC are cleared. The CAN error counters remain unchanged.

If the module is transmitting a message when the PDR bit is set, the transmission is continued until a successful transmission, a lost arbitration, or an error condition on the CAN bus line occurs. Then, the PDA bit is activated. Thus, the module causes no error condition on the CAN bus line.

To implement the local power-down mode, two separate clocks are used within the CAN module. One clock stays active all the time to ensure power-down operation (the wake-up logic). The other clock is enabled depending on the setting of the PDA bit or Internal Bus access.

Accessing HECC Mailboxes

NOTE: HECC mailboxes can be accessed in LPD mode. Contents of the mailbox registers can be READ or WRITTEN to even after the PDA bit is set in the CANES.

23.8.2 Global Power Down

Global power-down mode is requested by the system module (SYS LPM). When the module is able to enter the global power-down mode, all clocks to the CAN module are disabled.

During global power-down mode, a dominant signal on the CAN bus may generate a wake-up interrupt, thus enabling the CPU to exit global power-down mode. There is no internal filtering for the CAN bus line. The WUIF0/WUIF1 bit is set asynchronously to enable the assertion of an interrupt without any running clocks.

23.9 Timer Management Unit

Several functions are implemented in the HECC to control the time when messages are transmitted. A separate state machine is included in the HECC to handle the time-control functions. This state machine has lower priority than the CAN state machine when accessing the registers. Therefore, the time-control functions may be delayed by other on-going actions.

23.9.1 Time-Stamp Functions

To get an indication of the time of reception or transmission of a message, a free-running 32-bit local network time counter (LNT) is implemented in the module. The value of the LNT is stored in the local network time register (CANLNT). When a received message is stored or a message has been transmitted, the content of LNT is written into the message object time-stamp register (CANMOTS) of the corresponding mailbox.

The counter is driven from the bit clock of the CAN bus line. The timer is stopped during the initialization mode or if the module is in sleep or suspend mode. After power-up reset, the free-running counter is cleared.

The most-significant bit of the local network time register (CANLNT) is cleared by writing a 1 to the LNTM bit in the master control register (CANMC). The CANLNT may also be cleared when mailbox 16 transmitted or received a message successfully (depending on the setting of the MD16 bit in the mailbox direction register, CANMD) by setting the LNTC bit to 1 in CANMC. Therefore, it is possible to use mailbox 16 for global time synchronization of the network. The CPU can read and write the counter.

Overflow of the counter is detected by the LNT counter overflow interrupt (TCOIF0/TCOIF1) bit in the global interrupt flag register (CANGIF0/CANGIF1). An overflow occurs when the highest bit of the LNT counter changes to 1. Thus, the CPU has enough time to handle this situation.

See Section 23.10.2 for the time-stamp registers.

23.9.2 Time-Out Functions

To ensure that all messages are sent or received within a predefined period, each mailbox has its own message time-out register (CANMOTO). If a message has not been sent or received by the time indicated in CANMOTO and the corresponding TOC*n* bit in the time-out control register (CANTOC) is set to 1, a TOS*n* bit in the time-out status register (CANTOS) is set to 1.

For transmit mailboxes, the TOS*n* bit is cleared when the TOC*n* bit is cleared or when the corresponding TRS*n* bit in the transmit request set register (CANTRS) is cleared, no matter whether due to successful transmission or abortion of the transmit request. For receive mailboxes, the TOS*n* bit is cleared when the corresponding TOC*n* bit is cleared.

The CPU may also clear the TOS*n* bits by writing a 1 into CANTOS.

The CANMOTO is implemented as a RAM. The state machine scans all the CANMOTOs and compares them to the LNT counter value. If the value in CANLNT is equal to or greater than the value in CANMOTO, while the corresponding TRS*n* bit is set (applies to transmit mailboxes only) and the TOC*n* bit is set, then the appropriate TOS*n* bit is set. Since all the time-out registers are scanned sequentially, there may be a delay before the TOS*n* bit is set.

See Section 23.10.3 for the time-out registers.

23.9.3 Behavior/Usage of MAIF0/1 Bit in User Applications

The MAIF0/1 bit is automatically cleared by the CAN protocol kernel (CPK) (along with the CANTOS.n bit) upon transmission/reception by the mailbox, which asserted this flag in the first place. It can also be cleared by the user (via the CPU). On a time out condition, the MAIF0/1 bit (and the CANTOS.n bit) is set. On an (eventual) successful communication, these bits are automatically cleared by the CPK. The following are the possible behaviors/usages for the MAIF0/1 bit:

1. Time-out condition occurs. Both the MAIF0/1 and CANTOS.n bits are set. Communication is never successful, i.e., the frame was never transmitted (or received). An interrupt is asserted. Application handles the issue and eventually clears both MAIF0/1 and CANTOS.n bits.
2. Time-out condition occurs. Both the MAIF0/1 and CANTOS.n bits are set. However, communication is eventually successful, i.e., the frame gets transmitted (or received). Both the MAIF0/1 and CANTOS.n bits are cleared automatically by the CPK. When the interrupt service routine (ISR) scans the CANGIF register, it does not see the MAIF0/1 bit set. This is the phantom interrupt scenario. The application merely returns to the main code.
3. Time-out condition occurs. Both the MAIF0/1 and CANTOS.n bits are set. While executing the ISR pertaining to time-out, communication is successful. This situation must be handled carefully. The application must not re-transmit a mailbox if the mailbox is sent between the time the interrupt is asserted, and the time the ISR is attempting to take corrective action. One way of doing this is to poll the TM/RM bits in the CANES register. These bits indicate if the CPK is currently transmitting/receiving. If so, the application must wait until the communication is over and then check the CANTOS.n bit again. If the communication is still not successful, the application must take corrective action.

23.10 Registers

The SCC and HECC registers listed in Table 23-5 are used by the CPU to configure and control the CAN controller and the message objects. All CAN registers support 8-, 16-, and 32-bit accesses. Reserved register bits must be written as 0.

NOTE: System behavior is undefined for accesses to reserved registers and must be avoided.

Table 23-5. SCC/HECC Registers

Offset Address[1]	Acronym	SCC Register Name	HECC Register Name	Section
00h[2]	CANME	Mailbox enable	Mailbox enable	Section 23.10.1.1
04h[2]	CANMD	Mailbox direction	Mailbox direction	Section 23.10.1.2
08h[2]	CANTRS	Transmit request set	Transmit request set	Section 23.10.1.3
0Ch[2]	CANTRR	Transmit request reset	Transmit request reset	Section 23.10.1.4
10h[2]	CANTA	Transmission acknowledge	Transmission acknowledge	Section 23.10.1.5
14h[2]	CANAA	Abort acknowledge	Abort acknowledge	Section 23.10.1.6
18h[2]	CANRMP	Receive message pending	Receive message pending	Section 23.10.1.7
1Ch[2]	CANRML	Receive message lost	Receive message lost	Section 23.10.1.8
20h[2]	CANRFP	Remote frame pending	Remote frame pending	Section 23.10.1.9
24h[3]	CANGAM	Global acceptance mask	Reserved	Section 23.10.1.10
28h[2]	CANMC	Master control	Master control	Section 23.10.1.11
2Ch[2]	CANBTC	Bit-timing configuration	Bit-timing configuration	Section 23.10.1.12
30h[2]	CANES	Error and status	Error and status	Section 23.10.1.13
34h[2]	CANTEC	Transmit error counter	Transmit error counter	Section 23.10.1.14
38h[2]	CANREC	Receive error counter	Receive error counter	Section 23.10.1.15
3Ch[2]	CANGIF0	Global interrupt flag 0	Global interrupt flag 0	Section 23.10.1.16
40h[2]	CANGIM	Global interrupt mask	Global interrupt mask	Section 23.10.1.17
44h[2]	CANGIF1	Global interrupt flag 1	Global interrupt flag 1	Section 23.10.1.16
48h[2]	CANMIM	Mailbox interrupt mask	Mailbox interrupt mask	Section 23.10.1.18
4Ch[2]	CANMIL	Mailbox interrupt level	Mailbox interrupt level	Section 23.10.1.19
50h[2]	CANOPC	Overwrite protection control	Overwrite protection control	Section 23.10.1.20
54h[2]	CANTIOC	Transmit I/O control	Transmit I/O control	Section 23.10.1.21
58h[2]	CANRIOC	Receive I/O control	Receive I/O control	Section 23.10.1.22
5Ch[4]	CANLNT	Reserved	Local network time	Section 23.10.2.1
[5] [6]	CANMOTO	Reserved	Message time-out	Section 23.10.3.1
[5] [6]	CANMOTS	Reserved	Message time stamp	Section 23.10.2.2
60h[4]	CANTOC	Reserved	Time-out control	Section 23.10.3.2
64h[4]	CANTOS	Reserved	Time-out status	Section 23.10.3.3
[5] [7]	CANMID	Message identifier	Message identifier	Section 23.10.4.1
[5] [7]	CANMCF	Message control field	Message control field	Section 23.10.4.2
[5] [7]	CANMDL	Message data low word	Message data low word	Section 23.10.4.3
[5] [7]	CANMDH	Message data high word	Message data high word	Section 23.10.4.3
[5] [6]	CANLAM0	Local acceptance mask 0	Local acceptance mask	Section 23.10.4.4

(1) The actual addresses of the registers are device specific. See the device-specific datasheet to verify the register addresses. The offset address of the device must be added to the actual register address.
(2) Relative address = + SCC/HECC_Offset
(3) Relative address = + SCC_Offset
(4) Relative address = + HECC_Offset
(5) See Table 23-2 for offset address of SCC mailbox registers and Table 23-3 for offset address of HECC mailbox registers.
(6) Relative address = + HECC_Offset + (Mailbox# x 4)
(7) Relative address = Mailbox_RAM_Offset + (Mailbox# x 10h)

Table 23-5. SCC/HECC Registers (continued)

Offset Address[1]	Acronym	SCC Register Name	HECC Register Name	Section
[5] [6]	CANLAM3	Local acceptance mask 3	Reserved	Section 23.10.4.4

23.10.1 SCC/HECC control registers

23.10.1.1 Mailbox Enable Register (CANME)

Each mailbox *n* is enabled or disabled by a corresponding bit *n* in this register.

Figure 23-11. Mailbox Enable Register (CANME) [00h]

31	16
ME.31:16	
R/W-0	
15	0
ME.15:0	
R/W-0	

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

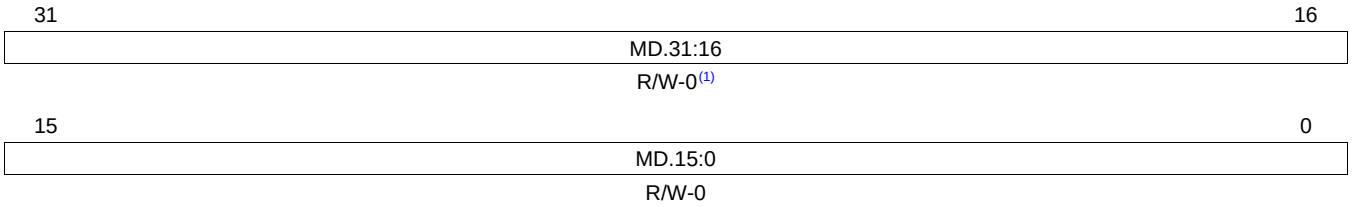
Table 23-6. Mailbox Enable Register (CANME) Field Descriptions

Bit	Field	Value	Description
31-0	ME		Mailbox Enable Register. After power-up, all bits in CANME are cleared to 0. Disabled mailboxes may be used as additional memory for the CPU.
		0	The corresponding mailbox *n* is disabled.
		1	The corresponding mailbox *n* is enabled for the CAN module. The mailbox must be disabled before writing to the contents of any identifier field. If the corresponding bit in CANME is set, the write access to the identifier of a message object is discarded.

23.10.1.2 Mailbox Direction Register (CANMD)

Each mailbox *n* is configured as a transmit or a receive mailbox by a corresponding bit *n* in this register.

Figure 23-12. Mailbox Direction Register (CANMD) [04h]



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

(1) HECC only, reserved in the SCC

Table 23-7. Mailbox Direction Register (CANMD) Field Descriptions

Bit	Field	Value	Description
31-0	MD		Mailbox Direction Register. After power-up, all bits in CANMD are cleared to 0.
		0	The corresponding mailbox *n* is configured as a transmit mailbox.
		1	The corresponding mailbox *n* is configured as a receive mailbox.

23.10.1.3 Transmission Request Set Register (CANTRS)

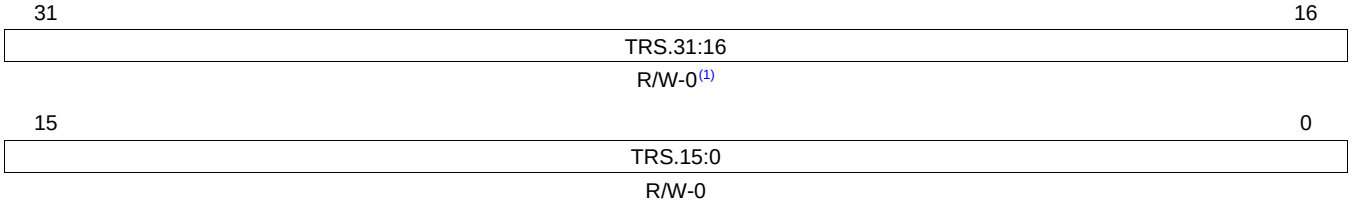
When mailbox *n* is ready to be transmitted, the CPU sets the corresponding TRS*n* bit to 1 in this register to start the transmission. A successful transmission also sets the GMIF0/GMIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1) and initiates a global mailbox interrupt, if the corresponding mailbox interrupt mask (MIM*n*) bit in the mailbox interrupt mask register (CANMIM) is set.

The TRS*n* bit is set by the CPU and reset by the CAN module logic. It is also set by the CAN module in the case of a remote frame request. The TRS*n* bit is reset in the case of a successful transmission or an aborted transmission (if requested). If a mailbox *n* is configured as a receive mailbox (MD*n* bit in the mailbox direction register, CANMD, is set), the corresponding TRS*n* bit is ignored. If the TRS*n* bit of a remote request mailbox is set, a remote frame is transmitted. If the CPU tries to set a bit while the CAN module tries to clear it, the bit is set.

Setting the TRS*n* bit causes the particular message *n* to be transmitted. Several bits can be set simultaneously. Therefore, all messages with the TRS*n* bit set are transmitted in turn, starting with the mailbox having the highest mailbox number (highest priority).

The bits in CANTRS are set by writing a 1 from the CPU; writing a 0 has no effect. After power-up, all bits in CANTRS are cleared to 0.

Figure 23-13. Transmission Request Set Register (CANTRS) [08h]



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

(1) HECC only, reserved in the SCC

Table 23-8. Transmission Request Set Register (CANTRS) Field Descriptions

Bit	Field	Value	Description
31-0	TRS		Transmit Request Set Register. After power-up, all bits in CANTRS are cleared to 0.
		0	No operation
		1	If the corresponding message is configured as a transmit mailbox or remote request mailbox, the data or remote message of this mailbox will be transmitted.

23.10.1.4 Transmission Request Reset Register (CANTRR)

Setting the TRR*n* bit of the message *n* causes a transmission request to be cancelled, if it was initiated by the corresponding TRS*n* bit in the transmission request set register (CANTRS) and is not currently being processed. If the corresponding message is currently being processed, the TRR*n* bit is reset in the case of a successful transmission (normal operation) or an aborted transmission due to a lost arbitration or an error condition detected on the CAN bus line. In the case of an aborted transmission, the corresponding AA*n* bit in the abort acknowledge register (CANAA) is set; in the case of a successful transmission, the corresponding TA*n* bit in the transmission acknowledge register (CANTA) is set.

The TRR*n* bit is set by the CPU and reset by the internal logic. The TRR*n* bit is reset in the case of a successful transmission or an aborted transmission. If the CPU tries to set a bit while the CAN module tries to clear it, the bit is set.

If a transmission reset is requested for a transmit mailbox with the corresponding TRS*n* bit cleared in CANTRS, for a disabled mailbox, or for a receive mailbox, then the CAN module clears the TRR*n* bit and sets the AA*n* bit in order to respond to the request.

The bits in CANTRR are set by writing a 1 from the CPU; writing a 0 has no effect.

Figure 23-14. Transmission Request Reset Register (CANTRR) [0Ch]

31	16
TRR.31:16	
R/W-0[(1)]	

15	0
TRR.15:0	
R/W-0	

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

(1) HECC only, reserved in the SCC

Table 23-9. Transmission Request Reset Register (CANTRR) Field Descriptions

Bit	Field	Value	Description
31-0	TRR		Transmit Request Reset Register.
		0	No operation.
		1	Setting the TRR*n* bit causes a transmission request to be cancelled, if it was initiated by the corresponding TRS*n* bit in the transmission request set register (CANTRS) and is not currently being processed.

23.10.1.5 Transmission Acknowledge Register (CANTA)

If the message of mailbox *n* is transmitted successfully, the TA*n* bit is set in this register. A successful transmission also sets the GMIF0/GMIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1) and initiates a global mailbox interrupt, if the corresponding mailbox interrupt mask (MIM*n*) bit in the mailbox interrupt mask register (CANMIM) is set.

The CPU resets the bits in CANTA by writing a 1. Writing a 1 also clears the interrupt, if an interrupt had been generated; writing a 0, has no effect. If the CPU tries to reset the bit while the CAN module tries to set it, the bit is set. After power-up, all bits in CANTA are cleared to 0.

Figure 23-15. Transmission Acknowledge Register (CANTA) [10h]

31	16
TA.31:16	
R/WC-0[(1)]	

15	0
TA.15:0	
R/WC-0	

LEGEND: R/WC = Read/Write 1 to clear, write of 0 has no effect; R = Read only; -*n* = value after reset

(1) HECC only, reserved in the SCC

Table 23-10. Transmission Acknowledge Register (CANTA) Field Descriptions

Bit	Field	Value	Description
31-0	TA		Transmit Acknowledge Register.
		0	The message is not sent.
		1	If the message of mailbox *n* is sent successfully, bit *n* is set.

23.10.1.6 Abort Acknowledge Register (CANAA)

If the transmission of the message in mailbox *n* is aborted, the corresponding abort acknowledge (AA*n*) bit is set in this register. An aborted transmission also sets the AAIF0/AAIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1) and initiates an interrupt, if the corresponding AAIM bit in the global interrupt mask register (CANGIM) is set.

Additional Conditions That Set the AA *n* Bit

NOTE: The AA*n* bit is set if a transmission reset is requested and the TRSn bit in CANTRS of the corresponding transmit mailbox is not set, a receive mailbox or a disabled mailbox is concerned.

The CPU resets the bits in CANAA by writing a 1; writing a 0, has no effect. If the CPU tries to reset the bit while the CAN module tries to set it, the bit is set. After power-up, all bits in CANAA are cleared to 0.

Figure 23-16. Abort Acknowledge Register (CANAA) [14h]

31															16
AA.31:16															
R/WC-0[(1)]															
15															0
AA.15:0															
R/WC-0															

LEGEND: R/WC = Read/Write 1 to clear, write of 0 has no effect; R = Read only; -*n* = value after reset

(1) HECC only, reserved in the SCC

Table 23-11. Abort Acknowledge Register (CANAA) Field Descriptions

Bit	Field	Value	Description
31-0	AA		Abort Acknowledge Register.
		0	The transmission is not aborted.
		1	If the transmission of the message in mailbox *n* is aborted, bit *n* is set.

23.10.1.7 Receive Message Pending Register (CANRMP)

If mailbox *n* contains a received message, the RMP*n* bit in this register is set. The RMP*n* bit is only reset by the CPU and set by the internal logic. A new incoming message overwrites the stored message, if the OPC*n* bit in the overwrite protection control register (CANOPC) is cleared to 0; otherwise, the next mailboxes are checked for a matching identification. If a message is overwritten, the corresponding RML*n* bit in the receive message lost register (CANRML) is set. The bits in CANRMP and CANRML are cleared to 0 by a write access to the base address of CANRMP, with a 1 at the corresponding bit location. If the CPU tries to reset the bit while the CAN module tries to set it, the bit is set.

The proper sequence to clear RMP*n* bits is shown below.

1. Read the RMP*n* bits (don't clear it)
2. When RMP*n* is set, read the message
3. Clear the RMP*n* bit

A received message also sets the GMIF0/GMIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1) and initiates an interrupt, if the corresponding MIM*n* bit in the mailbox interrupt mask register (CANMIM) is set.

Figure 23-17. Receive Message Pending Register (CANRMP) [18h]

31	16
RMP.31:16	
R/WC-0[1]	

15	0
RMP.15:0	
R/WC-0	

LEGEND: R/WC = Read/Write 1 to clear, write of 0 has no effect; -*n* = value after reset

(1) HECC only, reserved in the SCC

Table 23-12. Receive Message Pending Register (CANRMP) Field Descriptions

Bit	Field	Value	Description
31-0	RMP		Receive Message Pending Register.
		0	The mailbox does not contain a message.
		1	If mailbox *n* contains a received message, bit *n* is set.

23.10.1.8 Receive Message Lost Register (CANRML)

If an old message is overwritten by a new message in message object *n*, the RML*n* bit in this register is set. The RML*n* bit is only reset by the CPU and set by the internal logic. The RML*n* bit is cleared by a write access to the base address of the receive message pending register (CANRMP), with a 1 at the corresponding bit location. If the CPU tries to reset the bit while the CAN module tries to set it, the bit is set. The content of CANRML is not changed, if the OPC*n* bit in the overwrite protection control register (CANOPC) is set.

If one or more of the RML*n* bits in CANRML are set, the RMLIF0/RMLIF1 bit in the global interrupt flag register (CANGIF0/CANGIF1) is set and initiates an interrupt, if the RMLIM bit in the global interrupt mask register (CANGIM) is set.

Figure 23-18. Receive Message Lost Register (CANRML) [1Ch]

31	16
RML.31:16	
R/WC-0[(1)]	

15	0
RML.15:0	
R/WC-0	

LEGEND: R/WC = Read/write 1 to clear, write of 0 has no effect; -*n* = value after reset

(1) HECC only, reserved in the SCC

Table 23-13. Receive Message Lost Register (CANRML) Field Descriptions

Bit	Field	Value	Description
31-0	RML		Receive Message Lost Register.
		0	No message was lost.
		1	An old unread message has been overwritten by a new one in that mailbox.

23.10.1.9 Remote Frame Pending Register (CANRFP)

Whenever a remote frame request is received by the CAN module, the corresponding RFP*n* bit in this register is set. If a remote frame is stored in a receive mailbox *n* (AAM = 0 in the message identifier register, CANMID, and MD*n* = 1 in the mailbox direction register, CANMD), the CPU has to initiate the reply frame transmission and has to reset the RFP*n* bit. If a mailbox configured in auto-answer mode (AAM = 1 and MD*n* = 0) receives a remote frame, the CAN module clears the RFP*n* bit after the reply frame has been successfully transmitted.

In order to prevent an auto-answer mailbox from replying to a remote frame request, the CPU clears the RFP*n* bit and the TRS*n* bit in the transmission request set register (CANTRS) by setting the corresponding TRR*n* bit in the transmission request reset register (CANTRR). The AAM bit may also be cleared by the CPU to stop the CAN module from sending the message.

If the CPU tries to reset the bit while the CAN module tries to set it, the bit is not set. The CPU cannot interrupt an on-going transfer.

Figure 23-19. Remote Frame Pending Register (CANRFP) [20h]

31	16
RFP.31:16	
R/WC-0[1]	

15	0
RFP.15:0	
R/WC-0	

LEGEND: R/WC = Read/write 1 to clear, write of 0 has no effect; -*n* = value after reset

(1) HECC only, reserved in the SCC

Table 23-14. Remote Frame Pending Register (CANRFP) Field Descriptions

Bit	Field	Value	Description
31-0	RFP		Remote Frame Pending Register
		0	No remote frame request was received.
		1	A remote frame request was received by the CAN module.

23.10.1.10 Global Acceptance Mask Register (CANGAM)

The global acceptance mask is used by the SCC or HECC in SCC-compatible mode. The global acceptance mask is used for mailboxes 6 to 15, if the AME bit in the message identifier register (CANMID) of the corresponding mailbox is set. A received message is only stored in the first mailbox with a matching identifier.

The global acceptance mask is used for mailboxes 6 to 15 of the SCC.

Figure 23-20. Global Acceptance Mask Register (CANGAM) [24h]

31	30 29	28 18	17 16
AMI	Reserved	GAM.28:18	GAM.17:16
R/W-0	R-0	R/W-0	R/W-0

15 0
GAM.15:0
R/W-0

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

Table 23-15. Global Acceptance Mask Register (CANGAM) Field Descriptions

Bit	Field	Value	Description
31	AMI		Acceptance Mask Identifier Extension bit.
		0	The identifier extension bit stored in the mailbox determines which messages shall be received.
		1	Standard and extended frames are received. In the case of an extended frame, all 29 bits of the identifier are stored in the mailbox and all 29 GAM bits are used for the filter. In the case of a standard frame, only the first 11 bits (28 to 18) of the identifier and the GAM bits are used.
30-29	Reserved		Reads are undefined and writes have no effect.
28-0	GAM		Global Acceptance Mask. These bits allow any identifier bits of an incoming message to be masked.
		0	Received identifier bit value must match the corresponding identifier bit in the message identifier register (CANMID).
		1	Accept a 0 or a 1 (don't care) for the corresponding bit of the received identifier.

23.10.1.11 Master Control Register (CANMC)

This register is used to control the settings of the CAN module.

Update Data With CDR Bit Mechanism

NOTE: When using the CDR bit to update the data field of a mailbox, the SCC and the HECC behave differently. The SCC always considers new data after an arbitration loss or bus error. The HECC tries to send the former data if the message was loaded into the internal transmit buffer prior to the data update. If the software wants to force the HECC to transmit updated data as soon as possible, it has to perform a dummy write to the transmit request set register (CANTRS), for example CANTRS = 0000 0000h, after the data update is finished (CDR bit is cleared).

Figure 23-21. Master Control Register (CANMC) [28h]

31							16
Reserved							
R/W-0							
15	14	13	12	11	10	9	8
LNTC	LNTM	SCM	CCR	PDR	DBO	WUBA	CDR
R/W-0[1]	R/W-x[1]	R/W-0[1]	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4				0
ABO	STM	SRES	MBNR				
R/W-0	R/W-0	R/W-0	R/W-0[1]				

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

(1) HECC only, reserved in the SCC

(1) HECC only, reserved in the SCC

Table 23-16. Master Control Register (CANMC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reads are undefined and writes have no effect.
15	LNTC		Local Network Time Clear Bit.
		0	Local network time register (CANLNT) is not reset.
		1	Local network time register (CANLNT) is reset to zero after a successful transmission or reception of mailbox 16.
14	LNTM		Local Network Time MSB Clear Bit.
		0	Local network time register (CANLNT) is not changed.
		1	The most-significant bit of the local network time register (CANLNT) is reset to zero. The LNTM bit is reset after one clock cycle by the internal logic.
13	SCM		SCC-Compatibility Mode.
		0	HECC operates in SCC-compatibility mode only mailboxes 15 to 0 can be used. You must refer to the SCC specification in order to operate the HECC in this mode. Since this mode is selected by default, all software developed for the SCC runs without any change to the HECC module.
		1	HECC operates in normal mode.
12	CCR		Change Configuration Request.
		0	The CPU requests normal operation. This can only be done after the bit-timing configuration register (CANBTC) is set to the allowed values.
		1	The CPU requests write access to the bit-timing configuration register (CANBTC) and the acceptance mask registers (CANGAM, CANLAM0, and CANLAM3) of the SCC. After setting this bit, the CPU must wait until the CCE bit in the error and status register (CANES) is 1 before proceeding to CANBTC (see Section 23.10.1.12). This could also mean that the CAN module is in the bus off state (see the BO bit in CANES, Section 23.10.1.13).
11	PDR		Power-Down-Mode Request.
		0	The CAN module power-down mode is not requested (normal operation).

Table 23-16. Master Control Register (CANMC) Field Descriptions (continued)

Bit	Field	Value	Description
		1	The CAN module power-down mode is requested.
10	DBO		Data Byte Order. This bit selects the byte order of the message data field of the message data registers, CANMDL and CANMDH (see Section 23.10.4.3).
		0	The data is received or transmitted most-significant byte first.
		1	The data is received or transmitted least-significant byte first.
9	WUBA		Wake Up on Bus Activity.
		0	The CAN module leaves the power-down mode only after writing a 0 to the PDR bit.
		1	The CAN module leaves the power-down mode after detecting any bus activity.
8	CDR		Change Data Request. This bit allows fast data message update.
		0	The CPU requests normal operation.
		1	The CPU requests write access to the data field of the mailbox specified by the MBNR bits. The CPU must clear the CDR bit after accessing the mailbox. The CAN module does not transmit the mailbox content while the CDR bit is set. This is checked by the state machine before and after it reads the data from the mailbox to store it in the transmit buffer.
7	ABO		Auto Bus On.
		0	Bus-offstate can only be left by clearing the CCR bit after 128 • 11 recessive bits.
		1	After bus-off state, the module goes back automatically into the bus-on state after 128 • 11 recessive bits.
6	STM		Self-Test Mode.
		0	The CAN module is in normal mode.
		1	The CAN module is in self-test mode. In this mode, the CAN module generates its own acknowledge (ACK) signal, thus enabling operation without a bus connected to the module. The message is not sent, but read back and stored in the appropriate mailbox.
5	SRES		Software Reset. This is a write-only bit and is always read as zero.
		0	No effect
		1	A write access to this register causes a software reset of the CAN module, that is, all parameters, except the bits listed in Table 23-17, are reset to their default values. The mailbox contents and the error counters are not modified. Pending and on-going transmissions are canceled without disturbing the communication.
4-0	MBNR	0-1Fh	Mailbox Number. The CPU requests write access to the data field of the mailbox specified by the MBNR bits. This field is used in conjunction with the CDR bit.

Table 23-17. Bits Not Changed After Software Reset

Register	Bit Number	HECC Bit Name
CANBTC	23-16	BRP
	10	ERM
	9-8	SJW
	7	SAM
	6-3	TSEG1
	2-0	TSEG2
CANMC	15	LNTC
	14	LNTM
	13	SCM
	12	CCR
	11	PDR
	10	DBO
	9	WUBA
	7	ABO
	6	STM
CANTIOC	3	TXFUNC

Table 23-17. Bits Not Changed After Software Reset (continued)

Register	Bit Number	HECC Bit Name
CANRIOC	3	RXFUNC

23.10.1.12 Bit-Timing Configuration Register (CANBTC)

This register is used to configure the CAN node with the appropriate network-timing parameters. This register must be programmed before using the CAN module. This register can only be written in initialization mode, see Section 23.6.1.

CAUTION

Unallowable Configuration Values
To avoid unpredictable behavior of the CAN module, do not program CANBTC with values not allowed by the CAN protocol specification and by the bit timing rules listed in Section 23.6.1.

Figure 23-22. Bit-Timing Configuration Register (CANBTC) [2Ch]

31	24	23	16
Reserved		BRP	
R-x		R/W-0	

15	11	10	9	8	7	6	3	2	0
Reserved		ERM	SJW		SAM	TSEG1		TSEG2	
R-x		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; -x = value is indeterminate after reset

Table 23-18. Bit-Timing Configuration Register (CANBTC) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved		Reads are undefined and writes have no effect.
23-16	BRP	0-FFh	Time Quantum Prescaler bits specify the duration of a time quantum (TQ) in CAN module system clock units. See Section 23.6.1.
			BRP PRESCALER is programmable from 1 to 256. The value programmed into CANBTC has to be decremented by 1, because of the 8-bit size of the BRP value:
			$BRP_{CANBTC.SJW}$ = BRP PRESCALER - 1
15-11	Reserved		Reads are undefined and writes have no effect.
10	ERM		Edge Resynchronization Mode bit selects the synchronization mode with the receive data stream on the CAN bus.
		0	The CAN module resynchronizes on the falling edge only.
		1	The CAN module resynchronizes on both rising and falling edges.
9-8	SJW	0-3h	Synchronization Jump Width bits indicate how many TQ units a bit is allowed to be lengthened or shortened when resynchronizing with the receive data stream on the CAN bus. The synchronization is performed either with the falling edge (ERM = 0) or with both edges (ERM = 1) of the bus signal.
			SJW is programmable from 1 to 4 TQ. The maximum value of SJW is determined by the minimum value of $TSEG2_{CALC}$ and 4TQ.
			$SJW_{CALC(max)} \leq \min[4TQ, TSEG2_{CALC}]$
			The value programmed into the CANBTC register has to be decremented by 1, because of the 2-bit size of the binary SJW value (SJW.1.0):
			$SJW_{BTC} = SJW_{CALC} - 1$
			SJW_{CALC} Result of the calculation.
			$1 \leq SJW_{CALC} \leq 4$
			SJW_{BTC} Value to be programmed into the CANBTC register.
			$0 \leq SJW_{BTC} \leq 3$
7	SAM		Sample point bits set the number of samples used by the CAN module to determine the actual level of the CAN bus. When the SAM bit is set, the level determined by the CAN bus corresponds to the result from the majority decision of the last three values. The sample points are at the sample point and twice before with a distance of ? TQ.
		0	The CAN module samples only once at the sampling point.

Table 23-18. Bit-Timing Configuration Register (CANBTC) Field Descriptions (continued)

Bit	Field	Value	Description
		1	The CAN module samples three times and makes a majority decision. The triple sample mode shall be selected only for bit rate prescale values greater than 4 (BRP > 4).
6-3	TSEG1	0-Fh	TSEG1 3:0 Time Segment 1
			This parameter specifies the length of the TSEG1 segment in TQ units. The value of TSEG1 is calculated as follows:
			TSEG1 = 1 + TSEG1.0 + (2*TSEG1.1) + (4*TSEG1.2) + (8*TSEG1.3)
			TSEG1 combines PROP_SEG and PHASE_SEG1 segments:
			TSEG1 = PROP_SEG + PHASE_SEG1
			Where PROP_SEG and PHASE_SEG1 are the length of these two segments in TQ units.
			$TSEG1_{CALC}$ value is programmable in the range of 1 TQ to 16 TQ and has to fulfill the following timing rule: TSEG1 must be greater than or equal to TSEG2 and IPT (for more details about IPT, refer to Section 23.6.1.
			The value programmed into the CANBTC register has to be decremented by 1 because of the 4-bit size of the binary TSEG1 value (TSEG1.3:0):
			$TSEG1_{BTC} = TSEG1_{CALC} - 1$
			$TSEG1_{CALC}$ Result of the calculation.
			$2 \leq TSEG1_{CALC} \leq 16$
			$TSEG1_{BTC}$ Value to be programmed into the CANBTC register.
			$1 \leq TSEG1_{BTC} \leq 15$
2-0	TSEG2	0-7h	TSEG2 2:0 Time Segment 2
			TSEG2 defines the length of the PHASE_SEG2 segment in TQ units, and is calculated as follows:
			TSEG2 = 1 + TSEG2.0 + (2*TSEG2.1) + (4 * TSEG2.2)
			$TSEG2_{CALC}$ is programmable in the range of 1 TQ to 8 TQ and has to fulfill the following timing rule: TSEG2 must be smaller than or equal to TSEG1 and must be greater than or equal to IPT. For more details about IPT, refer to Section 23.6.1.
			The value programmed into the CANBTC register has to be decremented by 1 because of the 3-bit size of the binary TSEG2 value (TSEG2 3:0):
			$TSEG2_{BTC} = TSEG2_{CALC} - 1$
			$TSEG2_{CALC}$ Result of the calculation.
			$TSEG2_{BTC}$ Value to be programmed into the CANBTC.

23.10.1.13 Error and Status Register (CANES)

The error and status register comprises information about the actual status of the CAN module and displays bus error flags as well as error status flags. The way of storing the bus error flags (FE, BE, CRCE, SE, and ACKE) and the error status flags (BO, EP, and EW) in this register is subject to a special mechanism. If one of these error flags is set, then the current state of all other error flags is frozen. In order to update this register to the actual values of the error flags, the error flag that is set has to be acknowledged by writing a 1 to it. This mechanism allows the software to distinguish between the first error and all following errors.

Figure 23-23. Error and Status Register (CANES) [30h]

31	25	24	23	22	21	20	19	18	17	16
Reserved		FE	BE	SA1	CRCE	SE	ACKE	BO	EP	EW
R-0		R/WC-0	R/WC-0	R/WC-1	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

15	6	5	4	3	2	1	0
Reserved		SMA	CCE	PDA	Reserved	RM	TM
R-0		R-0	R-1	R-0	R-0	R-0	R-0

LEGEND: R = Read only; R/WC = Read/write 1 to reset, write of 0 has no effect; -*n* = value after reset

Table 23-19. Error and Status Register (CANES) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved		Reads are undefined and writes have no effect.
24	FE		Form Error Flag. This bit is cleared when a 1 is written to this bit.
		0	No form error is detected.
		1	A form error occurred on the bus. This means that one or more of the fixed-form bit fields had the wrong level on the bus.
23	BE		Bit Error Flag. This bit is cleared when a 1 is written to this bit.
		0	No bit error is detected.
		1	The received bit does not match the transmitted bit outside of the arbitration field or during transmission of the arbitration field, a dominant bit was sent but a recessive bit was received.
22	SA1		Stuck at dominant error. This bit is always 1 after a hardware reset, a software reset, or a bus-off condition. This bit is cleared when a recessive bit is detected on the bus or when a 1 is written to this bit.
		0	The CAN module detected a recessive bit.
		1	The CAN module never detected a recessive bit.
21	CRCE		CRC Error. This bit is cleared when a 1 is written to this bit.
		0	The CAN module never received a wrong CRC.
		1	The CAN module received a wrong CRC.
20	SE		Stuff Error. This bit is cleared when a 1 is written to this bit.
		0	No stuff bit error occurred.
		1	A stuff bit error occurred.
19	ACKE		Acknowledge Error. This bit is cleared when a 1 is written to this bit.
		0	All messages have been correctly acknowledged.
		1	The CAN module received no acknowledge.
18	BO		Bus-Off State. The CAN module is in bus-offstate. This bit is cleared when a 1 is written to this bit.
		0	Normal operation
		1	There is an abnormal rate of errors on the CAN bus. This condition occurs when the transmit error counter register (CANTEC) reaches the limit of 256. During bus off, no messages are received or transmitted. The only ways to exit this state are by clearing the CCR bit in the master control register (CANMC) or by setting the ABO bit in CANMC. After leaving bus off, the error counters are cleared.
17	EP		Error Passive State. This bit is cleared when a 1 is written to this bit.

Table 23-19. Error and Status Register (CANES) Field Descriptions (continued)

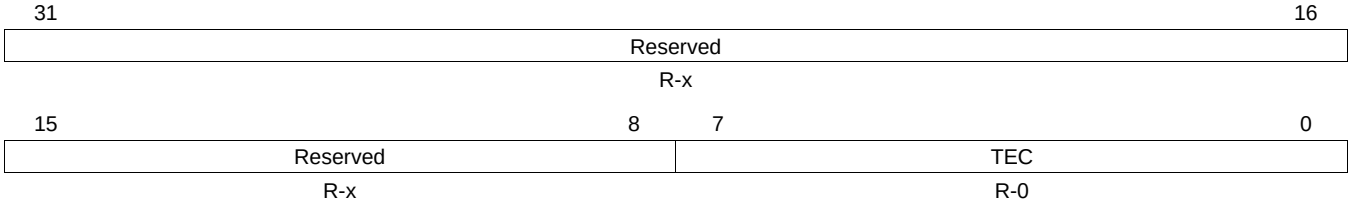
Bit	Field	Value	Description
		0	The CAN module is in error-active mode.
		1	The CAN module is in error-passive mode.
16	EW		Warning Status. This bit is cleared when a 1 is written to this bit.
		0	Both error counter registers (CANREC and CANTEC) are less than 96.
		1	One of the two error counter registers (CANREC or CANTEC) has reached the warning level of 96.
15-6	Reserved		Reads are undefined and writes have no effect.
5	SMA		Suspend Mode Acknowledge bit is set after a latency of one clock cycle—up to the length of one frame- after the suspend mode was activated. The suspend mode is activated with the debugger tool when the circuit is not in run mode. During the suspend mode, the CAN module is frozen and cannot receive or transmit any frame. However, if the CAN module is transmitting or receiving a frame when the suspend mode is activated, the module enters suspend mode only at the end of the frame.
		0	The CAN module is not in suspend mode.
		1	The CAN module has entered suspend mode.
4	CCE		Change Configuration Enable bit displays the configuration access. This bit is set after a latency of one clock cycle up to the length of one frame.
		0	The CPU is denied write access to the configuration registers.
		1	The CPU has write access to the configuration registers.
3	PDA		Power Down Mode Acknowledge.
		0	Normal operation
		1	The CAN module has entered the power-down mode.
2	Reserved		Reads are undefined and writes have no effect.
1	RM		CAN protocol kernel (CPK) Receive Mode bit reflects what the CPK is actually doing regardless of mailbox configuration.
		0	The CAN protocol kernel is not receiving a message.
		1	The CAN protocol kernel is receiving a message.
0	TM		CAN protocol kernel (CPK) Transmit Mode bit reflects what the CPK is actually doing regardless of mailbox configuration.
		0	The CAN protocol kernel is not transmitting a message.
		1	The CAN protocol kernel is transmitting a message.

23.10.1.14 Transmit Error Counter Register (CANTEC)

The CAN module contains two error counters: the receive error counter (CANREC) and the transmit error counter (CANTEC). The values of both counters are read by way of the CPU interface. These counters are incremented or decremented according to the CAN protocol specification version 2.0.

After reaching the bus-offstate, the transmit error counter is undefined while the receive error counter changes its function. After leaving initialization mode, the error counters are cleared.

Figure 23-24. Transmit Error Counter Register (CANTEC) [34h]



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

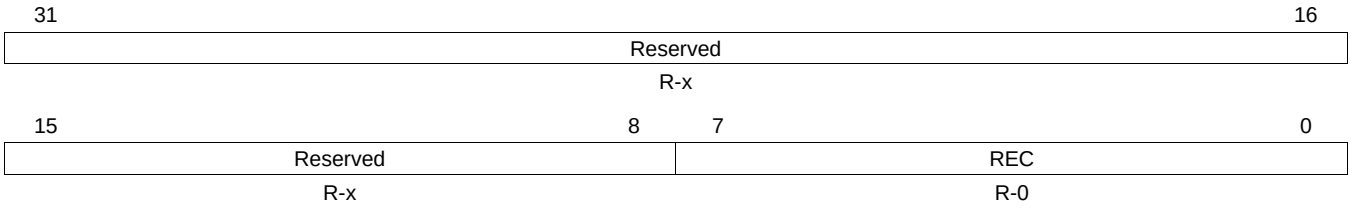
23.10.1.15 Receive Error Counter Register (CANREC)

The CAN module contains two error counters: the receive error counter (CANREC) and the transmit error counter (CANTEC). The values of both counters are read by way of the CPU interface. These counters are incremented or decremented according to the CAN protocol specification version 2.0.

After reaching or exceeding the error passive limit (128), the receive error counter is not incremented. When a message is received correctly, the counter is set again to a value between 119 and 127 (compare with CAN specification). After reaching the bus-offstate, the transmit error counter is undefined while the receive error counter changes its function.

After reaching the bus-off state, the receive error counter is cleared and is incremented after every 11 consecutive recessive bits on the bus. These 11 bits correspond to the gap between two frames on the bus. If the counter reaches 128, the CAN module automatically changes back to the bus-on status if this feature is enabled (ABO bit in the master control register, CANMC is set). All internal flags are reset and the error counters are cleared. After leaving initialization mode, the error counters are cleared.

Figure 23-25. Receive Error Counter Register (CANREC) [38h]



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

23.10.1.16 Global Interrupt Flag Registers (CANGIF0, CANGIF1)

These registers allow the CPU to identify the interrupt source.

Figure 23-26. Global Interrupt Flag 0 Register (CANGIF0) [3Ch]

31	18	17	16
Reserved		MAIF0	TCOIF0
R-x		R/WC-0[1]	R/WC-0[1]

15	14	13	12	11	10	9	8
GMIF0	AAIF0	WDIF0	WUIF0	RMLIF0	BOIF0	EPIF0	WLIF0
R-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

7	5	4	3	0
Reserved		MIV0.4	MIV0.3:0	
R-0		R-0[1]	R-0	

LEGEND: R = Read only; R/WC = Read/write 1 to clear, write of 0 has no effect; -*n* = value after reset; -x = value is indeterminate after reset

(1) HECC only, reserved in the SCC

(1) HECC only, reserved in the SCC

Figure 23-27. Global Interrupt Flag 1 Register (CANGIF1) [44h]

31	18	17	16
Reserved		MAIF1	TCOIF1
R-x		R/WC-0[1]	R/WC-0[1]

15	14	13	12	11	10	9	8
GMIF1	AAIF1	WDIF1	WUIF1	RMLIF1	BOIF1	EPIF1	WLIF1
R-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

7	5	4	3	0
Reserved		MIV1.4	MIV1.3:0	
R-0		R-0[1]	R-0	

LEGEND: R = Read only; R/WC = Read/write 1 to clear , write of 0 has no effect; -*n* = value after reset; -x = value is indeterminate after reset

(1) HECC only, reserved in the SCC

(1) HECC only, reserved in the SCC

Table 23-20. Global Interrupt Flag Registers (CANGIF) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved		Reads are undefined and writes have no effect.
17	MAIF		Message Alarm Flag.
		0	No time out for the mailboxes occurred.
		1	One of the mailboxes did not transmit or receive a message within the specified time frame.
16	TCOIF		Local Network Time Counter Overflow Flag.
		0	The MSB of the local network time register (CANLNT) is 0.
		1	The MSB of the local network time register (CANLNT) is 1.
15	GMIF		Global Mailbox Interrupt Flag
		0	No message has been transmitted or received.
		1	One of the mailboxes transmitted or received a message successfully.
14	AAIF		Abort-Acknowledge Interrupt Flag
		0	No transmission has been aborted.
		1	A transmission request has been aborted.

Table 23-20. Global Interrupt Flag Registers (CANGIF) Field Descriptions (continued)

Bit	Field	Value	Description
13	WDIF		Write-Denied Interrupt Flag
		0	The CPU write access to the mailbox was successful.
		1	The CPU write access to the mailbox was not successful.
12	WUIF		Wake-Up Interrupt Flag
		0	The CAN module is still in sleep mode or normal operation.
		1	During local power down, the CAN module has left sleep mode. During global power-down, there is activity on the CAN bus lines. This bit is also set when a global power-down is requested by the CPU, but the CAN module is not ready to enter power-down mode.
11	RMLIF		Receive-Message-Lost Interrupt Flag
		0	No message has been lost.
		1	At least for one of the receive mailboxes, an overflow condition has occurred.
10	BOIF		Bus-Off Interrupt Flag
		0	The CAN module is still in bus-on mode.
		1	The CAN module is in bus-off mode.
9	EPIF		Error-Passive Interrupt Flag
		0	The CAN module is not in error-passive mode.
		1	The CAN module is in error-passive mode.
8	WLIF		Warning Level Interrupt Flag
		0	None of the error counter registers (CANREC or CANTEC) has reached the warning level.
		1	At least one of the error counter registers (CANREC or CANTEC) has reached the warning level.
7-5	Reserved		Reads are undefined and writes have no effect.
4-0	MIV	0-1Fh	Message Interrupt Vector bits indicate the number of the message object that activated the global mailbox interrupt bit (GMIF0/GMIF1). These bits are held until the appropriate TA*n* bit in the transmission acknowledge register (CANTA) or RMP*n* bit in the receive message pending register (CANRMP) is cleared, or when a higher priority mailbox interrupt occurred. Then, the highest interrupt vector is displayed. In the HECC, mailbox 31 has the highest priority; in the SCC or SCC-compatible mode, mailbox 15 has the highest priority. When the GMIF0/GMIF1 bit is 0, the corresponding MIV0/MIV1 bits are undefined. MIV0.4 and MIV1.4 are not available in the SCC.

23.10.1.17 Global Interrupt Mask Register (CANGIM)

This register allows the various interrupts to be enabled. If a bit is set, the corresponding interrupt is enabled.

Figure 23-28. Global Interrupt Mask Register (CANGIM) [40h]

31							18	17	16
Reserved								MAIM	TCOIM
R-x								R/W-0[1]	R/W-0[1]

15	14	13	12	11	10	9	8
Reserved	AAIM	WDIM	WUIM	RMLIM	BOIM	EPIM	WLIM
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7				3	2	1	0
Reserved					SIL	I1EN	I0EN
R-0					R/W-0	R/W-0	R/W-0

LEGEND: R = Read only; R/W = Read/write; -*n* = value after reset; -x = value is indeterminate after reset

(1) HECC only, reserved in the SCC

Table 23-21. Global Interrupt Mask Register (CANGIM) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved		Reads are undefined and writes have no effect.
17	MAIM		Message Alarm Interrupt Mask.
		0	MAIF interrupt is disabled.
		1	MAIF interrupt is enabled.
16	TCOIM		Local Network Time Counter Overflow Interrupt Mask.
		0	TCOIF interrupt is disabled.
		1	TCOIF interrupt is enabled.
15	Reserved		Reads are undefined and writes have no effect.
14	AAIM		Abort Acknowledge Interrupt Mask
		0	AAIF interrupt is disabled.
		1	AAIF interrupt is enabled.
13	WDIM		Write Denied Interrupt Mask
		0	WDIF interrupt is disabled.
		1	WDIF interrupt is enabled.
12	WUIM		Wake-Up Interrupt Mask
		0	WUIF interrupt is disabled.
		1	WUIF interrupt is enabled.
11	RMLIM		Receive Message Lost Interrupt Mask
		0	RMLIF interrupt is disabled.
		1	RMLIF interrupt is enabled.
10	BOIM		Bus Off Interrupt Mask
		0	BOIF interrupt is disabled.
		1	BOIF interrupt is enabled.
9	EPIM		Error Passive Interrupt Mask
		0	EPIF interrupt is disabled.
		1	EPIF interrupt is enabled.
8	WLIM		Warning Level Interrupt Mask
		0	WLIF interrupt is disabled.
		1	WLIF interrupt is enabled.
7-3	Reserved		Reads are undefined and writes have no effect.

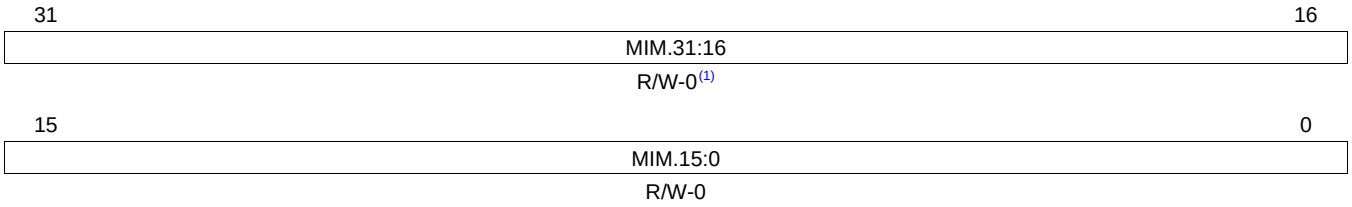
Table 23-21. Global Interrupt Mask Register (CANGIM) Field Descriptions (continued)

Bit	Field	Value	Description
2	SIL		System Interrupt Level. Define the interrupt level for TCOIF, WDIF, WUIF, BOIF, EPIF, AAIF, RMLIF and WLIF.
		0	All global interrupts are mapped to the HECC0INT/SCC0INT interrupt line.
		1	All system interrupts are mapped to the HECC1INT/SCC1INT interrupt line.
1	I1EN		Interrupt Line 1 Enable
		0	The HECC1INT/SCC1INT interrupt line is globally disabled.
		1	Globally enables all interrupts for the HECC1INT/SCC1INT interrupt line, if the corresponding masks are set.
0	I0EN		Interrupt Line 0 Enable
		0	The HECC0INT/SCC0INT interrupt line is globally disabled.
		1	Globally enables all interrupts for the HECC0INT/SCC0INT interrupt line, if the corresponding masks are set.

23.10.1.18 Mailbox Interrupt Mask Register (CANMIM)

There is one interrupt bit available for each mailbox. This may be a receive or a transmit interrupt depending on the configuration register. After power up, all interrupt mask bits are cleared and all interrupts are disabled.

Figure 23-29. Mailbox Interrupt Mask Register (CANMIM) [48h]



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

(1) HECC only, reserved in the SCC

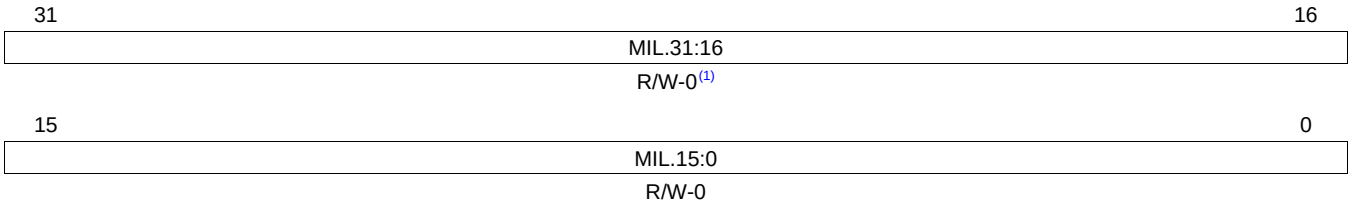
Table 23-22. Mailbox Interrupt Mask Register (CANMIM) Field Descriptions

Bit	Field	Value	Description
31-0	MIM		Mailbox Interrupt Mask bits allow any mailbox interrupt to be masked individually.
		0	Mailbox interrupt is disabled.
		1	Mailbox interrupt is enabled. An interrupt is generated if a message has been transmitted successfully (in case of a transmit mailbox) or if a message has been received without any error (in case of a receive mailbox).

23.10.1.19 Mailbox Interrupt Level Register (CANMIL)

Each mailbox may initiate an interrupt on one of the two interrupt lines. Depending on the setting in this register, the interrupt is generated on the HECC0INT/SCC0INT interrupt line (MIL*n* = 0) or on the HECC1INT/SCC1INT interrupt line (MIL*n* = 1).

Figure 23-30. Mailbox Interrupt Level Register (CANMIL) [4Ch]



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

(1) HECC only, reserved in the SCC

Table 23-23. Mailbox Interrupt Level Register (CANMIL) Field Descriptions

Bit	Field	Value	Description
31-0	MIL		Mailbox Interrupt Level bits allow any mailbox interrupt level to be selected individually.
		0	The mailbox interrupt is generated on HECC0INT/SCC0INT interrupt line.
		1	The mailbox interrupt is generated on HECC1INT/SCC1INT interrupt line.

23.10.1.20 Overwrite Protection Control Register (CANOPC)

If there is an overflow condition for mailbox *n,* the RMP*n* bit in the receive message pending register (CANRMP) is set to 1 and a new receive message would fit for mailbox *n*, the new message is stored depending on the settings in CANOPC.

- If the corresponding OPC*n* bit is set to 1, the old message is protected against being overwritten by the new message; thus, the next mailboxes are checked for a matching identification. If no other mailbox is found, the message is lost without further notification.
- If the corresponding OPC*n* bit is cleared to 0, the old message is overwritten by the new message.

An overwrite condition is monitored by the setting of the corresponding RML*n* bit in the receive message lost register (CANRML).

Figure 23-31. Overwrite Protection Control Register (CANOPC) [50h]

31	16
OPC.31:16	
R/W-0[(1)]	

15	0
OPC.15:0	
R/W-0	

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

(1) HECC only, reserved in the SCC

Table 23-24. Overwrite Protection Control Register (CANOPC) Field Descriptions

Bit	Field	Value	Description
31-0	OPC		Overwrite Protection Control
		0	The old message may be overwritten by a new message
		1	An old message stored in that mailbox is protected against being overwritten by the new message.

23.10.1.21 Transmit I/O Control Register (CANTIOC)

The CANTX pin may be configured as a normal I/O pin, if the HECC/SCC is not used, using CANTIOC.

CANTX Pin

NOTE: When the CANTX pin is configured for CAN transmit functions (TXFUNC = 1), the TXDIR and TXOUT bits have no meaning and can be set to either 0 or 1. The TXIN bit always shows the actual value of the CANTX pin.

Figure 23-32. Transmit I/O Control Register (CANTIOC) [54h]

31															16
Reserved															
R-x															
15							8	7			4	3	2	1	0
Reserved								Reserved				TXFUNC	TXDIR	TXOUT	TXIN
R-x								R-0				R/W-0	R/W-0	R/W-0	R-x

LEGEND: R = Read only; R/W = Read/write; -*n* = value after reset; -x = value is indeterminate after reset

Table 23-25. Transmit I/O Control Register (CANTIOC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reads are undefined and writes have no effect.
3	TXFUNC		External Pin I/O Enable
		0	The CANTX pin is used as an I/O pin, the CAN protocol kernel (CPK) output is unconnected.
		1	The CANTX pin is used for the CAN transmit functions.
2	TXDIR		Transmit Pin Direction
		0	The CANTX pin is used as an input pin, if CANTX is configured as an I/O pin (TXFUNC = 0).
		1	The CANTX pin is used as an output pin, if CANTX is configured as an I/O pin (TXFUNC = 0).
1	TXOUT		Output value for CANTX pin. This value is output on the CANTX pin, if CANTX is configured as an I/O pin (TXFUNC = 0) and as an output (TXDIR = 1).
0	TXIN		Reflects the value on the CANTX pin.
		0	Logic 0 present on CANTX pin.
		1	Logic 1 present on CANTX pin.

23.10.1.22 Receive I/O Control Registers (CANRIOC)

The CANRX pin may be configured as a normal I/O pin, if the HECC/SCC is not used, using CANRIOC.

CANRX Pin

NOTE: When the CANRX pin is configured for CAN receive functions (RXFUNC = 1), the RXDIR and RXOUT bits have no meaning and can be set to either 0 or 1. The RXIN bit always shows the actual value of the CANRX pin.

Figure 23-33. Receive I/O Control Register (CANRIOC) [58h]

31	16
Reserved	
R-x	

15	8	7	4	3	2	1	0
Reserved		Reserved		RXFUNC	RXDIR	RXOUT	RXIN
R-x		R-0		R/W-0	R/W-0	R/W-0	R-x

LEGEND: R = Read only; R/W = Read/write; -*n* = value after reset; -x = value is indeterminate after reset

Table 23-26. Receive I/O Control Register (CANRIOC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reads are undefined and writes have no effect.
3	RXFUNC		External Pin I/O Enable
		0	The CANRX pin is used as an I/O pin. The RX signal is still internally connected to the CAN protocol kernel (CPK).
		1	The CANRX pin is used for the CAN receive functions.
2	RXDIR		Receive Pin Direction
		0	The CANRX pin is used as an input pin, if CANRX is configured as an I/O pin (RXFUNC = 0).
		1	The CANRX pin is used as an output pin, if CANRX is configured as an I/O pin (RXFUNC = 0).
1	RXOUT		Output value for CANRX pin. This value is output on the CANRX pin, if CANRX is configured as an I/O pin (RXFUNC = 0) and as an output (RXDIR = 1).
0	RXIN		Reflects the value on the CANRX pin.
		0	Logic 0 present on CANRX pin.
		1	Logic 1 present on CANRX pin.

23.10.2 Time Stamp Registers

Time Stamp Registers

NOTE: The time stamp registers are available on the HECC only. The offset addresses are reserved in the SCC.

23.10.2.1 Local Network Time Register (CANLNT)

Figure 23-34. Local Network Time Register (CANLNT)

31	16
LNT.31:16	
R/W-0	
15	0
LNT.15:0	
R/W-0	

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

Table 23-27. Local Network Time Register (CANLNT) Field Descriptions

Bit	Field	Value	Description
31-0	LNT	0-FFFF FFFFh	Local Network Time Register. Value of the local network time counter (LNT) used for the time-stamp and the time-out functions.

23.10.2.2 Message Object Time Stamp Registers (CANMOTS)

Figure 23-35. Message Object Time Stamp Register (CANMOTS) [100h]

31	16
MOTS.31:16	
R/W-x	
15	0
MOTS.15:0	
RW-x	

LEGEND: R/W = Read/write; -*n* = value after reset; -x = value is indeterminate after reset

Table 23-28. Message Object Time Stamp Register (CANMOTS) Field Descriptions

Bit	Field	Value	Description
31-0	MOTS	0-FFFF FFFFh	Message Object Time Stamp Register. Value of the local network time counter (LNT) when the message has been actually received or transmitted.

23.10.3 Time-Out Registers

Time-Out Registers

NOTE: The time-out registers are available on the HECC only. The offset addresses are reserved in the SCC.

23.10.3.1 Message Object Time-Out Registers (CANMOTO)

Figure 23-36. Message Object Time-Out Registers (CANMOTO) [180h]

31	16
MOTO.31:16	
R/W-x	

15	0
MOTO.15:0	
R/W-x	

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

Table 23-29. Message Object Time-Out Registers (CANMOTO) Field Descriptions

Bit	Field	Value	Description
31-0	MOTO	0-FFFF FFFFh	Message Object Time-Out Register. Limit value of the local network time counter (LNT) to actually transmit or receive the message.

23.10.3.2 Time-Out Control Register (CANTOC)

Figure 23-37. Time-Out Control Register (CANTOC) [60h]

31	16
TOC.31:16	
R/W-0	

15	0
TOC.15:0	
R/W-0	

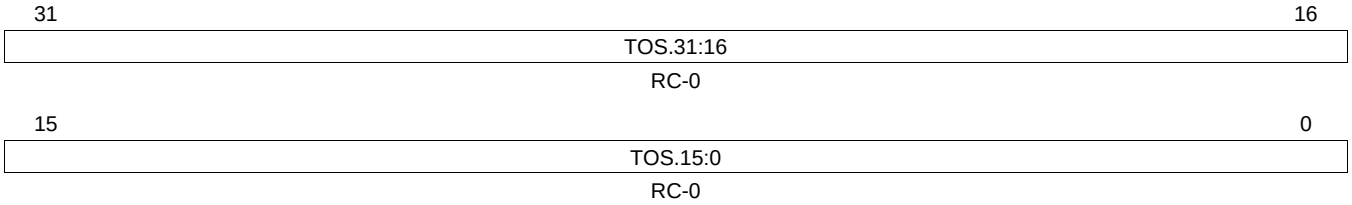
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23-30. Time-Out Control Register (CANTOC) Field Descriptions

Bit	Field	Value	Description
31-0	TOC		Time-Out Control Register.
		0	The time-out function is disabled. The TOS*n* bit is never set in the time-out status register (CANTOS).
		1	The TOC*n* bit has to be set by the CPU in order to enable the time-out function for mailbox *n*. Before setting the TOC*n* bit, load the corresponding message object time-out register (CANMOTO) with the time-out value relative to the local network time counter (LNT).

23.10.3.3 Time-Out Status Register (CANTOS)

Figure 23-38. Time-Out Status Register (CANTOS) [64h]



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

Table 23-31. Time-Out Status Register (CANTOS) Field Descriptions

Bit	Field	Value	Description
31-0	TOS		Time-Out Status Register.
		0	No time-out occurred or a specified mailbox is disabled.
		1	The value in the local network time register (CANLNT) is larger or equal to the value in the message object time-out register (CANMOTO) that corresponds to mailbox *n* and the TOC*n* bit is set in the time-out control register (CANTOC).

23.10.4 Message Mailbox Registers

23.10.4.1 Message Identifier Register (CANMID)

This register is only written if mailbox *n* is disabled (the ME*n* bit in the mailbox enable register, CANME, is cleared to 0).

Figure 23-39. Message Identifier Register (CANMID) [00h]

31	30	29	28											18	17	16
IDE	AME	AAM	ID.28:18												ID.17:16	
R/W-x	R/W-x	R/W-x	R/W-x												R/W-x	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ID.15:0																
R/W-x																

LEGEND: R/W = Read any time, write when mailbox is disabled or configured for transmission; -*n* = value after reset; -x = value is indeterminate after reset

Table 23-32. Message Identifier Register (CANMID) Field Descriptions

Bit	Field	Value	Description
31	IDE		Identifier Extension Bit
		0	Standard identifier (11 bits). The received message or the message to be sent has a standard identifier.
		1	Extended identifier (29 bits). The received message or the message to be sent has an extended identifier.
30	AME		Acceptance Mask Enable Bit. AME is only used for receiver mailboxes. It must not be set for automatic reply (AAM*n*=1, MD*n*=0) mailboxes; otherwise, the mailbox behavior is undefined. This bit is not modified by a message reception.
		0	No acceptance mask will be used, all identifier bits must match to receive the message.
		1	The corresponding acceptance mask is used.
29	AAM		Auto Answer Mode Bit. This bit is only valid for message mailboxes configured as transmit. For receive mailboxes, this bit has no effect: the mailbox is always configured for normal receive operation. This bit is not modified by a message reception.
		0	Normal transmit mode. The mailbox does not reply to remote requests. The reception of a remote request frame has no effect on the message mailbox.
		1	Auto answer mode. If a matching remote request is received, the CAN module answers to the remote request by sending the contents of the mailbox.
28-0	ID		Message Identifier.
			In standard identifier mode, if the IDE bit = 0, the message identifier is stored in ID bits 28-18. In this case, ID bits 17-0 have no meaning.
			In extended identifier mode, if the IDE bit = 1, the message identifier is stored in ID bits 28-0.

23.10.4.2 Message Control Field Register (CANMCF)

This register is only written if mailbox *n* is configured for transmission (MD*n* bit in the mailbox direction register, CANMD, is cleared to 0) or if the mailbox is disabled (ME*n* bit in the mailbox enable register, CANME, is cleared to 0).

Figure 23-40. Message Control Field Register (CANMCF) [04h]

31	16
Reserved	
R-0	

15	14	13	8	7	5	4	3	0
Reserved		TPL		Reserved		RTR	DLC	
R-0		R/W-x[1]		R-0		R/W-x	R/W-x	

LEGEND: R/W = Read at any time, write when the mailbox is disabled or configured in transmission; -*n* = value after reset; -x = value is indeterminate after reset

(1) HECC only, reserved in the SCC

Table 23-33. Message Control Field Register (CANMCF) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved		Reserved
13-8	TPL	0-3Fh	Transmit Priority Level. This 6-bit field in the HECC (reserved in the SCC) defines the priority of this mailbox as compared to the other 31 mailboxes. The highest number has the highest priority. In the case that two mailboxes have the same priority, the one with the higher mailbox number is transmitted. TPL applies only for transmit mailboxes. TPL is not used when in SCC-compatibility mode.
7-5	Reserved		Reserved
4	RTR		Remote Transmission Request Bit.
		0	No remote frame is requested.
		1	For receive mailbox: If the TRS*n* bit in the transmit request set register (CANTRS) is set, a remote frame is transmitted and the corresponding data frame will be received in the same mailbox. For transmit mailbox: If the TRS*n* bit in CANTRS is set, a remote frame is transmitted, but the corresponding data frame has to be received in another mailbox.
3-0	DLC	0-8h	Data Length Code. The number in these bits determines how many data bytes are sent or received. Valid value range is from 0 to 8. Values from 9 to 15 are not allowed.

23.10.4.3 Message Data Registers (CANMDL, CANMDH)

Eight bytes of the mailbox are used to store the data field of a CAN message. The setting of the DBO bit in the master control register (CANMC) determines the ordering of stored data. The data is transmitted or received from the CAN bus, starting with byte 0.

- When DBO = 1, the data is stored or read starting with the least-significant byte of CANMDL and ending with the most-significant byte of CANMDH.
- When DBO = 0, the data is stored or read starting with the most-significant byte of CANMDL and ending with the least-significant byte of CANMDH.

CANMDL*n* and CANMDH*n* are only written if mailbox *n* is configured for transmission (MD*n* bit in the mailbox direction register, CANMD, is cleared to 0) or if the mailbox is disabled (ME*n* bit in the mailbox enable register, CANME, is cleared to 0). If the TRS*n* bit in the transmission request set register (CANTRS) is set to 1, CANMDL*n* and CANMDH*n* cannot be written, unless the CDR bit in CANMC is set, with the MBNR bits in CANMC set to *n.* These settings also apply for a message object configured in reply mode (AAM bit in the message identifier register, CANMID, is set to 1).

Data Field

NOTE: The data field beyond the valid received data is modified by any message reception and is indeterminate.

Figure 23-41. Message Data Low Register with DBO = 0 (CANMDL) [08h]

31 24	23 16	15 8	7 0
Byte 0	Byte 1	Byte 2	Byte 3
R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: r/W = Read at any time, write when the mailbox is disabled or configured in transmission; -*n* = value after reset; -x = value is indeterminate after reset

Figure 23-42. Message Data High Register with DBO = 0 (CANMDH) [0Ch]

31 24	23 16	15 8	7 0
Byte 4	Byte 5	Byte 6	Byte 7
R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: r/W = Read at any time, write when the mailbox is disabled or configured in transmission; -*n* = value after reset; -x = value is indeterminate after reset

Figure 23-43. Message Data Low Register with DBO = 1 (CANMDL) [08h]

31 24	23 16	15 8	7 0
Byte 3	Byte 2	Byte 1	Byte 0
R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: r/W = Read at any time, write when the mailbox is disabled or configured in transmission; -*n* = value after reset; -x = value is indeterminate after reset

Figure 23-44. Message Data High Register with DBO = 1 (CANMDH) [0Ch]

31 24	23 16	15 8	7 0
Byte 7	Byte 6	Byte 5	Byte 4
R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: r/W = Read at any time, write when the mailbox is disabled or configured in transmission; -*n* = value after reset; -x = value is indeterminate after reset

23.10.4.4 Local Acceptance Mask Register (CANLAM)

The local acceptance filtering allows you to locally mask (don't care) any identifier bits of the incoming message.

In the SCC, the local acceptance mask register 0 (CANLAM0) is used for mailboxes 2 to 0. The local acceptance mask register 3 (CANLAM3) is used for mailboxes 5 to 3. For mailboxes 6 to 15, the global acceptance mask register (CANGAM) is used. CANLAM0 is located at address 80h of the SCC register memory; CANLAM3 is located at address 8Ch of SCC register memory.

After a hardware or a software reset of the SCC module, CANLAM0 and CANLAM3 are reset to zero. After a reset of the HECC, the CANLAM registers are not modified.

In the HECC, each mailbox (0 to 31) has its own mask register, CANLAM0 to CANLAM31. An incoming message is stored in the highest numbered mailbox with a matching identifier.

Figure 23-45. Local Acceptance Mask Register (CANLAM) [3000h]

31	30 29	28 18	17 16
LAMI	Reserved	LAM.28:18	LAM.17:16
R/W-x		R/W-x	R/W-x

15 0
LAM.15:0
R/W-x

LEGEND: R/W = Read/write; -*n* = value after reset; -x = value is indeterminate after reset for HECC, 0 for SCC

Table 23-34. Local Acceptance Mask Register (CANLAM) Field Descriptions

Bit	Field	Value	Description
31	LAMI		Local Acceptance Mask Identifier Extension Bit.
		0	The identifier extension bit stored in the mailbox determines which messages shall be received.
		1	Standard and extended frames can be received. In case of an extended frame, all 29 bits of the identifier are stored in the mailbox and all 29 bits of the local acceptance mask register are used for the filter. In case of a standard frame, only the first eleven bits (bits 28 to 18) of the identifier and the local acceptance mask are used.
30-29	Reserved		Reads are undefined and writes have no effect.
28-0	LAM		Local Acceptance Mask. These bits enable the masking of any identifier bit of an incoming message.
		0	Received identifier bit value must match the corresponding identifier bit of the message identifier register (CANMID).
		1	Accept a 0 or a 1 (don't care) for the corresponding bit of the received identifier.

Applications Processor Initialization

This chapter introduces the general-purpose device initialization.

Topic **Page**

24.1 Initialization Overview

This chapter provides an overview of the requirements for initializing the processor from power-on to firmware execution. An overview of the overall initialization process is given, including hardware- and software-related steps, a general overview of the boot ROM code operational requirements, and behavior expectations.

NOTE: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *Device Family* section, and your device-specific data manual.

24.1.1 Terminology

- Bootstrap: Initial software (SW) that is launched by the ROM code during the memory booting phase.
- Certificate: Data block plus trusted signature of the data block
- Downloaded SW: Initial SW that is downloaded into the internal SRAM by the ROM code during the peripheral booting phase
- eFuse: A one-time programmable memory location usually set at the factory
- Flash loader: Downloaded SW launched by the ROM code in preflashing. It also programs an image into external memories.
- GP device: General-purpose device. A type of device in which the security features, including the cryptographic HWA, are disabled. Intended for production.
- Initial SW: Software that is executed by any of the ROM code mechanisms (memory booting or peripheral booting). Initial SW is a generic term for bootstrap and downloaded SW.
- Memory booting: ROM code mechanism that consists of executing an Initial SW from external memory
- Peripheral booting: ROM code mechanism that consists of polling selected interfaces, downloading, and executing an Initial SW (in this, case called downloaded SW) in the internal RAM.
- Permanent booting device: Memory device containing, by default, the image to be executed during the booting sequence. It is the default memory booting device. The permanent booting device is used after warm reset.
- Preflashing: A specific case of peripheral booting where the ROM code mechanism is used to program the external flash memory
- R&D certificate: Certificate where the data block contains development configuration parameters
- ROM code: The on-chip SW in the processor's ROM which implements booting and security features
- Secure environment: Execution and storage environment, built with hardware and SW components, whose access is enabled when the processor runs in secure mode
- Secure mode: MPU secure working state, active when the ARM secure state is set, and reinforced by secure state machine (SSM) policy and attack countermeasures.
- Secure RAM: Writable area of the OCM-RAM. Protected applications are loaded for execution and secure data is temporarily stored in this area. Can only be accessed when the processor is in secure mode, or by a secure DMA channel.
- Secure ROM: Read-only memory inside the chip. Programmed at the chip manufacturing phase. Can be accessed only when the processor is in secure mode.

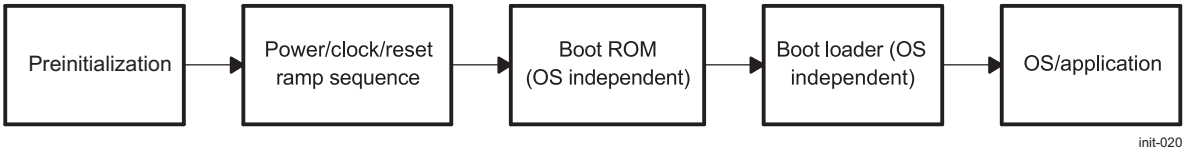
24.1.2 Initialization Process

A brief overview of the whole initialization process and its steps are shown in the Figure 24-1. Initialization consists of several steps:

- Preinitialization
- Power/clock/reset ramp sequence
- Boot ROM
- Boot loader
- OS/application

Each of these steps, up to OS/applications running, is explained in detail in the following sections.

Figure 24-1. Initialization Process



The first two steps in the initialization process are hardware oriented; however, they do require a good understanding of the process of configuring those system interface pins (balls on the device), which have SW configurable functionality. This configuration is an essential part of chip configuration and is application-dependent. This chapter refers to those pins and the associated configuration registers which are vital for proper device initialization.

See your device data manual for more information regarding the use of multiplexing configuration.

24.2 Preinitialization

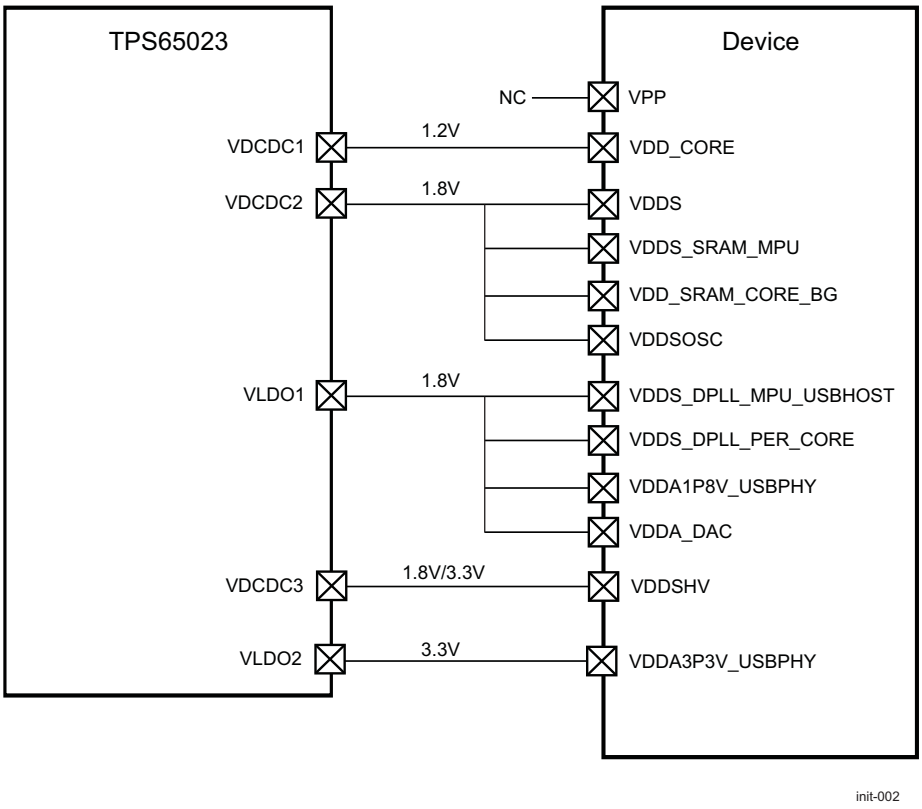
Certain hardware configuration settings must be made in order to accomplish a successful boot-up operation with a GP device. Clock, reset, and power connections, as well as pins involved in setting the boot memory space for the microprocessor unit (MPU), must be connected and driven properly for successful device initialization. The following sections detail the specific requirements for the preinitialization stage.

24.2.1 Power Connections

The device can be supplied by an external power IC. Texas Instruments provides a global solution with the processor connected to the TPS65023 power IC.

Figure 24-2 shows the power connections between the device and the TPS65023 power IC.

Figure 24-2. Power Connections



NOTE: Figure 24-2 is an example of power connections between the device and the TPS65023.

NOTE:

- The sys_clkreq output signal switches the system clock on or off.
- After power-on reset, the hardware configuration enables the internal oscillator (assuming the input clock comes from a crystal). The decision whether to bypass the internal oscillator is controlled by the polarity of the sys_boot[6] pin.

Table 24-1 describes of the power pins.

Table 24-1. Power Pin Descriptions

I/O Voltage Name	Description
VDD_CORE	Core power supply.
VDDS_SRAM_MPU	Analog power supply for MPU SLDO.
VDD_SRAM_CORE_BG	Analog power supply for CORE SLDO.
VDDSOSC	OSC VDDS supply.
VDDS_DPLL_MPU_USBHOST	Analog Power Supply for MPUSS/USBHOST DPLL.
VDDS_DPLL_PER_CORE	Analog supply for CORE/PER PLL & HSDIVIDER.
VDDA1P8V_USBPHY	1.8V supply for USB transceiver.
VDDA_DAC	Analog Power Supply for AVDAC.
VDDA3P3V_USBPHY	3.3V power supply for USB transceiver.
VDDSHV	1.8/3.3V IOs supply.
VDDS	Bias supply for IOs.
VPP	eFuse programming.

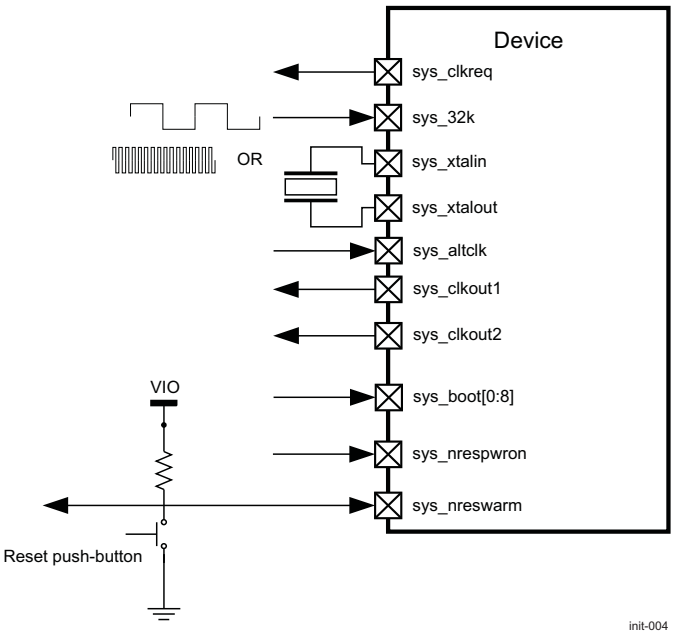
For a complete description of the power pins on the device package, see your device-specific data manual. For more information on power management, see the *Power, Reset, and Clock Management* chapter.

24.2.2 Clock and Reset

24.2.2.1 Clock and Reset Overview

Figure 24-3 shows the clock and reset environment which gathers the clocks and reset signals related at system level, as well as system expansion signals.

Figure 24-3. Clock and Reset Environment



The main features of the system interface are:

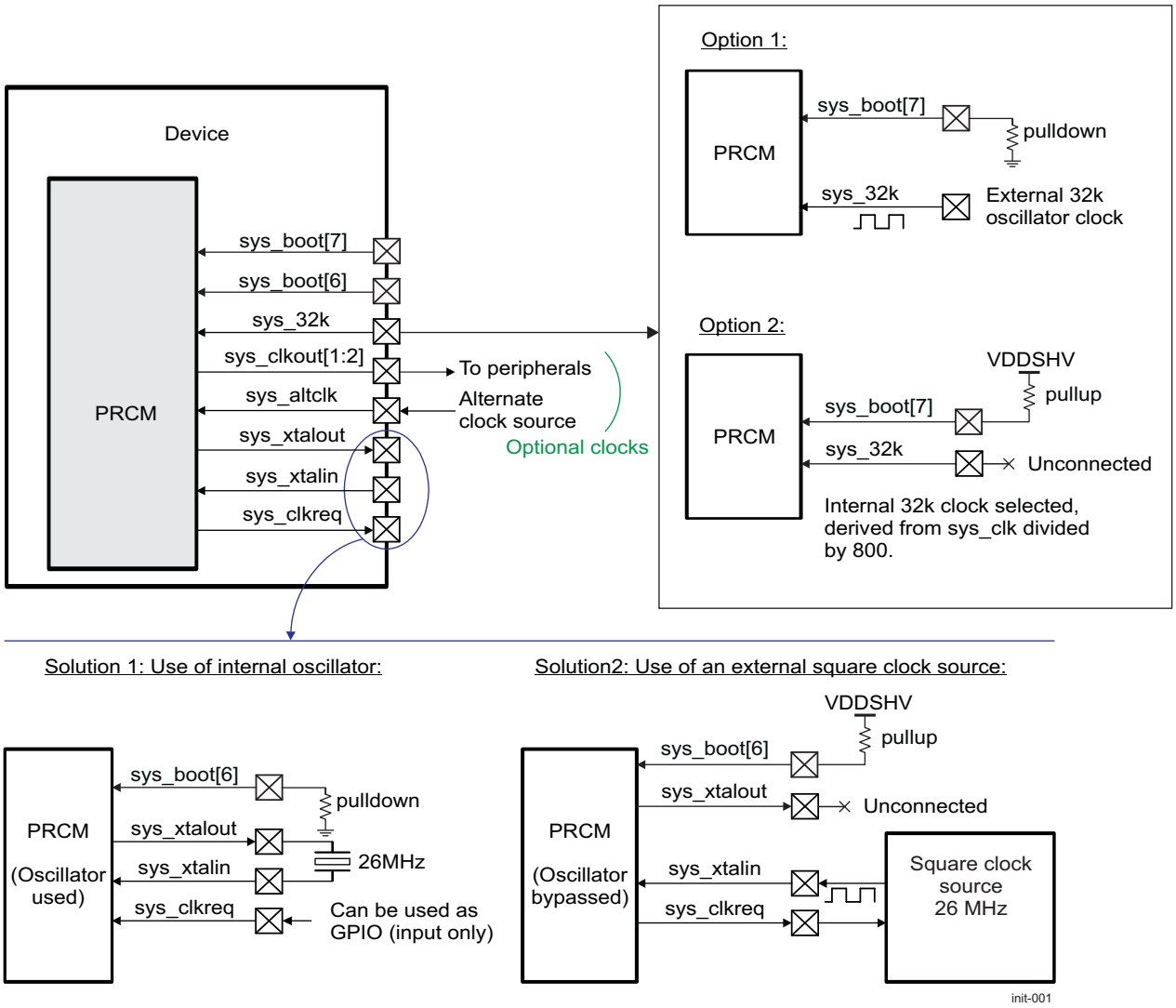
- A clock request output to an external square clock source
- 26MHz of system clock as reference clock input from either the external crystal oscillator or the digital clock input
- 32KHz clock for low frequency operation supplied either from external oscillator or generated internally
- An additional clock input up to 54 MHz
- Two configurable output clocks
- Nine input signals to define the boot mode
- Two reset sources
  - Power-on reset (cold reset)
  - Bidirectional warm reset

24.2.2.2 Clock Configuration

24.2.2.2.1 Required System Input Clocks

Figure 24-4 shows the clocks interface.

Figure 24-4. Clock Interface



NOTE:

- The SYS_CLKOUT output clock cannot be provided to peripherals when the core is off.
- The sys_clkreq output signal switches the system clock on or off.
- After power-on reset, the hardware configuration enables the internal oscillator (assuming the input clock comes from a crystal). The decision whether to bypass the internal oscillator is controlled by the polarity of the sys_boot[6] pin.
- Internal generation of 32K clock is enabled by providing pull-up on sys_boot[7] pin at power-on-reset.

The device operation requires two external input clocks as follows:

1. Sys_32k: The 32KHz clock for low frequency operation is supplied in following ways:
   - External – Supplied by an oscillator on sys_32K pin.
   - Internal – 32K clock generation using a fixed divider on HS system clock (26MHz).

2. Sys_xtal (in/out): The 26MHz system clock is the main clock source of the device. The system clock input can be connected in either of the two following ways:
   - Crystal quartz through the SYS_XTALIN and SYS_XTALOUT pins, using an internal oscillator.
   - CMOS digital clock (square clock) through the SYS_XTALIN pin. The oscillator is bypassed.

Table 24-2 lists the mapping for input sources.

Table 24-2. Mapping for Input Sources

Input Source	Mapping	Comment
Crystal quartz	SYS_XTALIN and SYS_XTALOUT	Requires use of internal oscillator
Square clock (1.8 CMOS signal)	SYS_XTALIN (SYS_XTALOUT unconnected)	Internal oscillator is bypassed

Only one source input at a time can be used because sys_32k, SYS_XTALIN and SYS_XTALOUT have permanently assigned pin locations with no pad configuration for these pins through a system control module register.

24.2.2.2.2 Optional System Input Clock: SYS_ALTCLK

An additional clock can be provided through the SYS_ALTCLK input pin to supply internal peripherals, phase-locked loops (PLLs), a precise clock for NTSC (54 MHz), USB full-speed controller (48 MHz). If not used as a clock input, this pin can be configured as a general-purpose input/output (GPIO), using the system control module CONTROL.CONTROL_PADCONF_I2C3_SDA register. As an example, to use the SYS_ALTCLK pin, the CONTROL.CONTROL_PADCONF_I2C3_SDA[18:16] MUXMODE1 field must be set to 0x01.

24.2.2.2.3 Optional System Output Clock: SYS_CLKOUT1 and SYS_CLKOUT2

Two output clocks (SYS_CLKOUT1 and SYS_CLKOUT2 pins) are available:

- SYS_CLKOUT1 can output the oscillator clock. Its OFF state polarity is programmable.
- SYS_CLKOUT2 can output the system clock (26MHz), the core clock (CORE DPLL output), 96 MHz, or 54 MHz. SYS_CLKOUT2 can be divided by 2, 4, 8, or 16, and its OFF state polarity is programmable.

CAUTION

Clock configurations depend on core voltage, and maximum clock frequencies may not be applicable to production. Please refer to your device-specific data manual for a description of the supported voltage levels and maximum clock frequencies.

The clocks can be managed by SW with the appropriate register in the power, reset, clock, management (PRCM) module:

- SYS_CLKOUT1 is managed using PRCM.PRM_CLKOUT_CTRL and PRCM.PRM_POLCTRL[2].
- SYS_CLKOUT2 is managed using PRCM.CM_CLKOUT_CTRL and PRCM.CM_CLKSEL1_PLL[28:27].

24.2.2.3 Reset Configuration

The sys_nrespwron reset pin is used to reset the entire chip during the power-on reset.

The sys_nreswarm reset pin is used to reset the entire chip when the device is supplied and operating, except for the following:

- Part of Memory Subsystem
- Part of PRCM

- Control module
- Watchdog
- 32-kHz synchronization timer
- DPLLs

CAUTION

sys_nrespwron must be driven low during a power-up sequence.

sys_nreswarm is a bidirectional reset. When an internal reset occurs, sys_nreswarm goes low and resets all the peripherals.

Both sys_nrespwron and sys_nreswarm have permanently assigned pin locations.

At reset, the cause of reset is stored in the PRCM.RM_RSTST_MPU register and used by the boot ROM code.

24.2.3 Boot Configuration

Six external pins (sys_boot[5:0]) are used to select interfaces or devices used for booting. The sys_boot[6] pin is used to select whether the internal oscillator is bypassed. The sys_boot[7] pin is used to select whether to use external 32K clock supply or 32K clock is generated internally.

Boot configuration pins sys_boot[5:0] are sampled and latched onto CONTROL.CONTROL_STATUS register and sys_boot[8:7] are sampled and latched onto CONTROL.CONTROL_DEVCONF2 register following a power-on reset. After booting, these pins can optionally be used for other functions and the associated CONTROL.CONTROL_STATUS bits are not updated by the new functionality. For more information about pad multiplexing configuration, see the *System Control Module* chapter.

NOTE: If used as GPIOs, these pins must be used in the output mode. To ensure the sys_boot[8:0] input values are selected by the pullup and pulldown on sys_boot[8:0] pins at power-on reset, these GPIOs must be used only in output mode. As these balls have no pullup or pulldown capacity, care must be taken when choosing to use these GPIOs.

Table 24-3, Table 24-4, and Table 24-5 are decoding tables for sys_boot pins. Depending on sys_boot pin configuration during the reset (first column), the ROM code tries to boot on the first device listed. If the boot failed on this first device, the ROM code tries the second device, then the third, then the fourth.

The following names are used in the tables:

- Memory types:
  - XIP: XIP memory without wait monitoring enabled (NOR flash memories)
  - XIP wait: XIP memory with wait monitoring
  - DOC: DiskOnChip™ memory (H3 device types)
  - NAND: NAND flash memories (non XIP)
  - OneNAND: OneNAND flash memories
  - MMC1: MMC or SD flash cards with active primary partition of type FAT12/16/32 connected to the first multimedia card/secure digital/secure digital I/O (MMC/SD/SDIO1) card interface
  - MMC2: MMC or SD flash cards with active primary partition of type FAT12/16/32 connected to the second multimedia card/secure digital/secure digital I/O (MMC/SD/SDIO2) card interface
- Peripheral interfaces:
  - Ethernet: EMAC RMII interface
  - USB: HS/FS USB
  - UART3: UART interface
- Permanent booting devices are in *italics*.

Table 24-3 and Table 24-4 list the sys_boot pin configuration after a power-on reset (POR).

CAUTION

- UART boot: UART3 is the only possible UART from which boot can be performed. Additionally, only UART3 pads that provide UART3 functionality in their MUXMODE 0 can be used. No other UART configuration allows booting from the UART.

Table 24-3. Memory Booting Configuration Pins after POR

sys_boot [4:0]	Booting Sequence When SYS.BOOT[5] = 0			
	Memory Booting Preferred Order			
	First	Second	Third	Fourth
0b00000	*OneNAND*	EMAC	USB	
0b00001	*NAND*	EMAC	USB	
0b00010	*OneNAND*	EMAC	USB	MMC1
0b00011	*MMC2*	EMAC	USB	MMC1
0b00100	*OneNAND*	USB		
0b00101	*MMC2*	USB		
0b00110	*MMC1*	USB		
0b00111	*XIP*	EMAC	USB	
0b01000	*XDOC*	EMAC	USB	
0b01001	*MMC2*	EMAC	USB	
0b01010	*XIP*	EMAC	USB	MMC1
0b01011	*XDOC*	EMAC	USB	MMC1
0b01100	*NAND*	EMAC	USB	MMC1
0b01101	*XIP*	USB	UART	MMC1
0b01110	*XDOC*	USB	UART	MMC1
0b01111	*NAND*	USB	UART	MMC1
0b10000	*OneNAND*	USB	UART	MMC1
0b10001	*MMC2*	USB	UART	MMC1
0b10010	*MMC1*	USB	UART	
0b10011	*XIP*	UART		
0b10100	*XDOC*	UART		
0b10101	*NAND*	UART		
0b10110	*OneNAND*	UART		
0b10111	*MMC2*	UART		
0b11000	*MMC1*	UART		
0b11001	*XIP*	USB		
0b11010	*XDOC*	USB		
0b11011	*NAND*	USB		
0b11100	*SPI*	UART		
0b11101			Reserved[1]	
0b11110				
0b11111	*Fast XIP booting wait monitoring OFF*	USB	UART3	

(1) Must not be selected

NOTE: SYS_BOOT[6] is used to bypass internal oscillator.
SYS_BOOT[7] is used to generate 32K clock internally.
SYS_BOOT[8] is used to select reduced pin mode.
For more details refer to System Control Module chapter.

Table 24-4. Peripheral Booting Configuration Pins after POR

sys_boot [4:0]	Booting Sequence When SYS.BOOT[5] = 1 Peripheral Booting Preferred Order			
	First	Second	Third	Fourth
0b00000	*EMAC*	USB	OneNAND	
0b00001	*EMAC*	USB	NAND	
0b00010	*EMAC*	USB	MMC1	OneNAND
0b00011	*EMAC*	USB	MMC1	MMC2
0b00100	*USB*	OneNAND		
0b00101	*USB*	MMC2		
0b00110	*USB*	MMC1		
0b00111	*EMAC*	USB	XIP	
0b01000	*EMAC*	USB	XDOC	
0b01001	*EMAC*	USB	MMC2	
0b01010	*EMAC*	USB	MMC1	XIP
0b01011	*EMAC*	USB	MMC1	XDOC
0b01100	*EMAC*	USB	MMC1	NAND
0b01101	*USB*	UART	MMC1	XIP
0b01110	*USB*	UART	MMC1	XDOC
0b01111	*USB*	UART	MMC1	NAND
0b10000	*USB*	UART	MMC1	OneNAND
0b10001	*USB*	UART	MMC1	MMC2
0b10010	*USB*	UART	MMC1	
0b10011	*UART*	XIP		
0b10100	*UART*	XDOC		
0b10101	*UART*	NAND		
0b10110	*UART*	OneNAND		
0b10111	*UART*	MMC2		
0b11000	*UART*	MMC1		
0b11001	*USB*	XIP		
0b11010	*USB*	XDOC		
0b11011	*USB*	NAND		
0b11100	*UART*	SPI		
0b11101			Reserved[1]	
0b11110				
0b11111	*Fast XIP booting wait monitoring*	USB	UART3	

(1) Must not be selected

NOTE: SYS_BOOT[6] is used to bypass internal oscillator. SYS_BOOT[7] is used to generate 32K clock internally. SYS_BOOT[8] is used to select reduced pin mode. For more details refer to System Control Module chapter.

Table 24-5 shows the sys_boot pin configuration after a warm reset.

Table 24-5. Booting Configuration Pins after a Warm Reset

sys_boot[4:0]	Booting Sequence When SYS.BOOT[5] = 0		Booting Sequence When SYS.BOOT[5] = 1	
	Memory Booting Preferred Order		Peripheral Booting Preferred Order	
	First	Second	First	Second
0b00000	*OneNAND*		OneNAND	
0b00001	*NAND*		NAND	
0b00010	*OneNAND*		OneNAND	
0b00011	*MMC2*		MMC2	
0b00100	*OneNAND*		*OneNAND*	
0b00101	*MMC2*		*MMC2*	
0b00110	*MMC1*		*MMC1*	
0b00111	*XIP*		XIP	
0b01000	*XIPwait*	*DOC*	XIPwait	*DOC*
0b01001	*MMC2*		MMC2	
0b01010	*XIP*		XIP	
0b01011	*XIPwait*	*DOC*	XIPwait	*DOC*
0b01100	*NAND*		NAND	
0b01101	*XIP*		USB	*XIP*
0b01110	*XIPwait*	*DOC*	*XIPwait*	*DOC*
0b01111	*NAND*		*NAND*	
0b10000	*OneNAND*		*OneNAND*	
0b10001	*MMC2*		*MMC2*	
0b10010	*MMC1*		*MMC1*	
0b10011	*XIP*		*XIP*	
0b10100	*XIPwait*	*DOC*	*XIPwait*	*DOC*
0b10101	*NAND*		*NAND*	
0b10110	*OneNAND*		*OneNAND*	
0b10111	*MMC2*		*MMC2*	
0b11000	*MMC1*		*MMC1*	
0b11001	*XIP*		*XIP*	
0b11010	*XIPwait*	*DOC*	*XIPwait*	*DOC*
0b11011	*NAND*		*NAND*	
0b11100	*SPI*		*SPI*	
0b11101	Reserved[1]			
0b11110				
0b11111	*ROM code fast XIP booting*		ROM code fast XIP booting	

(1) Must not be selected

24.3 Power, Clocks, and Reset Power-Up Sequence

See the section *Power-Up Sequence* in the *Power, Reset, and Clock Management* chapter for the power-on sequence, including power supplies, clocks, and reset signals.

See your device data manual for more information regarding the device power-up sequence.

24.4 Device Initialization by ROM Code

This section describes the high-level booting concepts and provides basic knowledge of booting on the device.

24.4.1 Booting Overview

CAUTION

To use the level 2 (L2) cache with the device, the ROM code provides three primitive services. These services are implemented in monitor mode and do not use any resources outside the MPU subsystem. The services are described below. To call a service, a register r12 must be set to service ID and the SMI instruction must be executed.

- r12=1: To use the L2 cache, all L2 line data must be invalidated through the CP15 registers. This service invalidates the entire L2 cache and must be performed after a POR or a loss of L2 cache after reset. This register can also be read.
- r12=2: This service writes the value of the central processing unit (CPU) register R0 in the L2 cache auxiliary control register. This register can also be read.
- r12=3: This service writes the value of the CPU register R0 in the auxiliary control. This register can also be read. For more information about ARM L2 cache and registers, see the *Cortex-A8 Technical Reference Manual*. For more information about ARM CP15 registers, see the *ARM Architecture Reference Manual*.

24.4.1.1 Booting Types

Booting is the process of starting a bootstrap from one of the booting memories.

The ROM code has two functionalities for booting: peripheral booting and memory booting.

- In peripheral booting, the ROM code polls a selected communication interface such as UART or USB, downloads the executable code over the interface, and executes it in internal RAM. Downloaded software from an external host can be used to program flash memories connected to the device. This special case of peripheral booting is called preflashing; software downloaded for preflashing is called the flash loader. The flash loader burns a new client application image in external flash memory. Initial software is a generic term for bootstrap, downloaded software, and flash loader. After the image is burned, a software reset can be performed.
- In memory booting, the ROM code finds the bootstrap in permanent memories such as flash memory or memory cards and executes it. This process is normally performed after cold or warm device reset.

The ROM code detects whether the device should download software from a peripheral interface (USB, UART3 or EMAC) by using the sys_boot pin configuration. This mechanism encompasses initial flashing in production (external memory is empty) and reflashing in service (external memory is already programmed).

Table 24-6 lists the pin multiplexing according to boot peripheral.

Table 24-6. Pin Multiplexing According to Boot Peripheral

Boot Device	Pins
NAND/OneNAND	General-purpose memory controller (GPMC) pins in mode 0
XIP memory	GPMC pins in mode 0
DiskOnChip	GPMC pins in mode 0
MMC/SD1	MMC1 pins in mode 0
MMC/SD2	MMC2 pins in mode 0
UART3	UART3 pins in mode 0
EMAC	EMAC pins in mode 0

CAUTION

- UART boot: UART3 is the only possible UART from which boot can be performed. Additionally, only UART3 pads that provide UART3 functionality in their MUXMODE 0 can be used. No other UART configuration allows booting from the UART.

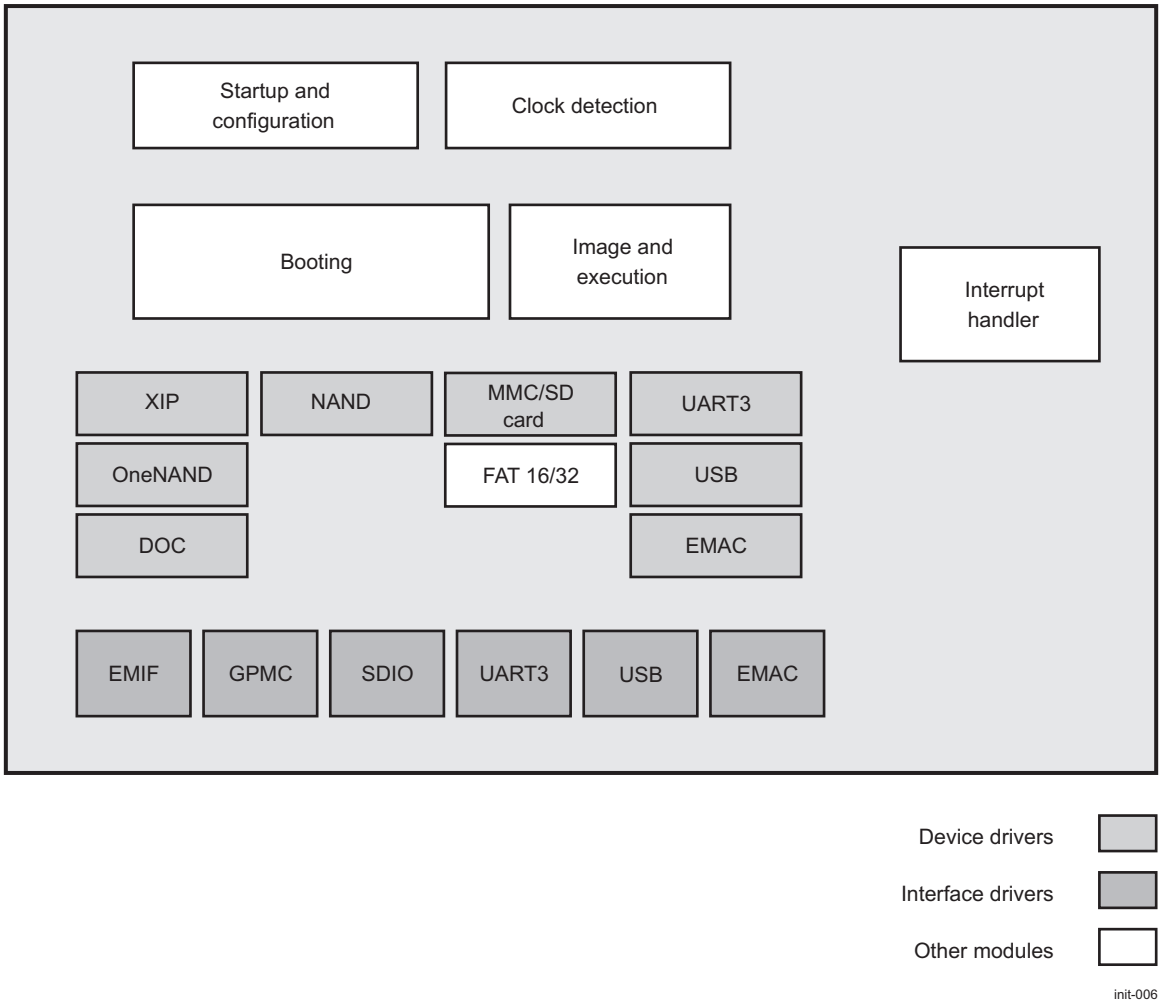
24.4.1.2 Main Features

The ROM code architecture is shown in Figure 24-5. The ROM code is made up of several modules:

- The start-up module takes care of basic system configuration and dispatches control into the booting module.
- The clock detection module supports start-up.
- The booting module uses several device drivers as it must boot from one of the external devices.
- The device drivers implement device protocols and perform logical operation on a device. They are abstracted from interface hardware by several interface drivers which are responsible for interacting with hardware interface facilities.
- The interrupt handler module provides services used among all modules. It allows registering interrupt service routines (ISRs) and calls them when an interrupt occurs.

Figure 24-5 shows each module.

Figure 24-5. ROM Code Architecture

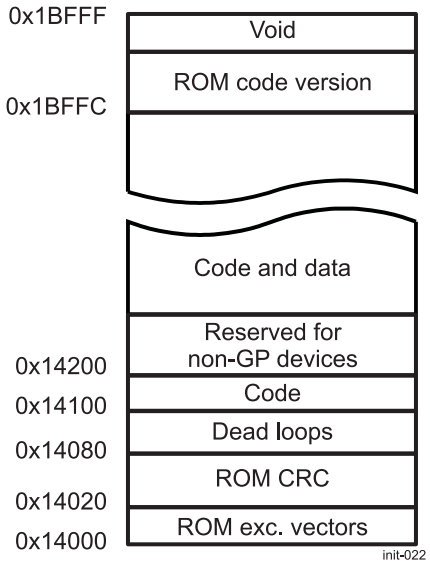


24.4.2 Memory Map

24.4.2.1 ROM Memory Map

Figure 24-6 shows the ROM memory map.

Figure 24-6. 32KB ROM Memory Map



- ROM exception vectors
  Exceptions are redirected to ROM exception vectors (see Table 24-7). The reset exception is redirected to the public ROM code startup. Other exceptions are redirected to RAM handlers by loading appropriate addresses in the PC register.

Table 24-7. ROM Exception Vectors

Address	Exception	Content
14000h	Reset	Branch to the public ROM code startup
14004h	Undefined	PC = 4020FFC8h
14008h	Software interrupt (SWI)	PC = 4020FFCCh
1400Ch	Prefetch abort	PC = 4020FFD0h
14010h	Data abort	PC = 4020FFD4h
14014h	Unused	PC = 4020FFD8h
14018h	IRQ	PC = 4020FFDCh
1401Ch	FIQ	PC = 4020FFE0h

- ROM code cyclic redundancy check (CRC)
  The ROM code CRC is calculated as 32-bit CRC code (CRC-32-IEEE 802.3) for the address range 14000h–1BFFFh. The 4-byte CRC code is stored at location 14020h, which is filled with FFFF FFFFh before the CRC is calculated.
- Dead loops
  Dead loops are branch instructions coded in ARM mode. They have multiple purposes (see Table 24-8).

Table 24-8. Dead Loops

Address	Purpose
14080h	Undefined exception default handler
14084h	SWI exception default handler
14088h	Prefetch abort exception default handler
1408Ch	Data abort exception default handler
14090h	Unused exception default handler
14094h	IRQ exception default handler
14098h	FIQ exception default handler
1409Ch	Validation tests Pass
140A0h	Validation tests Fail
140A4h	Booting failed: No more devices
140A8h	Image not executed or returned
140ACh	Reserved
140B0h	Reserved
140B4h	Reserved
140B8h	Reserved
140BCh	Reserved

The fixed location of these dead loops facilitates debugging and testing. The first seven dead loops are default exception handlers linked with RAM exception vectors.

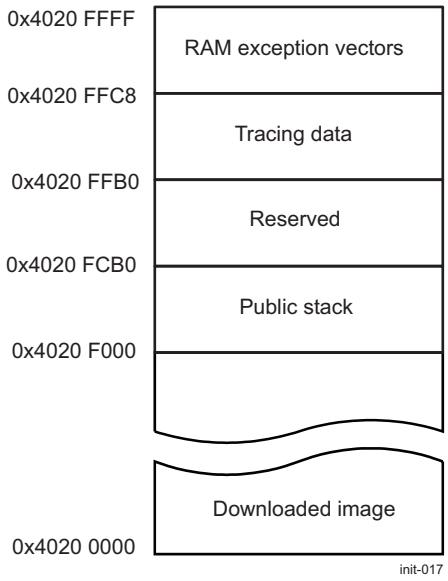
Dead loops can be called directly from code, but there is also a special function called from ROM code to execute a dead loop. This function is at address 140C0h. The function is assembly code in ARM mode, which takes the dead loop address from the R0 register. The main purpose of the function is to issue a global software reset before going to a dead loop. In addition, the function clears the global cold reset status before issuing the global software reset.

- Code
  This space is used to keep code.
- Code and data
  This space is used to keep code and other data.
- ROM code version
  The ROM code version consists of two decimal numbers: major and minor. The major number is always 14. The minor number identifies the ROM code version. The minor number is not aligned with the ROM code release number, but it can identify it. The ROM code version is coded as hexadecimal readable values (for example, ROM version 14.04 is coded as a 3-bit word: 00001404h).

24.4.2.2 RAM Memory Map

Figure 24-7 shows RAM memory map. The shown partitioning of the on-chip SRAM is only used during the booting process.

Figure 24-7. 64KB RAM Memory Map of GP Devices



- Downloaded image
  This space is used by the public ROM code to store a downloaded booting image.
- Public stack
  Space reserved for stacks.
- Tracing data
  The public ROM code tracing data is described in Table 24-9. More information about ROM code tracing can be found in Section 24.4.9, *Tracing*.

Table 24-9. Tracing Data

Address	Size [bytes]	Description
0x4020FFB0	4	Current tracing vector, word 1
0x4020FFB4	4	Current tracing vector, word 2
0x4020FFB8	4	Current copy of the PRM_RSTST register (reset reasons)
0x4020FFBC	4	Cold reset run tracing vector, word 1
0x4020FFC0	4	Cold reset run tracing vector, word 2
0x4020FFC4	4	Reserved

- RAM exception vectors

The RAM exception vectors provide an easy way to redirect exceptions to the custom handler. Table 24-10 shows the contents of the RAM space reserved for RAM vectors. The first seven addresses are ARM instructions which load into the PC the value located in the next seven addresses. Theses instructions are executed when an exception occurs, since they are called from ROM exception vectors. By default, all exceptions are redirected to the exception dead loops. Users can redirect an exception to other handler by writing its address to the appropriate position from 0x4020FFE4 to 0x4020FFFC, or by overriding instructions between addresses from 0x4020FFC8 to 0x4020FFE0.

Table 24-10. RAM Exception Vectors

Address	Exception	Content
0x4020FFC8	Undefined	PC = [0x4020FFE4]
0x4020FFCC	SWI	PC = [0x4020FFE8]
0x4020FFD0	Pre-fetch abort	PC = [0x4020FFEC]
0x4020FFD4	Data abort	PC = [0x4020FFF0]

Table 24-10. RAM Exception Vectors (continued)

Address	Exception	Content
0x4020FFD8	Unused	PC = [0x4020FFF4]
0x4020FFDC	IRQ	PC = [0x4020FFF8]
0x4020FFE0	FIQ	PC = [0x4020FFFC]
0x4020FFE4	Undefined	0x14080
0x4020FFE8	SWI	0x14084
0x4020FFEC	Pre-fetch abort	0x14088
0x4020FFF0	Data abort	0x1408C
0x4020FFF4	Unused	0x14090
0x4020FFF8	IRQ	0x14094
0x4020FFFC	FIQ	0x14098

24.4.3 Overall Booting Sequence

Figure 24-8 shows the ROM code flowchart.

The main loop of the booting module goes through the booting device list and tries to get an image from the currently selected booting device. The ROM code follows these steps:

Step 1. The ROM code performs basic configurations and initializations.

Step 2. The path named fast external boot is a special low-latency boot mode. It consists of a blind jump to the external addressable memory as soon as possible.

Step 3. A booting device list is created (see Section 24.4.4.3, *Booting Device List Setup*). The list consists of all devices to be searched for a booting image. The list is created based on the sys_boot pins.

Step 4. Once the booting device list is set, the booting procedure examines the devices on the list serially and either executes memory booting or peripheral booting, depending on current booting device type:

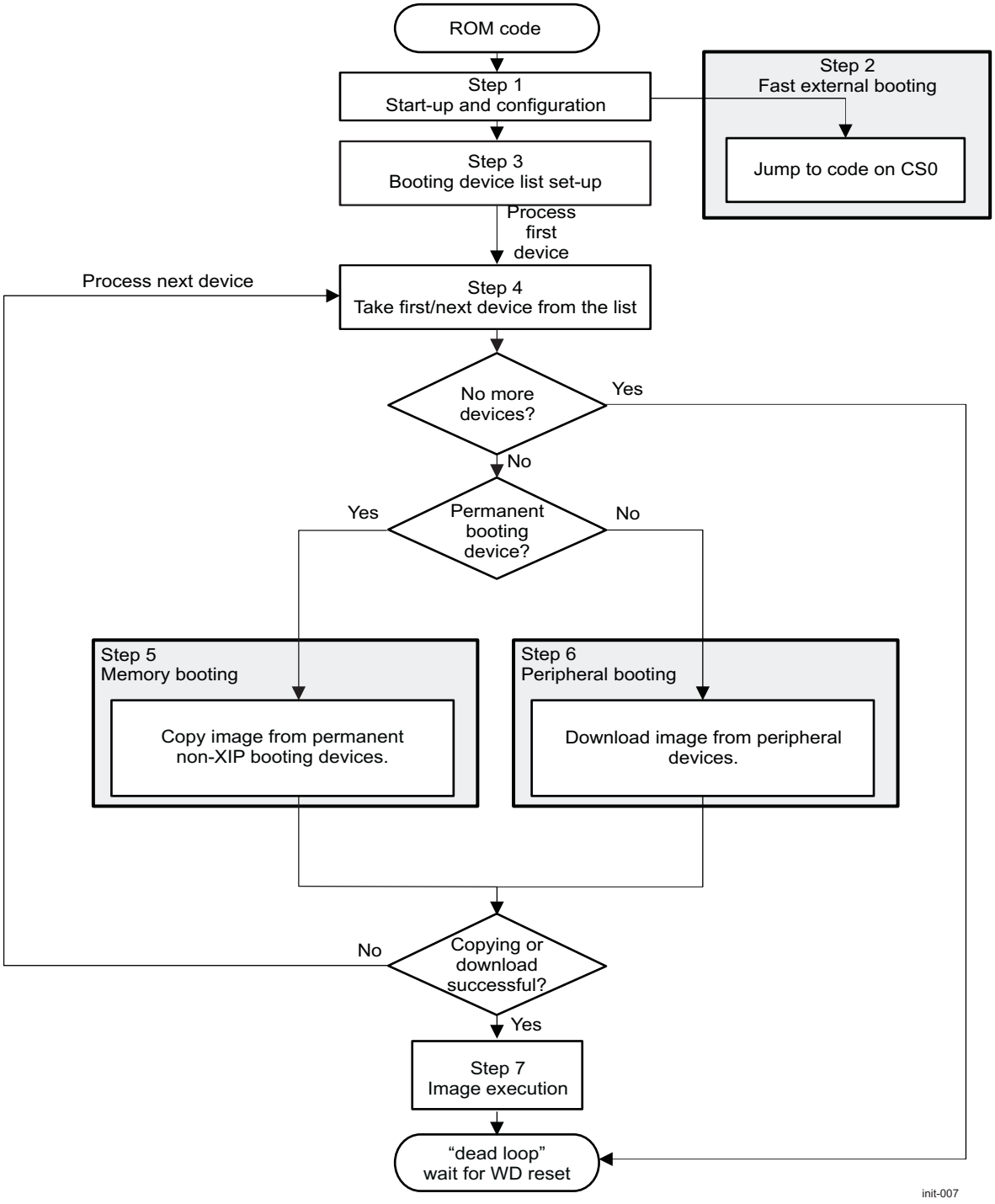
- Memory booting is executed when the booting device is permanent: XIP memory, NAND, DiskOnChip™, OneNAND, or MMC/SD cards.
- The peripheral booting is executed when device is temporary: USB, UART or EMAC.

Step 5. The memory booting procedure reads data from memory type devices. The memory booting is described in detail in Section 24.4.7, *Memory Booting*.

Step 6. The peripheral booting procedure downloads data from communication interfaces. The ROM code uses a simple logical protocol with peripheral booting. First, the processor sends an ASIC ID structure to inform the host about peripheral booting start. Next, the host responds by sending a booting message that can have one of the following meanings: skip peripheral booting, continue peripheral booting, or change the booting device. If the message is to continue, the host sends the whole image preceded by its size. The peripheral booting is described in detail in Section 24.4.5, *Peripheral Booting*.

Step 7. The image is simply started.

Figure 24-8. Overall Booting Sequence



24.4.4 Start-Up and Configuration

24.4.4.1 Start-Up

The ROM code starts at address 0x0001 4000.

24.4.4.2 Clocking Configuration

The ROM code configures the clocks and DPLLs required for ROM code execution.

The configured DPLLs are:

- Peripheral DPLL, set to provide 96 MHz and 48 MHz for peripheral blocks.
- MPU DPLL, set to provide 48 MHz for the MPU.
- Core DPLL, set to provide 192, 96, 48, or 24 MHz for various blocks, such as interconnect, clocked by this DPLL output.

The multipliers and dividers of the DPLLs are set to values which depend on the input clock.

Table 24-11 summarizes the ROM code default settings for key clocks.

Table 24-11. ROM Code Default Clock Settings

Clock	Frequency [MHz]	Source
CORE.CLK	192	CORE DPLL output
L3x2.CLK	192	CORE.CLK
L3.ICLK	96	CORE.CLK/2
L4.ICLK	48	L3.ICLK/2
RM clock	24	L4.ICLK/2
MPU	48	CORE.CLK/4 (MPU DPLL in bypass)

The DPLLs and other settings are configured by default after each type of reset to give the ROM code the same working conditions.

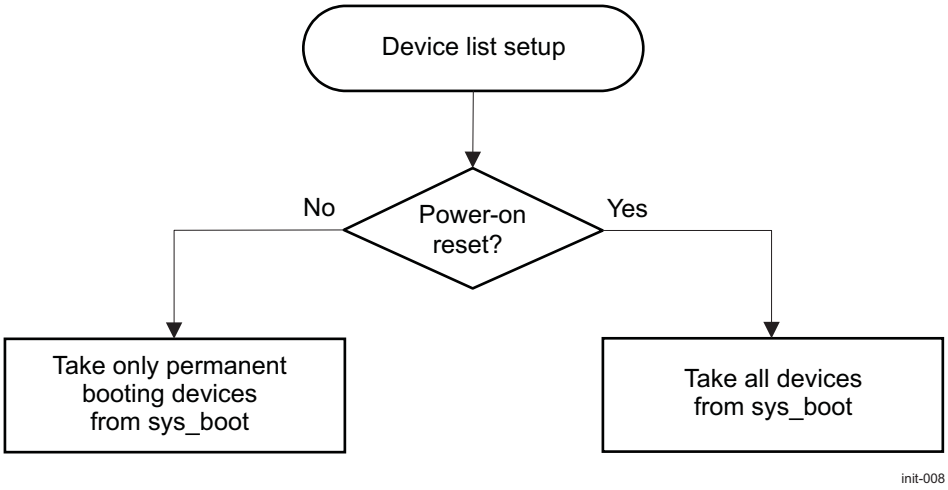
24.4.4.3 Booting Device List Set-Up

The ROM code creates the device list based on the values of sys_boot[5:0]. The sys_boot[5:0] pins are used to index the device table from which the list of devices is extracted.

Figure 24-9 shows how the ROM code sets up the device list depending on the reset source.

NOTE: Only permanent booting devices are put on the list when reset is non power-on and devices are taken from the sys_boot pins.

Figure 24-9. Device List Set-Up



24.4.5 Peripheral Booting

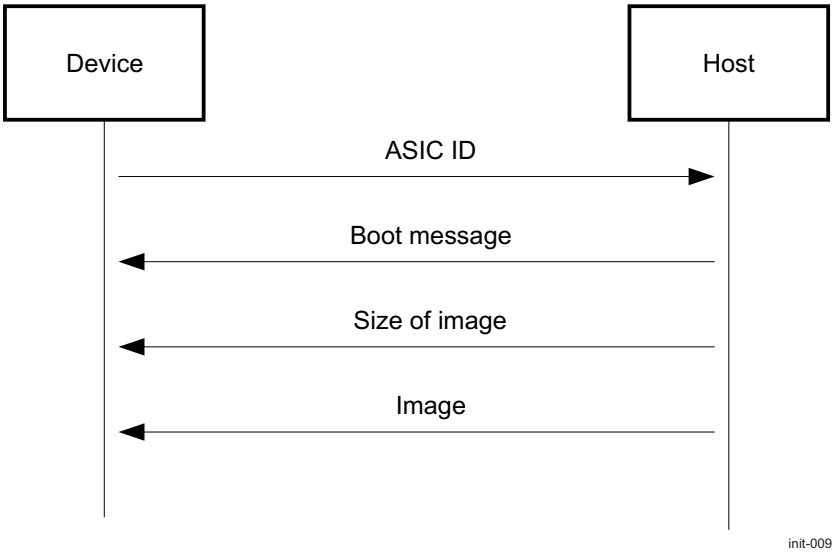
24.4.5.1 Overview

The ROM code can boot from different peripherals:

- HS/FS USB: High/Full-speed USB Interface.
- UART 3: Baud rate 115.2K bps, 8 bits, even parity, 1 stop-bit.
- Ethernet : EMAC RMII Interface.

The purpose of booting from a peripheral is to boot from an external host, such as a PC. This booting method is mostly used for programming flash memories connected to the processor. The protocol used is common to all peripherals. Some minor exceptions are described in the following sections. The common peripheral booting protocol is shown in Figure 24-10.

Figure 24-10. Common Peripheral Booting Protocol



The ROM code first initializes the interface and sends a message called ASIC ID to a host. The content of this message is summarized in Table 24-12. The host uses this message to send only appropriate data to the processor according to the identification codes sent.

The ROM code waits 300 ms for an answer from the host. If a time-out occurs, the peripheral booting returns to the main booting procedure with TIMEOUT status.

Table 24-12. ASIC ID Structure

ASIC ID Item	Size [Bytes]	Description
Items	1	Number of subblocks
ID subblock	7	Device identification information
Reserved for non-GP devices	4	Reserved
ID subblock	23	Identification data
Reserved for non-GP devices	23	Reserved
Checksum subblock	11	CRC (4 bytes)

The host can send different messages as described in Table 24-13. If the second or third message is not received, the ROM code stops the current peripheral booting procedure and returns to the main booting, which determines the next booting device according to the boot message received.

If the first message is received without a time-out, the image size (as a 32-bit word) and the image itself are expected to be received. The image is downloaded directly at address 0x40200000 in the internal RAM.

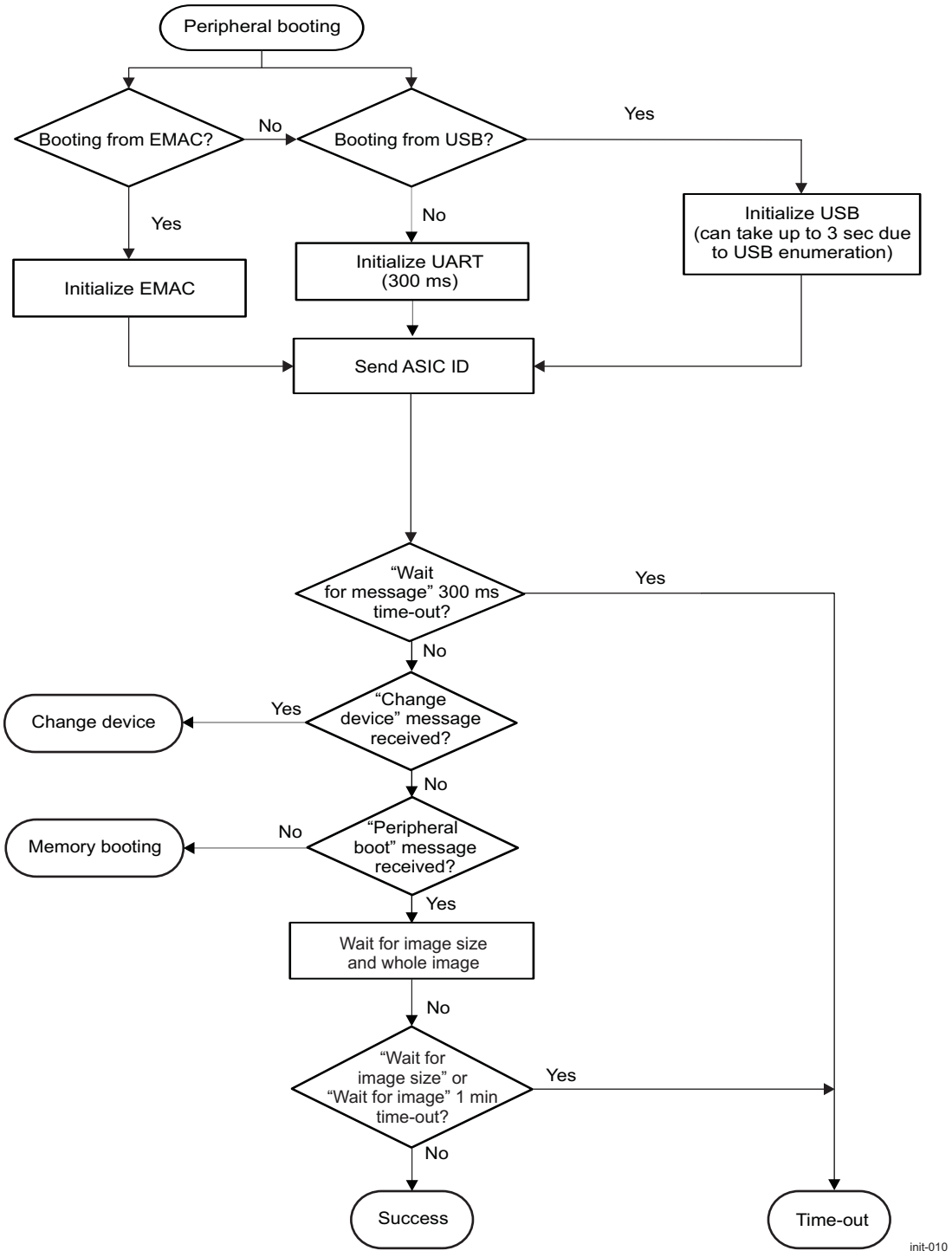
The ROM code waits up to one minute for completion. If the downloading procedure does not complete before this period, the peripheral booting fails. If the download passes, the peripheral booting succeeds and the image can be executed.

Table 24-13. Boot Messages

Message Name	Value	Description
Peripheral boot	0xF003 0002	Continue peripheral booting.
Change device	0xF003XX06	Skip current peripheral booting and continue booting from device type indicated by XX: 0x00 - Void, no device 0x01 - XIP memory 0x02 - NAND 0x03 - OneNAND 0x04 - Reserved 0x05 - MMC/SD2 0x06 - MMC/SD1 0x07 - XIP memory with wait monitoring 0x08..0x0F - Reserved 0x10 - UART 0x11 - HS USB 0x12 - EMAC 0x13 - EMAC Others - Reserved
Next device	0xFFFF FFFF	Skip current device and move to the next device on the device list.
Memory booting	Others	Skip current peripheral booting and move to the first device for memory booting.

The peripheral booting procedure is summarized in Figure 24-11.

Figure 24-11. Peripheral Booting Procedure



24.4.5.2 UART

The ROM code supports booting from a UART interface with the following characteristics:

- UART interface 3.
- Communication parameters set to 115.2k baud, 8 bits, even parity, 1 stop-bit.
- Two-pin interface: RX/TX with SW flow control (XON/XOFF). The other UART pins are left at their default configuration.
- The UART time-out is 300 ms.

24.4.5.3 USB

The ROM code supports booting from a USB interface with the following characteristics:

- HS USB interface.
- The enumeration time-out is 3 seconds.

The ROM code USB driver conforms with the USB 2.0 specification and the USB on-the-go (OTG) supplement. It supports transactions at HS (that is, 480 Mbps) and FS (that is, 12 Mbps). During peripheral booting, only the USB device functionality is used. The driver resides in the on-chip memory (OCM) ROM, which is small. The driver therefore contains the minimum functionality needed as a USB device and is not a full-fledged driver. It does not contain the functionality needed for a USB host. The device functionality of the USB OTG controller is used for the peripheral booting process in the ROM code.

24.4.5.3.1 USB Driver Descriptors

USB devices report their attributes using descriptors. A descriptor is a data structure with a defined format. Each descriptor begins with a byte-wide field that contains the total number of bytes in the descriptor followed by a byte-wide field that identifies the descriptor type. Using descriptors allows concise storage of the attributes of individual configurations so that each configuration can reuse descriptors or portions of descriptors from other configurations that have the same characteristics. Where appropriate, descriptors contain references to string descriptors. String descriptors contain displayable, human-readable information that describes a descriptor. These descriptor details can be used for tool development or debugging:

- Device descriptor.

A device descriptor contains general information about a USB device, including information that applies globally to the device and all device configurations. A USB device has only one device descriptor. Because the ROM code uses the HS feature of the USB core, a device-qualifier descriptor is required. Table 24-14 describes the device descriptors.

Table 24-14. Device Descriptor

Field	Value	Description
bLength	0x12	Size of this descriptor in bytes
bDescriptorType	0x01	Device descriptor type
bcdUSB	0x0210	USB specification release number in binary coded decimal (BCD) format
bDeviceClass	Vendor-specific (0xFF)	Class code
bDeviceSubClass	Vendor-specific (0xFF)	Subclass code
bDeviceProtocol	Vendor-specific (0xFF)	Protocol code
bMaxPacketSize0	0x40	Maximum packet size for endpoint 0
idVendor	0x0451	Vendor ID
idProduct	0xD009	Product ID
bcdDevice	0x0000	Device release number
iManufacturer	See values in Section 24.4.5.3.2.	Index of string descriptor describing manufacturer
iProduct	See values in Section 24.4.5.3.2.	Index of string descriptor describing product

Table 24-14. Device Descriptor (continued)

Field	Value	Description
iSerialNumber	See values in Section 24.4.5.3.2.	Index of string descriptor describing device serial number
bNumConfigurations	0x01	Number of possible configurations

- Device-qualifier descriptor

The device-qualifier descriptor contains information about a HS-capable device that changes if the device operates at its other speed. This descriptor is retrieved by the host using the GetDescriptor() request (standard device request). Table 24-15 describes a device-qualifier descriptor.

Table 24-15. Device-Qualifier Descriptor

Field	Value	Description
bLength	0x0a	Size of this descriptor in bytes
bDescriptorType	0x06	Device-qualifier descriptor type
bcdUSB	0x0210	USB specification release number in BCD
bDeviceClass	0xFF	Class code
bDeviceSubClass	0xFF	Subclass code
bDeviceProtocol	0xFF	Protocol code
bMaxPacketSize0	0x40	Maximum packet size for endpoint 0
bNumConfigurations	0x01	Number of possible configurations
bReserved	0x00	Reserved for future use

- Configuration descriptor

This descriptor gives information about a specific device configuration. The descriptor describes the number of interfaces supported by the configuration. See Table 24-16 for details.

Table 24-16. Configuration Descriptor

Field	Value	Description
bLength	0x09	Size of this descriptor in bytes
bDescriptorType	0x02	Configuration descriptor type
wTotalLength	–	Combined length of all descriptors
bNumInterfaces	0x01	Number of interfaces supported
bConfigurationValue	0x01	Value to use as an argument for the SetConfiguration() request
iConfiguration	Index	Index of string descriptor describing this configuration
bmAttributes	0xc0	Power setting and remote wake-up
bMaxPower	0x32	Maximum power consumption of the USB device

- Other speed configuration descriptor

This descriptor describes the configuration of a HS-capable device if it operates at its other possible speed. See Table 24-17 for details.

Table 24-17. Other Speed Configuration Descriptor

Field	Value	Description
bLength	0x09	Size of this descriptor in bytes
bDescriptorType	0x07	Other speed configuration descriptor type
wTotalLength	–	Combined length of all descriptors
bNumInterfaces	0x01	Number of interfaces supported
bConfigurationValue	0x01	Value to use as an argument for the SetConfiguration() request

Table 24-17. Other Speed Configuration Descriptor (continued)

Field	Value	Description
iConfiguration	Index	Index of string descriptor describing this configuration
bmAttributes	0xc0	Power setting and remote wake-up
bMaxPower	0x32	Maximum power consumption of the USB device

- Interface descriptor

This descriptor describes a specific interface in a configuration. See Table 24-18 for details.

Table 24-18. Interface Descriptor

Field	Value	Description
bLength	0x09	Size of this descriptor in bytes
bDescriptorType	0x04	Interface descriptor type
bInterfaceNumber	0x00	Number of this descriptor
bAlternateSetting	0x00	Value to select the alternate setting
bNumEndpoints	0x02	Number of endpoints used for this interface
bInterfaceClass	0xFF	Class code
bInterfaceSubClass	0xFF	Subclass code
bInterfaceProtocol	0xFF	Protocol code
iInterface	Index	Index of string descriptor describing this interface

- Endpoint descriptor

  Each endpoint used for an interface has its own descriptor. This descriptor contains information required by the host to determine the bandwidth requirements of each endpoint. This descriptor is returned as part of the GetDescriptor(Configuration) request. See Table 24-19 and Table 24-20 for details.

Table 24-19. BULK IN Endpoint Descriptor

Field	Value	Description
bLength	0x07	Size of this descriptor in bytes
bDescriptorType	0x05	Endpoint descriptor type
bEndpointAddress	0x81 (1 IN)	Address of the endpoint on the USB device
bmAttributes	0x02 (Bulk)	Type of transfer
wMaxPacketSize	See (1).	Number of endpoints used for this interface
bInterval	0x00	Maximum NAK rate

(1) The maximum size is 0x0200 (512 bytes) for HS bulk endpoint and 0x0040 (64 bytes) for FS bulk endpoint.

Table 24-20. BULK OUT Endpoint Descriptor

Field	Value	Description
bLength	0x07	Size of this descriptor in bytes
bDescriptorType	0x05	Endpoint descriptor type
bEndpointAddress	0x01 (1 OUT)	Address of the endpoint on the USB device
bmAttributes	0x02 (Bulk)	Type of transfer
wMaxPacketSize	See (1).	Number of endpoints used for this interface
bInterval	0x00	Maximum NAK rate

(1) The maximum size is 0x0200 (512 bytes) for HS bulk endpoint and 0x0040 (64 bytes) for FS bulk endpoint.

- String descriptors

  String descriptors use UNICODE encoding. The strings in a USB device can support multiple languages. When requesting a string descriptor, the requester specifies the desired language using a 16-bit language ID (LANGID) defined by the USB interface. String index 0 for all languages returns a string descriptor that contains an array of 2-byte LANGID codes supported by the device.

  See the tables describing string descriptors:

  – The language ID string descriptor (Table 24-21)
  – The manufacturer ID string descriptor (Table 24-22)
  – The product ID string descriptor (Table 24-23)
  – The configuration string descriptor (Table 24-24)
  – The interface string descriptor (Table 24-25)

Table 24-21. Language ID String Descriptor

Field	Value	Description
bLength	0x04	Size of this descriptor in bytes
bDescriptorType	0x03	String descriptor type
wLangId	0x0409 (US English)	Language ID code

Table 24-22. Manufacturer ID String Descriptor

Field	Value	Description
bLength	0x06	Size of this descriptor in bytes
bDescriptorType	0x03	String descriptor type
bString	TI	Manufacturer string

Table 24-23. Product ID String Descriptor

Field	Value	Description
bLength	0x0c	Size of this descriptor in bytes
bDescriptorType	0x03	String descriptor type
bString	processor or specific vendor string	Product string

Table 24-24. Configuration String Descriptor

Field	Value	Description
bLength	0x07	Size of this descriptor in bytes
bDescriptorType	0x03	String descriptor type
bString	pbc	Configuration string

Table 24-25. Interface String Descriptor

Field	Value	Description
bLength	0x07	Size of this descriptor in bytes
bDescriptorType	0x03	String descriptor type
bString	pbi	Interface string

24.4.5.3.2 USB Customized Descriptors

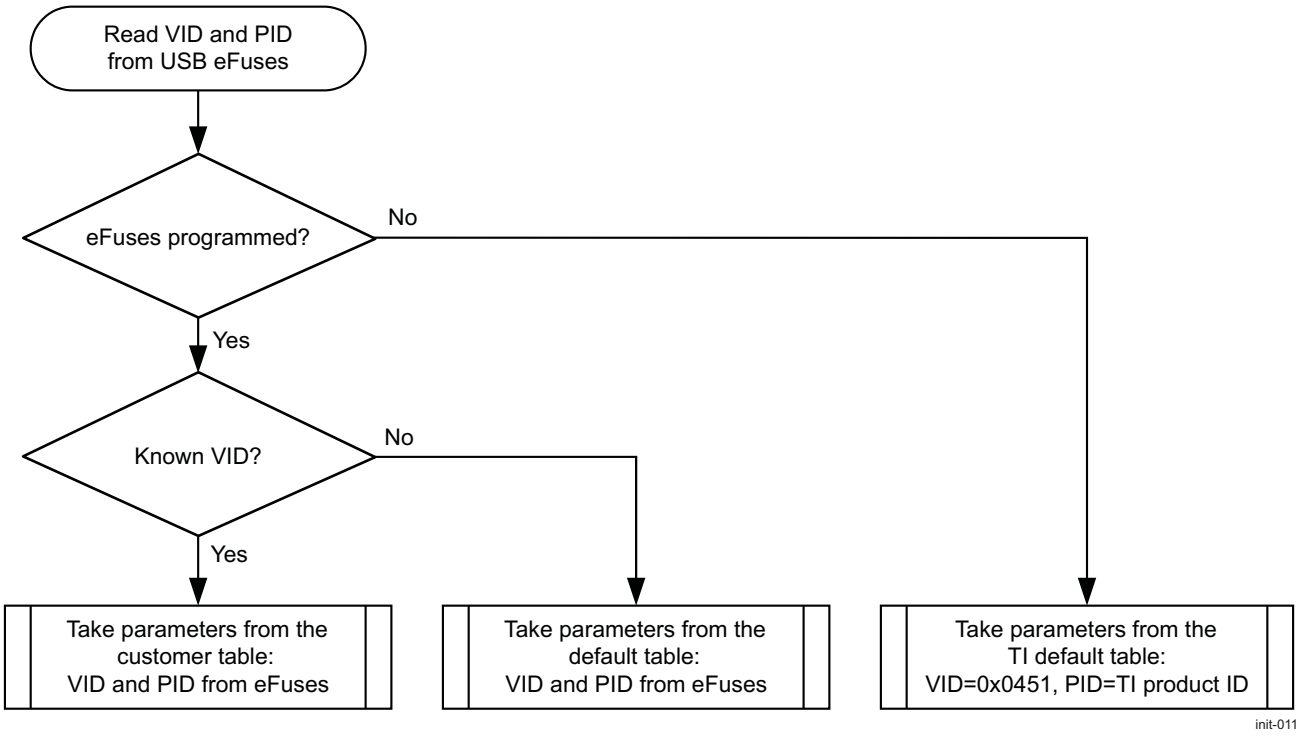
There are two parameters in USB descriptors that customers can define after the chip is created: vendor ID (VID) and product ID (PID). The ROM code uses dedicated eFuses that hold VID and PID values. Other parameters can also be changed based on VID value. The ROM code has an encoded set of parameters for customers who have defined their requirements before the ROM code has been done. Table 24-26 lists the parameters that depend on VID value.

Table 24-26. Customized Descriptor Parameters

Parameter	Size [Bytes]	Default Values	TI Default Values
Device ID code	2	0x0000	0x0000
Device class	1	0xFF	0xFF
Device subclass	1	0xFF	0xFF
Device protocol	1	0xFF	0xFF
Manufacturer	String	N/A	TI
Product	String	processor or specific vendor string	processor or specific vendor string
Serial number	String	N/A	N/A

Figure 24-12 describes an additional customer parameter selection method. It is based on the VID burned in the USB eFuses.

Figure 24-12. Customer USB Descriptor Selection



24.4.5.3.3 USB Driver Functionality

- Transactions supported
  The following transactions are supported:
  - Control transactions: Used for standard device requests
  - Bulk transactions: Used for data transfer in the image downloading stage. The ASIC ID is sent on the Bulk IN endpoint and the image is transferred over the Bulk OUT endpoint from the host.

  The processor's USB device first attaches to the host as an FS device. In the reset mechanism, the

USB core requests HS operation. If the HS negotiation in the reset phase is successful, further transactions are at HS; otherwise, they are at FS. After reset, the USB driver checks for the speed of the device, whether it is FS or HS. Depending on the speed configured by the host, the standard USB device requests are responded to with the corresponding descriptors.

- Standard device request restrictions

Because the USB driver is used only by the ROM code for peripheral booting, some standard device requests are not supported by the driver. Table 24-27 lists the standard device requests supported by the driver.

Table 24-27. Standard Device Requests Supported

Request	Description	Support
CLEAR_FEATURE	Sets/clears a specific feature	Supported only for ENDPOINT_HALT feature
GET_CONFIGURATION	Returns the current device configuration value	Yes
GET_DESCRIPTOR	Returns the specified descriptor	Yes
GET_INTERFACE	Returns the selected alternate setting for the specified interface	Yes
GET_STATUS	Returns the status for the specified recipient	Yes
SET_ADDRESS	Sets the device address	Yes
SET_CONFIGURATION	Sets the device configuration	Yes
SET_DESCRIPTOR	Updates existing descriptors or adds new descriptors	No. Runtime updating of descriptors is not supported.
SET_FEATURE	Sets or enables a specific feature	Supported only for ENDPOINT_HALT feature
SET_INTERFACE	Selects an alternate setting in an interface	No. Runtime setting of alternate features is not supported.
SYNCH_FRAME	Sets and reports an endpoint synchronization frame	No, because isochronous transfers are not used

24.4.5.4 EMAC

The ROM code supports boot over the network. The image obtained over the network is first shadowed into SRAM and then it is executed. The following requirements are to be met for using Ethernet boot.

- The EMAC is configured for full duplex, 100MBPS mode.
- Once the switch has been configured, the ROM code broadcasts the ASIC ID on the LAN using a UDP datagram.
- The ASIC ID packet is sent only once. The server then responds with hex value 0xF003 0002 (Boot Message). The server then sends the size of the boot image.
- The server then sends the boot image binary.
- The MAC address is obtained from below control module registers:
  - Ethernet MAC Address (LSB-24bits) --> 0x4800 2380 (Refer to system control module for more details.)
  - Ethernet MAC Address (MSB-24bits) --> 0x4800 2384 (Refer to system control module for more details.)
- The download address & max size of images used for EMAC boot are exactly the same as USB/UART boot.

24.4.5.4.1 Boot Host Servers

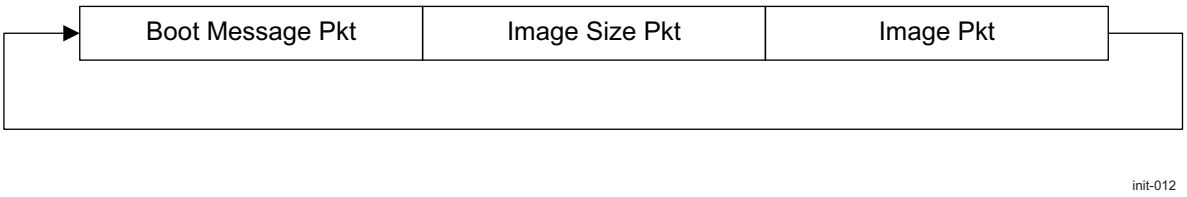
Host servers have two schemes that send the boot image over the LAN in response to broadcasted boot announce frame (which has ASIC ID).

- Dumb Server Response

- Smart Server Response

Dumb servers do not wait for ASIC ID to be received. Instead, dumb servers have a list of EMAC addresses that they use to transmit the boot message, image size, and the image packets. The dumb server sends the image binary packets continuously to every EMAC address.

Figure 24-13. Dumb Servers Boot Response



Smart servers can be present in each subnet or LAN, to respond to a boot announce frame which consists ASIC ID. As a back up, they can also use a statically configured list of EMAC addresses.

Once the smart boot server receives a boot announce frame, it starts transmitting the boot message, image size, and the image binary in subsequent packets to the MAC address that was sent in the boot announce frame. Each time the device receives a packet, it will send the sequence number of the successfully received packet to the smart boot server as acknowledgement.

When the boot server receives the ACK packet from the device, it will send the next packet. This sequence continues untill the entire image is transmitted.

Table 24-28. Boot Announce Frame

Field	Value
Destination MAC address	FF:FF:FF:FF:FF:FF
Source MAC address	From e-fuse
Protocol	IP (0x0800)
Header length and version	IPV4
Type of service	0
Total length of data and header	Calculated and stored
Identification	0x1 (unused)
Fragment offset field and flag	0
Time to live	0xFF (max)
Protocol	UDP (17)
Checksum	Calculated and stored
Source IP address	0:0:0:0
Destination IP address	255.255.255.255
UDP source port	55556
UDP destination port	55555
Length of UDP header and data	Calculated and stored
Checksum	0 (Not used)
Start of Data marker	Always 0xFF
Sequence number (8bit)	Incrementing, starting from 0
Data – ASIC ID	

24.4.5.4.2 EMAC Boot UseCase

If the device needs to boot a 768 byte image, the boot server will split the image into three 256 byte packages and continue looping while the packages are being transmitted. The boot server will keep increasing the sequence number in each package. In addition to the three 256 byte packets, the boot server will also send one packet with the boot message and one packet with the image size.

The messages will be sent in the following sequence:

1. On Power ON Reset the device transmits the Boot Announce frame with ASIC ID.
2. The boot server receives the boot announce frame, gets the MAC address of the device, and transmits the boot message to the device with the sequence number in the frame set to 0.
3. The boot server then transmits the size of the image binary to the device with the sequence number in the frame set to 1.
4. The boot server then splits the boot image binary into 256 byte chunks and transmits the chuncks to the device one by one, incrementing the sequence number for each frame.
5. The boot server repeats steps 2, 3, and 4 in a loop untill it gets an ACK packet from the device.
6. The device receives the frames transmitted by the boot server sequentially. If it receives a frame out of sequence, the device will discard it and wait untill it receives a frame with the correct sequence number. Once a frame is received, the device sends an ACK packet back to the boot server containing the sequence number of the frame it just received.
7. When the boot server receives the ACK sequence number, it stops sending that particular packet in the next cycle of packet transmission but continues to send the packets with next sequence number.
8. If the boot server is transmitting packets in a loop with the following sequence numbers 0-1-2-3-4-0-1-2-3-4-0-1-2-3-4…
9. The device receives packet sequence 0 and sends an ACK to the boot server.
10. The boot server then changes its transmit sequence to 1-2-3-4-1-2-3-4-1-2-3-4…
11. If the next packet the device receives has sequence number 2 (and the packet with sequence number 1 got lost in the network), the device will ignore packets 2, 3, and 4 and wait untill it sees packet 1 on the network again.
12. When the device finally receives packet 1, it will send the boot server an ACK for packet 1.
13. The boot server will change its transmit sequence to 2-3-4-2-3-4-2-3-4…

This process will continue untill the device ACKs packet 4 and then the boot server can stop transmitting.

Figure 24-14. EMAC Boot Packet Sequences

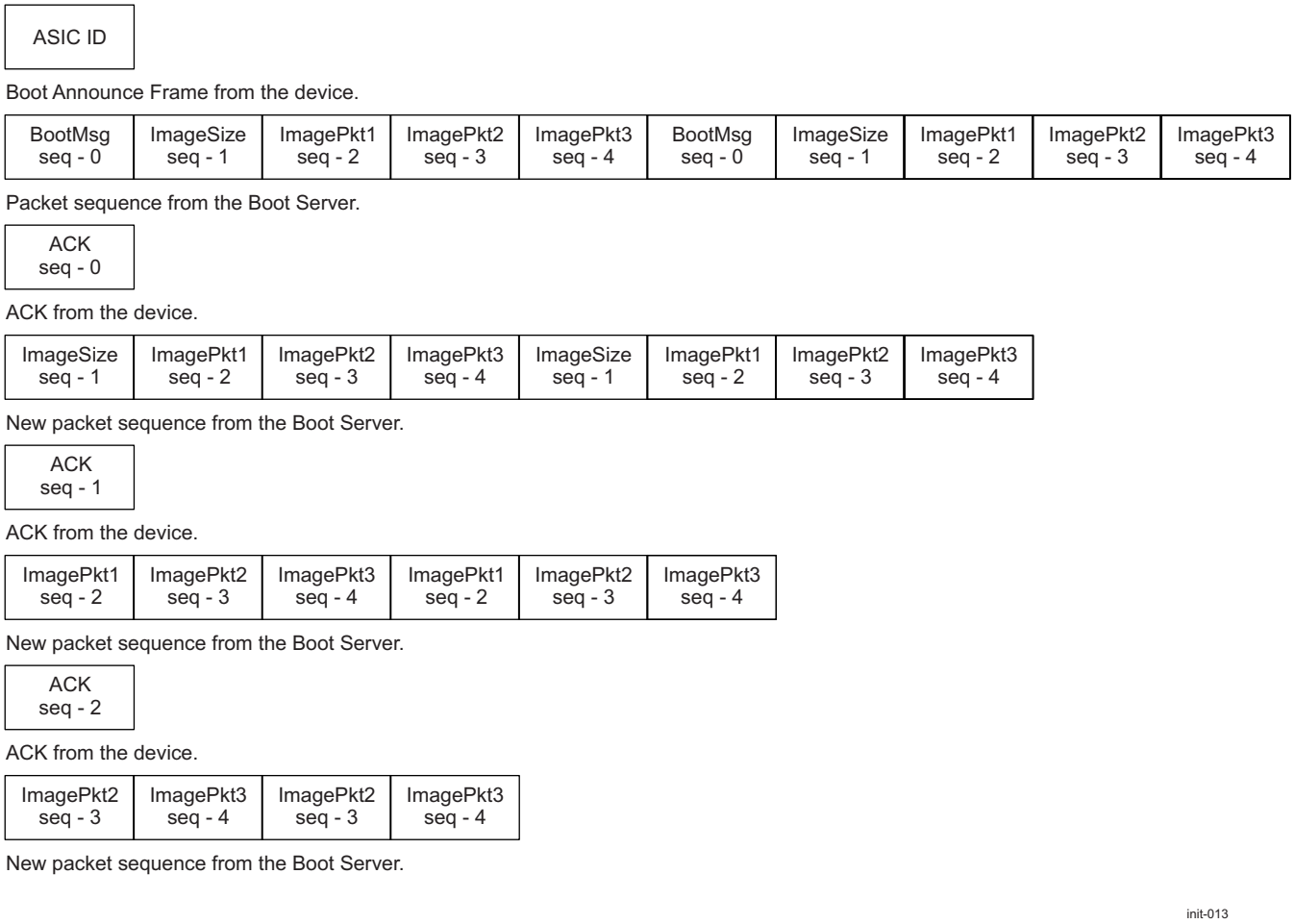


Table 24-29. Frame transmitted by the Boot Server

Field	Value
Destination MAC address	From the boot announce frame source MAC address
Source MAC address	Boot server's MAC address
Protocol	IP (0x0800)
Header length and version	IPV4
Type of service	0
Total length of data and header	Calculated and stored
Identification	0x1 (unused)
Fragment offset field and flag	0
Time to live	0xFF (max)
Protocol	UDP (17)
Checksum	Calculated and stored
Source IP address	Boot server's IP address
Destination IP address	255.255.255.255
UDP source port	55556
UDP destination port	55555
Length of UDP header and data	Calculated and stored
Checksum	0 (Not used)
Start of Data marker	Always 0xFF

Table 24-29. Frame transmitted by the Boot Server (continued)

Field	Value
Sequence number (8bit)	Incrementing, starting from 0
Data[1]	

(1)

(a) When the sequence number is 0, Data is the BOOT MESSAGE that ROM code frame work expects.

(b) When the sequence number is 1, Data is the size of the image binary.

(c) When the sequence number is greater than or equal to 2, Data is the image binary, split into 256byte chunks.

Table 24-30. ACK frame sent by the Device to the Boot Server

Field	Value
Destination MAC address	From the boot announce frame source MAC address
Source MAC address	From E-Fuse
Protocol	IP (0x0800)
Header length and version	IPV4
Type of service	0
Total length of data and header	Calculated and stored
Identification	0x1 (unused)
Fragment offset field and flag	0
Time to live	0xFF (max)
Protocol	UDP (17)
Checksum	Calculated and stored
Source IP address	0.0.0.0
Destination IP address	From the boot server frame source ip address
UDP source port	55556
UDP destination port	55555
Length of UDP header and data	Calculated and stored
Checksum	0 (Not used)
Start of Data marker	Always 0xFF
Sequence number (8bit)	From the last boot server frame
Pad – 24 bytes	To make the Ethernet packet greater than 64 bytes

24.4.6 Fast External Booting

24.4.6.1 Overview

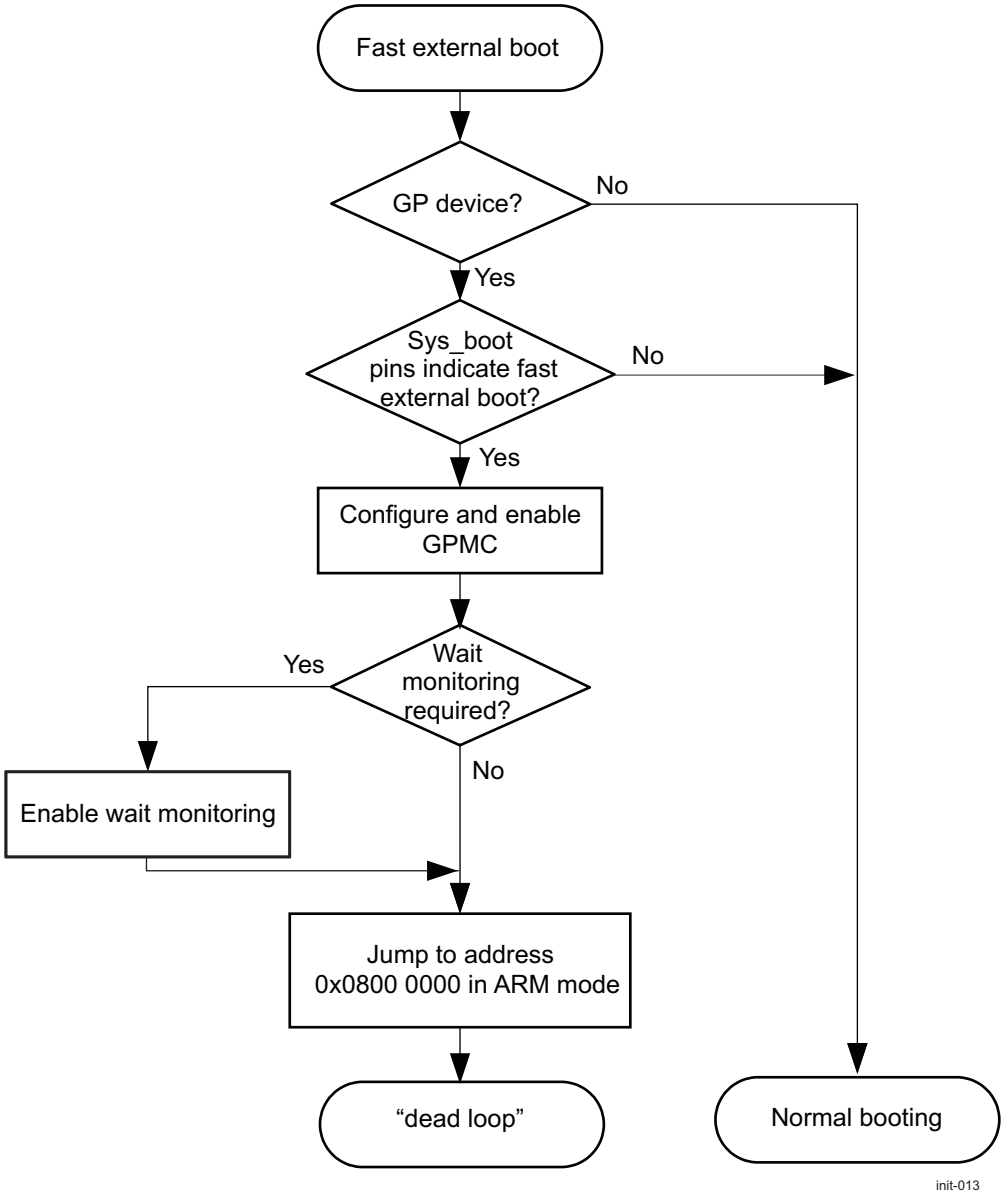
The fast external boot is a special memory booting mode. It is a blind jump to a code in an external XIP device connected to CS0. Fast external booting lets customers create their own booting code.

The jump is performed with minimum on-chip ROM code execution.

24.4.6.2 External Booting

Figure 24-15 shows the fast external boot procedure. The code is at the beginning of the public part and is written in assembly. The code does not use any RAM.

Figure 24-15. Fast External Boot



24.4.7 Memory Booting

24.4.7.1 Overview

The memory booting process takes care of starting an external code in memory type devices. These devices are called permanent booting devices because the processor always uses them for booting. The supported permanent booting devices are:

- NOR all devices up to 1G bit (128M bytes)
- NAND devices from 64M bits
- OneNAND devices from 512M bits
- SD/MMC flash cards with active primary partition of type FAT12/16/32
- DiskOnChip™ H3 devices
- SPI EEPROM

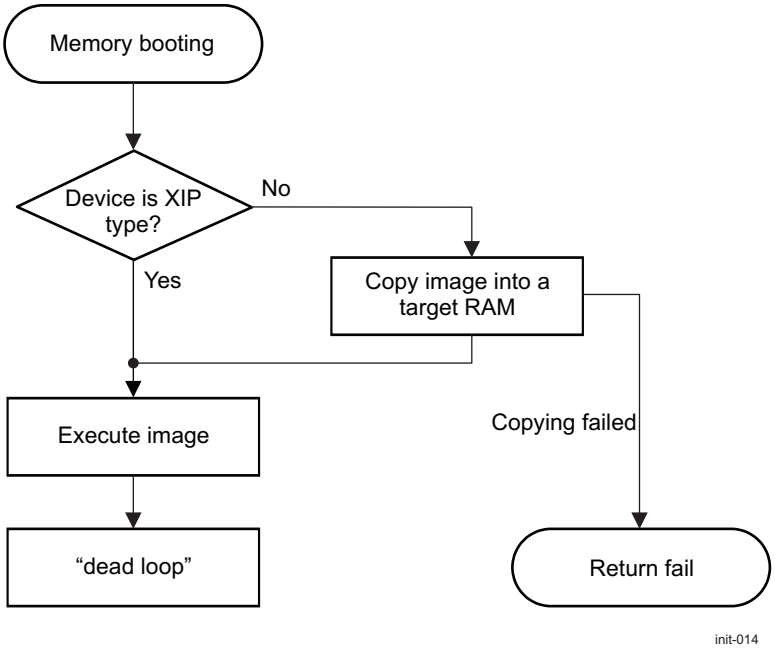
There are two main groups of permanent booting devices distinguished by code shadowing. Code shadowing means copying a code from a nondirectly addressable device (non-XIP) into RAM, where the code can be executed. Directly addressable devices are called eXecute-in-place (XIP) devices.

The memory booting flowchart shown in Figure 24-16 is an overview of common procedure for all types of devices. The first step is to shadow the image, if the device is not XIP. The last step is image authentication and execution.

Unsuccessful authentication or return from image results in dead loop.

If shadowing fails, memory booting returns to the main booting procedure, which selects the next device for booting.

Figure 24-16. Memory Booting



24.4.7.2 Non-XIP Memory

Figure 24-17 details the procedure used when memory booting runs with non-XIP devices. The grayed procedures are specific to each device. NAND and OneNAND devices use up to four copies of the image in the first four physical blocks. Therefore, the ROM code searches for the image in the first four physical blocks of these devices. Other devices use only one copy of the image and the block loop runs only once.

Figure 24-17. Detailed Memory Booting for Non-XIP Devices

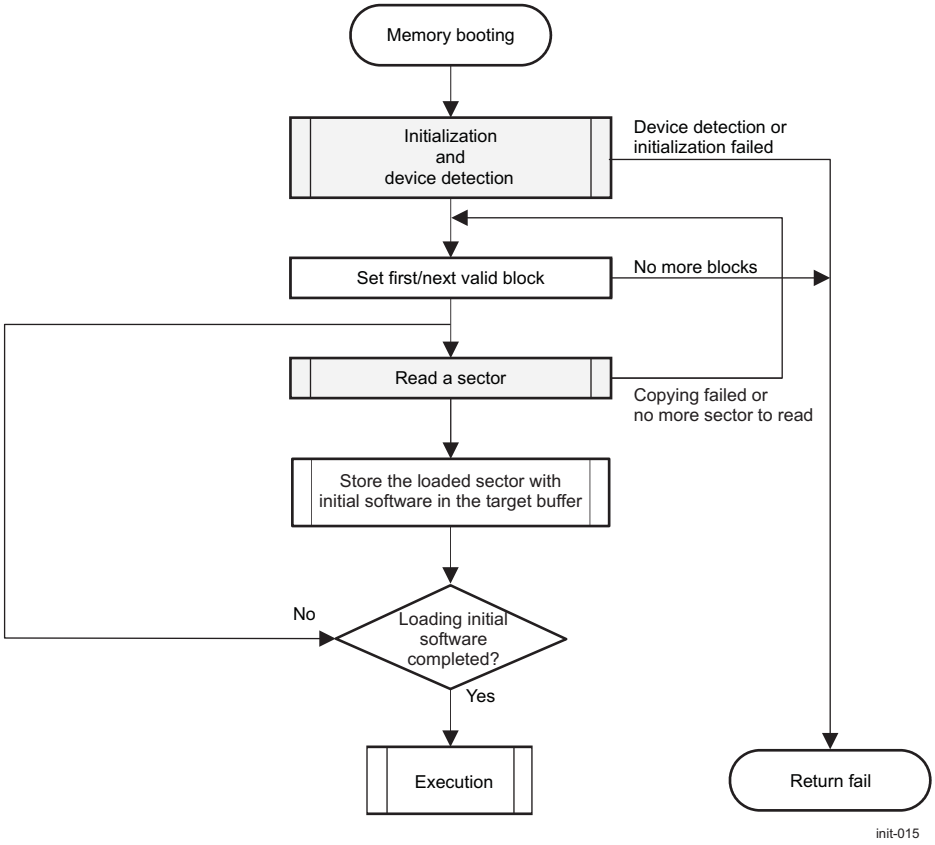


Table 24-31 summarizes numbers of blocks and sectors which are searched during the memory booting from devices requiring image shadowing. NANDs and OneNAND are organized with blocks, which are erasable units. DOC memory has only one block reserved for booting purposes, which overlaps the XIP part. MMC/SD card booting consists of reading a file. Since there is only one file read, it can be considered as one block trial.

Table 24-31. Blocks and Sectors Searched on Non-XIP Memories

Memory	Maximum Number of Checked Blocks	Number of Sectors Searched
NAND	First 4	Number of sectors in a block[1]
OneNAND	First 4	8
DOC[2]	1	4
MMC/SD	One file	1

(1) Depends on NAND type.

(2) DOC memory image must contain first sector filled with 0xFF or 0x00 because XIP booting always precedes DOC booting and the same data is used for XIP area. Therefore, a void sector at the beginning makes XIP booting failed and moves to DOC booting.

The following sections detail each supported device type.

For more details about the GPMC module, see the *Memory Subsystem* chapter.

24.4.7.3 XIP Memory

The ROM code can boot directly from XIP devices. Typical XIP devices are NOR flash memories. Supported XIP devices have the following characteristics:

- The GPMC is used as the communication interface.
- Memories up to 1G bit (128M bytes) can be connected.

- x16 data bus width only
- Asynchronous protocol and address/data multiplexed mode
- The GPMC clock is 48 MHz.
- The device is connected to cs0 mapped to address 0x0800 0000.
- The wait pin signal gpmc_wait0 is monitored according to the sys_boot configuration pins.

Depending on the sys_boot option, the GPMC can be configured to use the wait signal connected or not to the gpmc_wait0 pin. Wait pin polarity is set to stall accessing memory when gpmc_wait0 is low. Wait monitoring is to be used with memories that require a long time for initialization after reset, or that must pause while reading data. An example of such memory is DiskOnChip™.

For an XIP memory booting, no user intervention is needed; the following steps are described for debugging.

The booting from an XIP device can be described in the following points:

1. Configure the GPMC for XIP device access.
2. Set the image location to 0x0800 0000.
3. Verify that a bootable image is present at the image location.
4. Execute the image if it is found.
5. If the image is not found, return from XIP booting to the main booting loop.

24.4.7.3.1 GPMC Initialization

Table 24-32 describes the timing settings of GPMC set for XIP and other address-data accessible devices, like DiskOnChip or OneNAND. Table 24-32 is included for debug information.

Table 24-32. XIP Timing Parameters

Parameter	Value [Clock Cycles]	Register Initialization (i = 0–7)	Reset Value
Write cycle time	17	The GPMC_CONFIG5_i[12:8] WRCYCLETIME bit field is set to 0x11.	0x11
Read cycle time	17	The GPMC_CONFIG5_i[4:0] RDCYCLETIME bit field is set to 0x11.	0x11
CS low time	1	The GPMC_CONFIG2_i[3:0] CSONTIME bit field is set to 0x1.	0x1
CS high time	16	The GPMC_CONFIG2_i[12:8] CSRDOFFTIME bit field is set to 0x10.	0x10
ADV low time	1	The GPMC_CONFIG3_i[3:0] ADVONTIME bit field is set to 0x1.	0x1
ADV high time	2	The GPMC_CONFIG3_i[12:8] ADVRDOFFTIME bit field is set to 0x2.	0x2
OE low time	3	The GPMC_CONFIG4_i[3:0] OEONTIME bit field is set to 0x3.	0x3
OE high time	16	The GPMC_CONFIG4_i[12:8] OEOFFTIME bit field is set to 0x10.	0x10
WE low time	3	The GPMC_CONFIG4_i[19:16] WEONTIME bit field is set to 0x3.	0x03
WE high time	15	The GPMC_CONFIG4_i[28:24] WEOFFTIME bit field is set to 0xF.	0x10
Data latch time	15	The GPMC_CONFIG5_i[20:16] RDACCESSTIME bit field is set to 0xF.	0x0F

24.4.7.4 NAND

NAND flash memory is not an XIP device; it requires shadowing before the code can be executed. ROM code support for the NAND flash devices has the following characteristics:

- The GPMC is the communication interface.

- Device from 64 Mb (8MB)
- x8 and x16 bus width
- Small page size (512 bytes + 16 bytes) and large page size (2048 bytes + 64 bytes)
- Chip enable (CE) don't care devices only
- Single level cell (SLC)
- Device identification is based on standard identification data or ID2 protocol.
- One-bit error checking and correction (ECC) is used to protect a 512-byte sector.
- GPMC timings are adjusted for NAND access.
- The GPMC clock is 48 MHz.
- The device is connected to CS0.
- The wait pin signal gpmc_wait0 is connected to the NAND BUSY output.
- Four physical blocks are searched for image. Block size depends on the device.

For NAND memory booting, no user intervention is needed; the information in the following subsections is included for debugging.

24.4.7.4.1 Initialization and NAND Detection

The initialization routine for NAND consists of three parts: GPMC initialization, device detection with parameter determination, and bad block detection/verification.

- GPMC initialization

The GPMC interface is configured so that it can access NANDs. Because NANDs do not need the address bus, it is released. The data bus width is initially set to 8 bits. If necessary, it is changed to 16 bits after the device parameters are determined. Table 24-33 shows the GPMC configuration used during NAND boot. Table 24-33 is included for debug information.

Table 24-33. NAND Timing Parameters

Parameter	Value [Clock Cycles]	Register Initialization (i = 0–7)	Reset Value
Write cycle time	20	The GPMC_CONFIG5_i[12:8] WRCYCLETIME bit field is set to 0x14.	0x11
Read cycle time	20	The GPMC_CONFIG5_i[4:0] RDCYCLETIME bit field is set to 0x14.	0x11
CS low time	0	The GPMC_CONFIG2_i[3:0] CSONTIME bit field is set to 0x0.	0x1
OE low time	5	The GPMC_CONFIG4_i[3:0] OEONTIME bit field is set to 0x5.	0x3
OE high time	16	The GPMC_CONFIG4_i[12:8] OEOFFTIME bit field is set to 0x10.	0x10
WE low time	3	The GPMC_CONFIG4_i[19:16] WEONTIME bit field is set to 0x3.	0x3
WE high time	15	The GPMC_CONFIG4_i[28:24] WEOFFTIME bit field is set to 0xF.	0x10
Data latch time	14	The GPMC_CONFIG5_i[20:16] RDACCESSTIME bit field is set to 0xE.	0xF

- Device detection and parameters

The ROM code must first identify the NAND type connected on the GPMC interface. The GPMC is initialized using 8 bits, asynchronous mode. The NAND device is reset and its status is polled until it is ready for operation, then the Read ID command is issued. If the Read Device ID is recognized as a supported device, the device parameters are extracted from an internal ROM code table. Table 24-34 lists the supported devices.

Table 24-34. Supported NAND Devices

Capacity	Device ID	Bus Width	Page Size in KB
64Mb	E6h	8	512
128Mb	33h	8	512
128Mb	73h	8	512
128Mb	43h	16	512
128Mb	53h	16	512
256Mb	35h	8	512
256Mb	75h	8	512
256Mb	45h	16	512
256Mb	55h	16	512
512Mb	36h	8	512
512Mb	76h	8	512
512Mb	46h	16	512
512Mb	56h	16	512
512Mb	A2h	8	2048
512Mb	F2h	8	2048
512Mb	B2h	16	2048
512Mb	C2h	16	2048
1Gb	39h	8	512
1Gb	79h	8	512
1Gb	49h	16	512
1Gb	59h	16	512
1Gb	78h	8	512
1Gb	72h	16	512
1Gb	74h	16	512
1Gb	A1h	8	2048
1Gb	F1h	8	2048
1Gb	B1h	16	2048
1Gb	C1h	16	2048
2Gb	AAh	8	2048
2Gb	DAh	8	2048
2Gb	BAh	16	2048
2Gb	CAh	16	2048
2Gb	71h	8	512
2Gb	51h	16	512
2Gb	31h	8	512
2Gb	41h	16	512
4Gb	ACh	8	2048
4Gb	DCh	8	2048
4Gb	BCh	16	2048
4Gb	CCh	16	2048
8Gb	A3h	8	2048
8Gb	D3h	8	2048
8Gb	B3h	16	2048
8Gb	C3h	16	2048
16Gb	A5h	8	2048
16Gb	D5h	8	2048
16Gb	B5h	16	2048
16Gb	C5h	16	2048

Table 24-34. Supported NAND Devices (continued)

Capacity	Device ID	Bus Width	Page Size in KB
32Gb	A7h	8	2048
32Gb	B7h	16	2048
64Gb	AEh	8	2048
64Gb	BEh	16	2048

After retrieving parameters from the table, page size and block size are updated based on the fourth byte of the NAND ID data. Because of inconsistency among manufacturers, only devices recognized to be at least 2Gb have these parameters updated. Therefore, the ROM code supports 4-KB page devices, but only if their size, according to the table, is at least 2Gb. Devices smaller than 2Gb have the block size parameter set to 32KB when the page size is 512KB and to 128KB when the page size is 2048KB. Table 24-35 shows the fourth ID data byte encoding used in the ROM code.

Table 24-35. Fourth NAND ID Data Byte

Item	Description	I/O Number							
		7	6	5	4	3	2	1	0
Page Size	1KB							0	0
	2KB							0	1
	4KB							1	0
	8KB							1	1
Block Size	64KB			0	0				
	128KB			0	1				
	256KB			1	0				
	512KB			1	1				

The detection procedure is described in Figure 24-18. If the device ID is not recognized, the ROM code tries to read ID2 from the device; the sequence is described in Figure 24-19. The description of the ID2 data content is summarized in Table 24-36. If the ROM code fails to identify device ID or ID2, it returns with FAIL. When the device is successfully detected, the ROM code changes the GPMC to 16-bit bus width if necessary.

Figure 24-18. NAND Device Detection

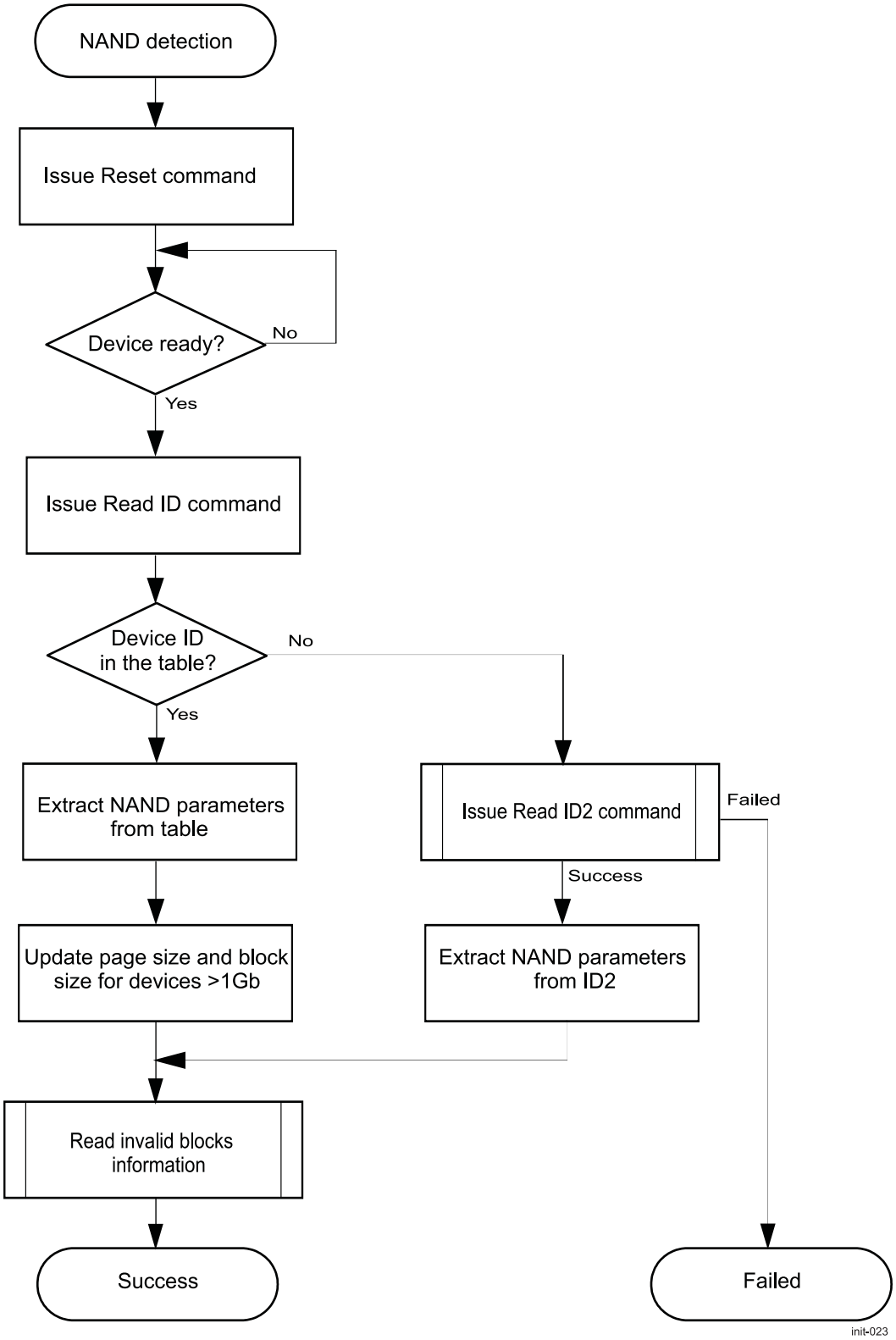


Figure 24-19. NAND ID2 Detection

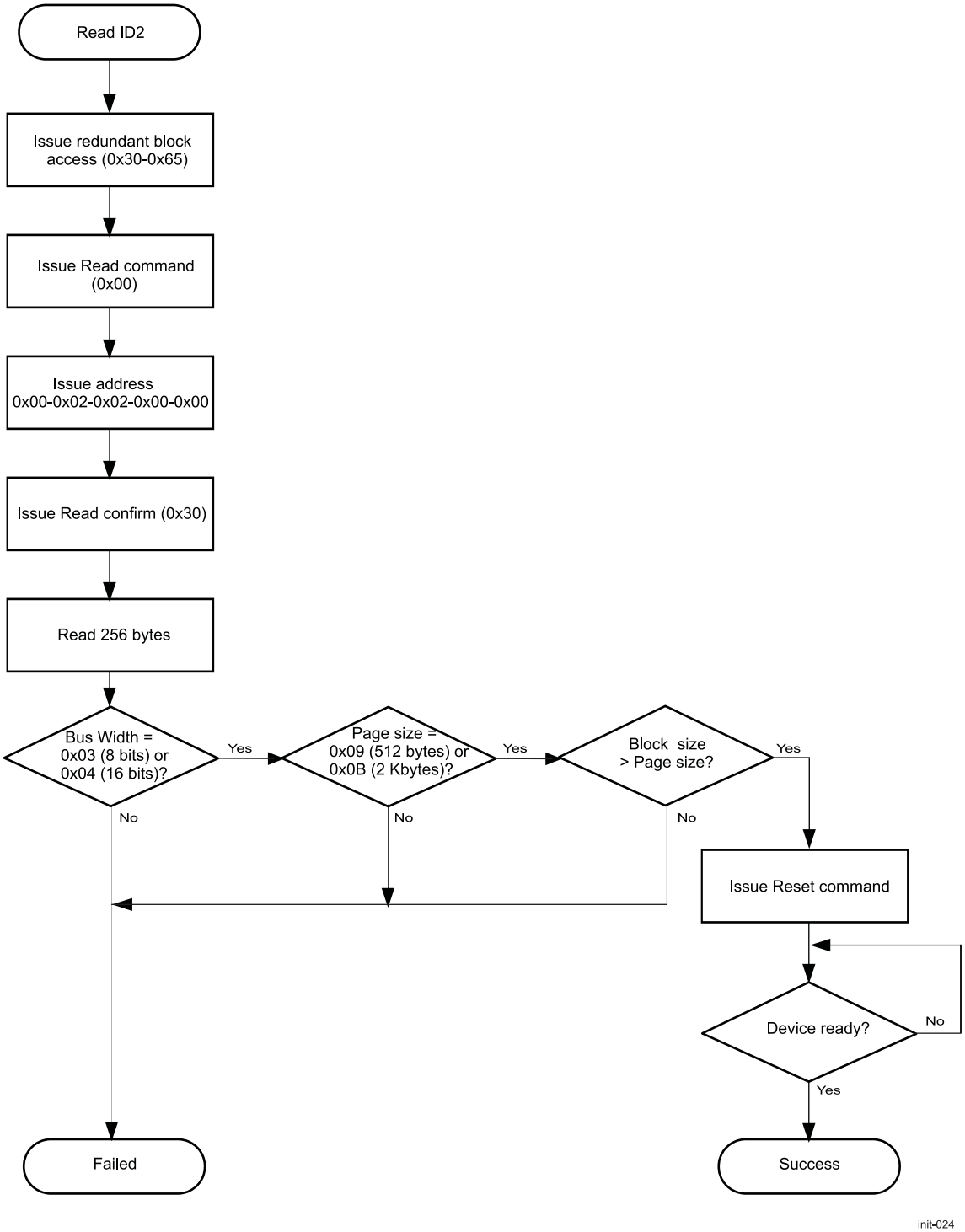


Table 24-36. ID2 Byte Description

Byte Number	Name	Value	Unit	Notes
1	Page size	2X	Bytes	00H = 1B 09H = 512B 0BH = 2KB
2	Block size	2X	Bytes	00H = 1B 0EH = 16KB 11H = 128KB
3	Block count	2X	Pcs	00H = 1 pc 0BH = 2048 pcs 0CH = 4096 pcs
4	Spare size	2X	Bytes	00H = 1B 04H = 16B 06H = 64B
5	Column address	X	Pcs	Higher nibble 1H = 1 column address sequence 2H = 2 column address sequence
	Row address	X	Pcs	Lower nibble 1H = 1 row address sequence 2H = 2 row address sequence
6	ECC type	X	Bit ECC	Higher nibble 0H = No ECC needed 1H = 1-bit ECC 4H = 4-bit ECC
	Bus width	2x	Width	Lower nibble 3H = 8-bit NAND interface 4H = 16-bit NAND interface
7	Number of CEs	X	Pcs	Higher nibble 1H = 1x CE# 2H = 2x CE#
	Cell type	X	Bit/cell	Lower nibble 1H = 1 bit per cell 2H = 2 bits per cell
8	Boot block	X	kB	0H = No boot block 1H = 1KB boot block 2H = 2KB boot block
9	Multiple page prg	X	Pcs	Higher nibble 1H = 1 plane 4H = 4 planes For future use
10	Partial prg count	X	Per page	1H = No partial prg allowed 2H = 2 per page
11	Read time maximum			
12	Prg time maximum			
13	Erase time maximum			
252nd 255th	Identification number	X		B2184D7Bh
256th	Register/spec version	XvX		Higher nibble: Major digit Lower nibble: Decimal digit Registers according to spec: 2v0: 20h

- Bad block detection/verification

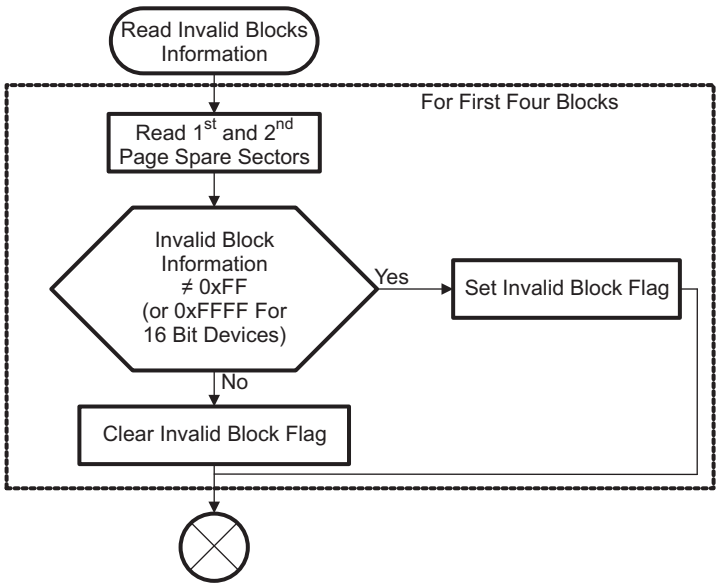
Invalid blocks contain invalid bits whose reliability cannot be ensured by the manufacturer. These bits are identified in the factory or during the programming and reported in the initial invalid block information in the spare area on the first and second page of each block. Because the ROM code looks for an image in the first four blocks, it detects the validity status of these blocks. Blocks detected as invalid are not accessed later. Blocks validity status is coded in the spare areas of the first two pages of a block. Table 24-37 describes validity status coding for all four NAND families.

Table 24-37. Bad Block Marks Locations in NAND Spare Areas

	Small Page NAND	Large Page NAND
8 bit Device		
Block invalid when any byte not equal FFh	6th byte in 1st page	1st byte in 1st page
	6th byte in 2nd page	1st byte in 2nd page
16 bit Device		
Block invalid when any work not equal FFFFh	1st word in 1st page	
	6th work in 1st page	1st word in 1st page
	1st word in 2nd page	1st work in 2nd page
	6th word in 2nd page	

Figure 24-20 depicts the invalid block detection routine. The routine consist in reading spare areas and checking data according to the conditions. The flags are used internally to convey information about each block validity.

Figure 24-20. NAND Invalid Block Detection



24.4.7.4.2 Read Sector Procedure

During the booting procedure, the ROM code reads 512-byte sectors from the NAND device. The reading fails in two cases:

- The accessed sector is in a block marked as invalid.
- The accessed sector contains an error that cannot be corrected with ECC.

Pages can contain errors caused by memory alteration. To correct these errors, the ROM code uses ECC, based on Hamming codes for SLC NAND. The computed ECC is compared to ECC stored in the spare area of the corresponding page. If there are uncorrectable errors, the ROM code returns with FAIL.

24.4.7.5 OneNAND

ROM code support for OneNAND devices has the following characteristics:

- Devices from 512Mb
- The GPMC is the communication interface.
- x16 data bus width only
- Asynchronous protocol and address/data multiplexed mode

- GPMC reset default timings are used.
- The GPMC clock is 48 MHz.
- The device is connected to CS0 mapped to address 0x0800 0000.
- The wait pin signal gpmc_wait0 is not monitored.
- Four physical blocks are searched for image. The block size is 128KB.

The OneNAND device is a NAND matrix coupled with RAM buffers and a NOR-type interface. ECC correction handling is done automatically by the internal state-machine. The page to be accessed is first loaded in the RAM buffer using memory-mapped registers. Then, the page is read directly from the buffer using a NOR-type interface.

For OneNAND memory booting, no user intervention is needed. The information in the following subsections is included for debugging.

24.4.7.5.1 Initialization and OneNAND Detection

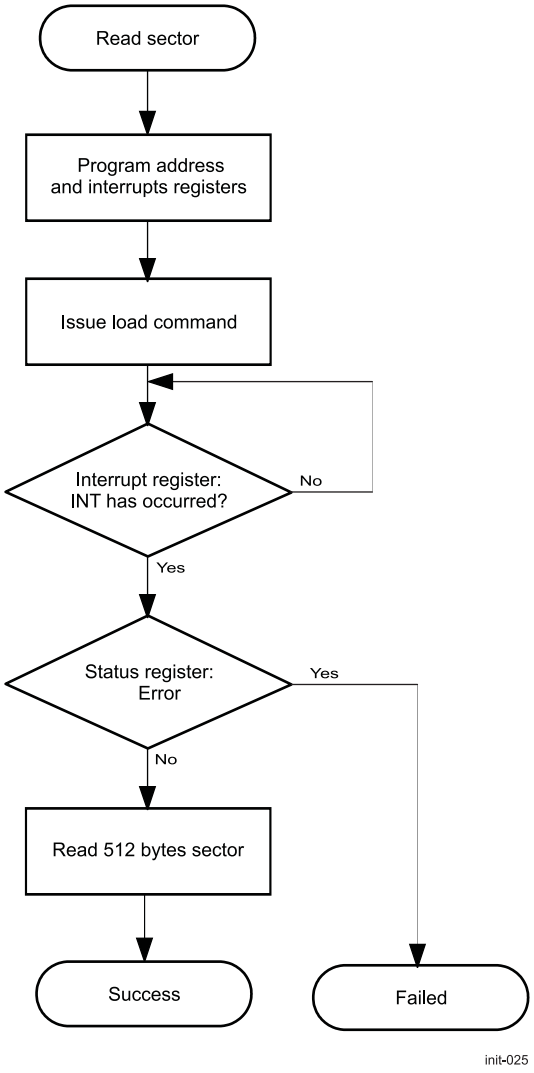
The initialization routine for OneNAND consists of two parts: GPMC initialization and device detection with parameter determination:

- GPMC initialization
  The ROM code first initializes the GPMC interface the same way as for an XIP memory (that is, asynchronous 16-bit multiplexed mode). Wait signal monitoring is disabled. See Table 24-32.
- Device detection and parameters
  The ROM code identifies a OneNAND device by reading the device identification data. There are two ways to read identification data: using serial commands and reading from fixed memory mapped registers. The ROM code reads identification data using both methods and compares the result. When the comparison passes, the ROM code assumes that the OneNAND device is connected. If the device is successfully recognized, the ROM code reads the device configuration (amount and size of data buffers) and configures it for asynchronous mode (default).

24.4.7.5.2 OneNAND Read Sector Procedure

When booting requests a sector from the OneNAND device, the ROM code issues the load operation, which transfers the content of the requested sector to the data buffer RAM. The ROM code waits until the operation completes, polling the OneNAND interrupt register. The status register is then checked and the ROM code returns FAIL if the operation completes with an error. Otherwise, the data buffer RAM is copied to the destination buffer. Figure 24-21 shows this procedure.

Figure 24-21. OneNAND Read Sector



24.4.7.6 MMC/SD Cards

The ROM code supports MMC/SD cards, with some limitations:

- Supports MMC/SD cards compliant with the *Multimedia Card System Specification v4.2* from the MMCA Technical Committee and the *SD I/O Card Specification v 2 .0* from the SD Association. Includes high-capacity (size >2GB) cards: HC-SD and HC MMC
- 1.8V or 3.3V I/O voltage on PORT 1
- 1.8V or 3.3V I/O voltage on PORT 2. External transceiver mode on PORT 2 is not supported.
- Initial 1-bit MMC mode, 4-bit SD mode
- Clock frequency:
  - Initialization sequence mode: 160 kHz
  - Identification mode: 400 kHz
  - Data transfer mode: 1/2 maximum detected rate
- Only one card connected to the bus
- Raw mode, image data read directly from card sectors
- FAT12/16/32 support, with or without a master boot record (MBR).

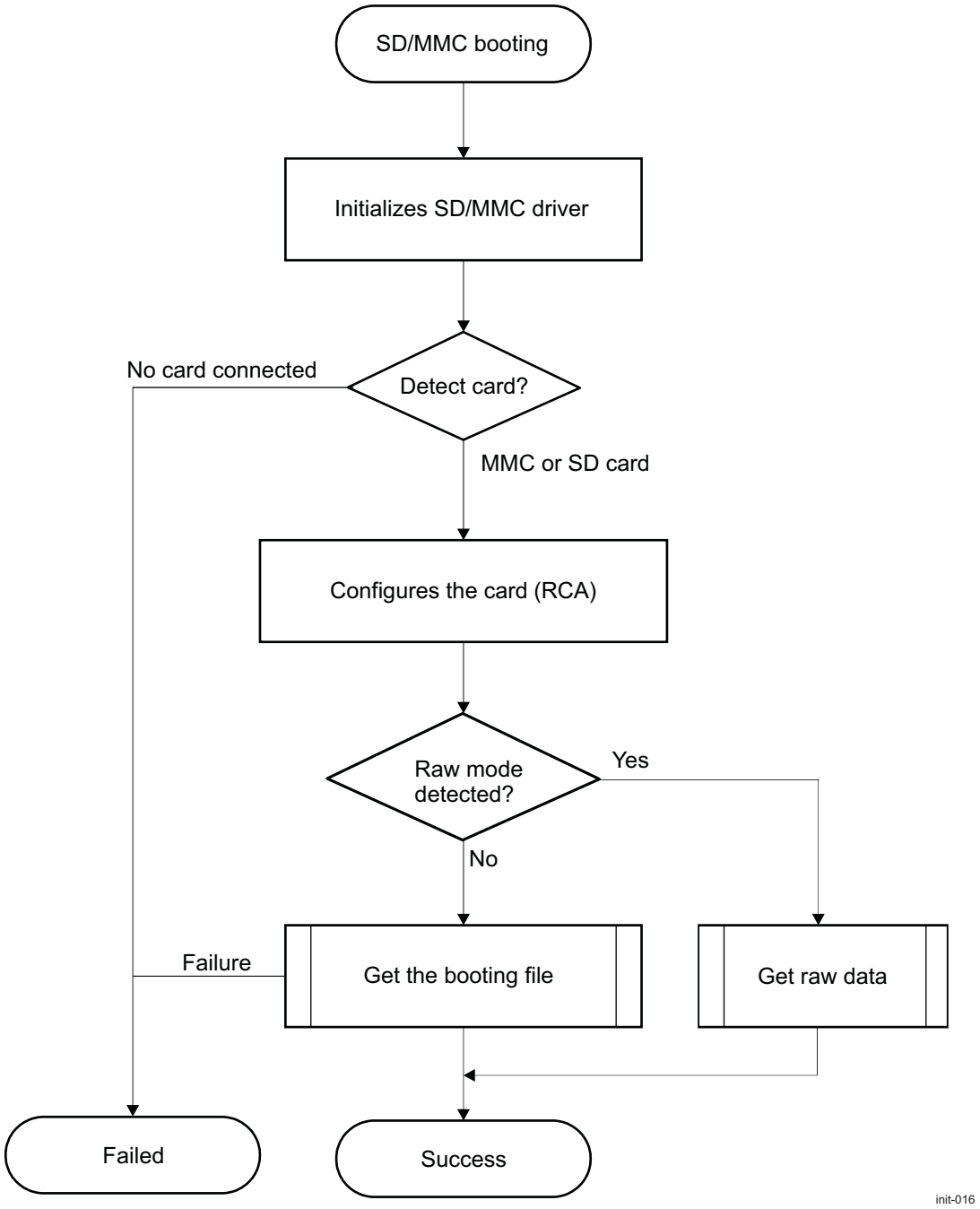
- In case of a FAT (12/16/32) formatted memory card, the booting file must not exceed 128 KB.
- In case of a raw mode memory card, the booting image must not exceed 128 KB.

For MMC memory booting, no user intervention is needed. The information in the following subsections is included for debugging. Failure in MMC/SD card detection causes a return to the main booting procedure, which selects the next device for booting.

The HS MMC/SD/SDIO host controllers (MMCHS) handle the physical layer, while the ROM code handles the simplified logical protocol layer (read-only protocol). A limited range of commands is implemented in the ROM code. The MMC/SD specification defines two operating voltages for standard or HS cards. The ROM code supports only 1.8V or 3.3V MMC/SD cards, depending on VDDSHV.

The ROM code reads a booting file from the card file system and boots from it. Figure 24-22 shows the complete procedure.

Figure 24-22. MMC/SD Booting



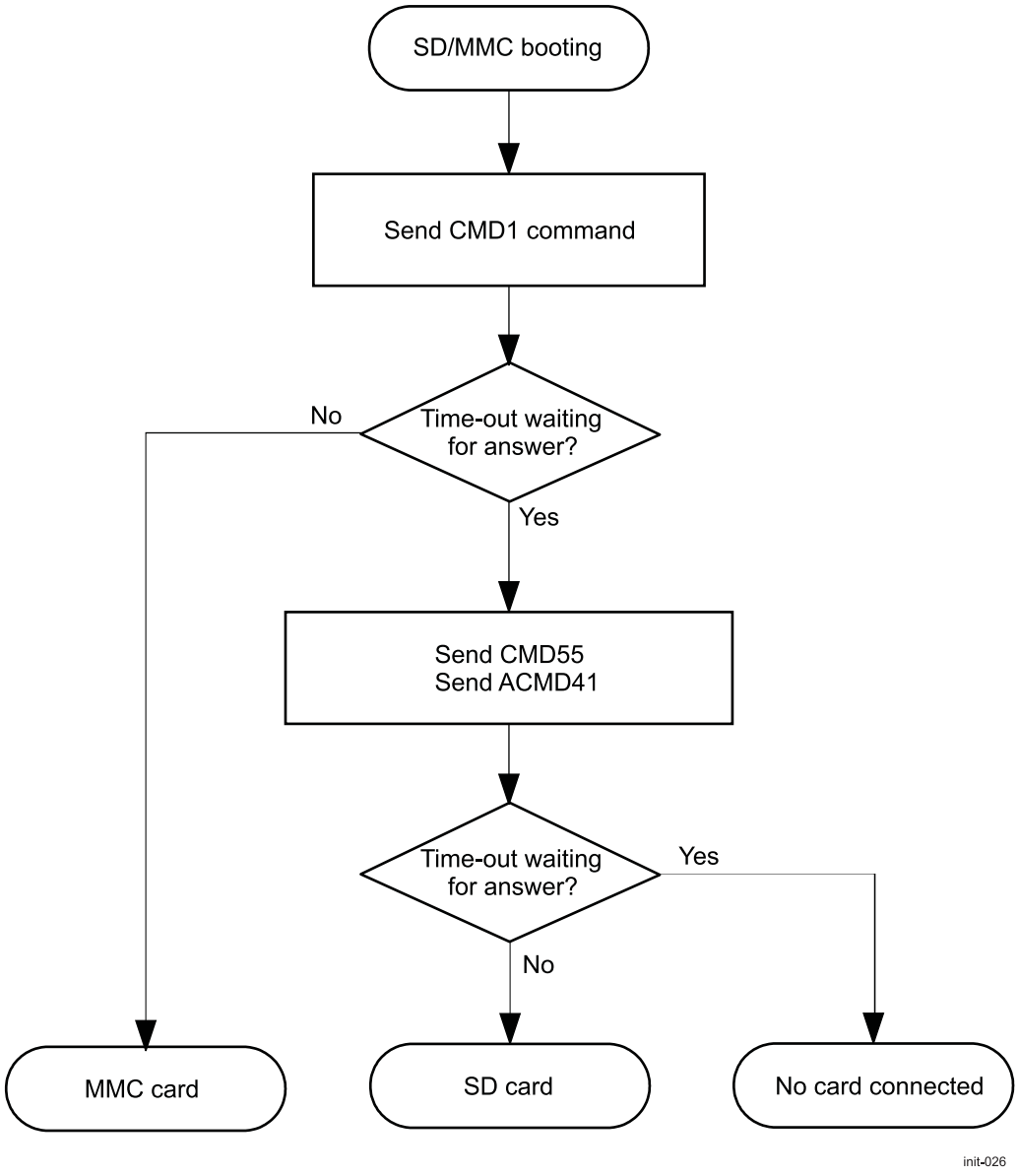
24.4.7.6.1 Initialization and MMC/SD Card Detection

The ROM code initializes the card connected on interface 1 using the voltage dictated by VDDSHV.. If no card is present, the ROM code goes to the next booting device. The standard identification process and relative card address (RCA) assignment are used. However, the ROM code searches for only one card connected on the bus. This is done using the CMD line common to the SD and MMC cards. The MMC and SD standards describe this phase as the initialization phase. They differ in the first commands, CMD1 and ACMD41. The ROM code uses this command difference to differentiate between MMC and SD cards;

that is, CMD1 is supported only by MMC and ACMD41 is supported only by SD. The ROM code first sends a CMD1 to the card and gets an answer only if an MMC card is connected. If no answer is received, ACMD41 (a combination of CMD55 and ACMD41) is sent, and an answer is expected from an SD card. If no answer is received, no cards are connected and the ROM code exits MMC/SD booting with FAIL.

Figure 24-23 shows the MMC/SD detection procedure.

Figure 24-23. MMC/SD Detection Procedure



24.4.7.6.2 Read Sector Procedure

- Raw mode
  In raw mode, an image can be at offset 0 or 128KB and must not be bigger than 128KB. Raw mode is detected by reading sector 0. ROM Code decides to switch into the raw mode if the first read sector contains a TOC structure with an MLO item. If raw mode is not detected, file system mode is assumed. Image data is read directly from continuous sectors of a card.
- File system handling
  The sector read procedure uses the standard MMC/SD read data procedure. The sector address is

generated based on the booting memory file map collected during initialization. Thus, the ROM code can address sectors freely in the booting file space.

24.4.7.6.3 File System Handling

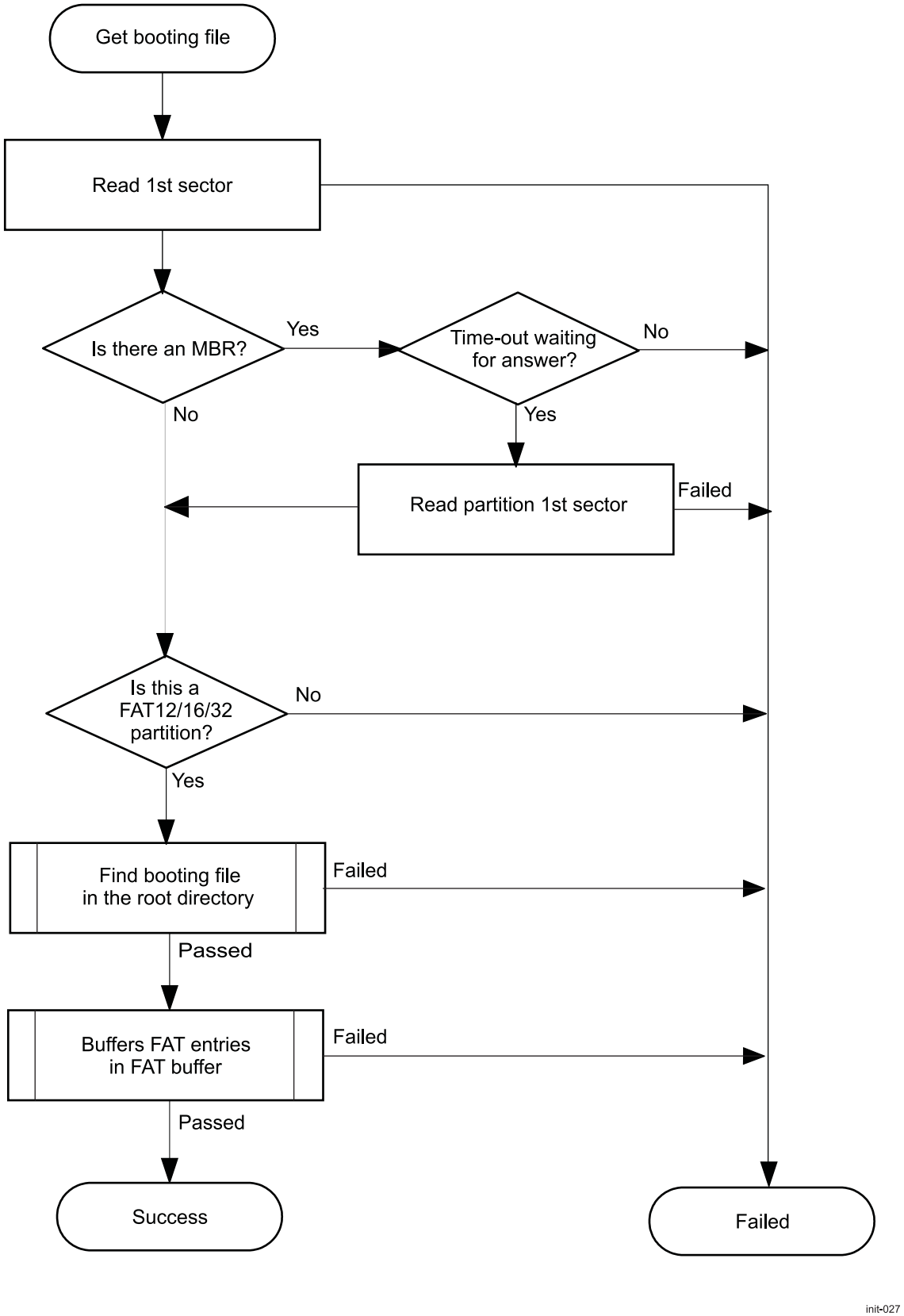
MMC/SD Cards may hold a file system which ROM Code reads. The image used by the Booting procedure is taken from a specific booting file named MLO. This file has to be located in the root directory on an active primary partition of type FAT12/16 or FAT32.

An MMC/SD card can be configured as floppy-like or hard-drive-like:

- When acting like a floppy, the content of the card is a single FAT12/16/32 file system without an MBR holding a partition table.
- When acting like a hard drive, an MBR is present in the first sector of the card. This MBR holds a table of partitions, one of which must be FAT12/16/32, primary, and active.

According to the *MultiMediaCard FAT16 File System Specification* from the MMCA Technical Committee, the card should always hold an MBR, except for MMC cards using a floppy-like file system. However, depending on the operating system used, the MMC/SD card is formatted with or without partition(s) (using an MBR). The ROM code supports both types: floppy-like or hard-drive-like. The ROM code retrieves a map of the booting file from the FAT table. The booting file map is a collection of all FAT table entries related to the booting file (a FAT entry points to a cluster holding part of the file). The booting procedure uses this map to access any 512-byte sector in the booting file without involving the ROM code FAT module. Figure 24-24 shows the complete process.

Figure 24-24. SD/MMC Booting



- MBR and FAT file system

This paragraph describes functionalities used by the ROM code to recognize whether an MBR with a FAT is used. It is not intended to fully describe the MBR and the FAT file system detection and reading procedure. The ROM code can detect FAT12/16/32 allocation table types. It cannot boot on devices with NTFS or Linux FS partitions. Some memory devices that support file systems can be formatted with or without MBR; therefore, the first task of the ROM code is to detect whether the device is holding an MBR in the first sector.

The MBR is the first sector of a memory device. It consists of executable code, four partition entries, and one signature. The aim of such a structure is to divide the hard disk in partitions used primarily to boot different systems (for instance, Microsoft Windows™). This structure is described in Table 24-38; partition table entry is described in Table 24-39.

Table 24-38. Master Boot Record Structure

Offset	Length [Bytes]	Entry Description
0000h	446	Optional code
01BEh	16	Partition table entry
01CEh	16	Partition table entry
01DEh	16	Partition table entry
01EEh	16	Partition table entry
01FEh	2	Signature (= 0xAA55)

Table 24-39. Partition Table Entry

Offset	Length [Bytes]	Entry Description	Value
0000h	1	Partition state	00h: Inactive 80h: Active
0001h	1	Partition start head	Hs
0002h	2	Partition start cylinder and sector	Cs[7:0]–Cs[9:8]–Ss[5:0]
0004h	1	Partition type	01h: FAT12 04h, 06h, 0Eh: FAT16 0Bh, 0Ch, 0Fh: FAT32
0005h	16	Partition end head	He
0006h	2	Partition end cylinder and sector	Ce[7:0]–Ce[9:8]–Se[5:0]
0008h	4	First sector position relative to the beginning of media	LBAs = Cs.H.S+ Hs.S+ Ss–1
000Ch	4	Number of sectors in partition	LBAe = Ce.H.S+ He.S+ Se–1 Nb s= LBAe–LBAs + 1

1. Find the booting file
   When a partition is found, the root directory entries are searched for a specific booting file named “MLO” inside the Root Directory of the FAT12/16/32 file system. The file is not searched in any other location. For a FAT12/16 file system, the Root Directory has a fixed location which is cluster 0. For a FAT32 file system, its cluster location is given by BPB_RootClus. The formula to find the sector number (relative to device sector 0, not partition sector 0) of a cluster is given by:

$$Cluster_{sector} = BPB\_HiddSec + BPB\_RsvdSecCnt + BPB\_NumFATs \cdot BPB\_FATSz + Cluster \cdot BPB\_SecPerCLus \quad \text{(23)}$$

init-E001

NOTE: BPB_FatSz is BPB_FatSz16 for FAT12/16 or BPB_FatSz32 for FAT32

NOTE: The BPB_HiddSec field can contain 0 even though the FAT file system is located somewhere other than on sector 0 (floppy-like). The ROM Code actually uses the partition offset taken from the MBR instead of this field which can be wrong. If no MBR was found (floppy-like) the value 0 is used.

Each entry in the Root Directory is 32 bytes long and hold information about the file, i.e. filename, date of creation, rights, cluster location etc. This is described in Table 24-40.

The ROM Code checks each entry in the Root Directory until either the booting file is found or the entry is empty (first byte is 00h) or when the end of the Root Directory has been reached. Entries with ATTR_LONG_NAME attribute (LFN) and with first byte at E5h (erased file) are ignored. When found, the first cluster offset of the file is read from the DIR_FstClusHi/DIR_FstClusLo fields.

There is a slight difference between FAT12/16 and FAT32 when handling the Root Directory. On FAT12/16, this directory has a fixed location (see above) and length fixed by BPB_RootEntCnt which is the total number of 32 bytes entries. Handling this directory is therefore straight forward. On FAT32, the Root Directory is like a standard file, the File Allocation Table (FAT) has to be used in order to retrieve each sector of the Directory. The way the FAT is handled is described in the following paragraph.

Table 24-40. FAT Directory Entry

Offset	Length [Bytes]	Name	Description	
0000h	11	DIR_Name	Short name (8 + 3)	
000Bh	1	DIR_Attr	File Attributes:	
			ATTR_READ_ONLY	01h
			ATTR_HIDDEN	02h
			ATTR_SYSTEM	04h
			ATTR_VOLUME_ID	08h
			ATTR_DIRECTORY	10h
			ATTR_ARCHIVE	20h
			ATTR_LONG_NAME	ATTR_READ_ONLY I ATTR_HIDDEN I ATTR_SYSTEM I ATTR_VOLUME_ID
000Ch	1	DIR_NTRes	Reserved, set to 00h	
000Dh	1	DIR_CrtTimeTenth	Millisecond stamp at file creation	
000Eh	2	DIR_CrtTime	Time file was created	
0010h	2	DIR_CrtDate	Date file was created	
0012h	2	DIR_LstAccDate	Last Access date	
0014h	2	DIR_FstClusHi	High work of this entry's first cluster number	
0016h	2	DIR_WrtTime	Time of last write	
0018h	2	DIR_WrtDate	Date of last write	
001Ah	2	DIR_FstClusLo	Low word of this entry's first cluster number	
001Ch	4	DIR_FileSize	File size in bytes	

2. Buffer FAT entries in the FAT buffer
When the booting file is found, the ROM code reads the FAT and buffers the singly-linked chain of clusters in a FAT buffer used by booting to access the file directly, sector by sector. For FAT12/16 and for FAT32, multiple copies of the FAT exist (ROM code supports only two copies) located after the Boot Sector:

$$FATn_{sector} = BPB\_HiddSec + BPB\_RsvdSecCnt + BPB\_FatSz \cdot n \quad \text{(24)}$$

init-E002

The size of the FAT buffer is given by BPB_FATSz16 or BPB_FATSz32. The ROM code checks each copy of the FAT if they are identical. If the values are different, the ROM code uses the value from the last FAT copy. With the FAT32 file system, the copy system can be disabled according to a flag in BPB_ExtFlags[7]. If this flag is set, the FAT BPB_ExtFlags[3:0] bit field is used. In this case no verification is made by the ROM code with other copies of FAT.

The FAT is a simple array of values, each referring to a cluster located in the data area. One entry of the array is 12, 16, or 32 bits, depending on the file system in use. The value in an entry defines whether the cluster is being used or not, and if another cluster must be considered. This creates a singly-linked chain of clusters defining the file. The meaning of an entry is described in Table 24-41.

NOTE: For compatibility, clusters 0 and 1 are not used for files and those entries must contain FF8h and FFFh (for FAT12); FFF8h and FFFFh (for FAT16); ?FFFFFF8h and ?FFFFFFFh (for FAT32).

Table 24-41. FAT Entry Description

FAT12	FAT16	FAT32	Description
000h	0000h	?0000000h	Free Cluster
001h	0001h	?0000001h	Reserved Cluster
002h-FEFh	0002h-FFEFh	00000002h-?FFFFFEFh	Used Cluster; value points to next cluster
FF0h-FF6h	FFF0h-FFF6h	?FFFFFF0h-?FFFFFF6h	Reserved values
FF7h	FFF7h	?FFFFFF7h	Bad Cluster
FF8h-FFFh	FFF8h-FFFFh	?FFFFFFh-?FFFFFFFh	Last Cluster in File

NOTE: FAT32 uses only bits [27:0]; the upper 4 bits are usually 0 and must be left untouched.

When accessing the root directory for FAT32, the ROM code starts from the root directory cluster entry and follows the linked chain to retrieve the clusters.

When the booting file has been found, the ROM code buffers each FAT entry corresponding to the file in a sector way. This means each cluster is translated to one or several sectors, depending on how many sectors are in a cluster (BPB_SecPerClus). This buffer is used later by the booting procedure to access the file.

24.4.7.7 DiskOnChip™

The ROM code support for DiskOnChip™ devices has the following characteristics:

- DiskOnChip™ H3
- 1.8-V power supply
- Density from 256M bits

The DiskOnChip™ contains an XIP part that can hold a boot image. In this case, the DiskOnChip™ is connected just like a regular NOR device, and uses the same procedure as for NOR described in Section 24.4.7.3.

If booting determines that it must boot from DOC, the ROM code initializes the GPMC using standard asynchronous, 16-bit, multiplexed mode. The DOC device is then detected using the M-System driver. This proprietary driver handles all DiskOnChip protocols and identification. If the device cannot be identified, the ROM code returns a failed code. Otherwise, booting directly requests sectors from the M-System driver.

SD/MMC booting consists of several steps:

1. MBR detection
   The ROM code first checks whether the MBR signature is present and then it searches an active FAT12/16/32 partition in all four MBR partition entries, based on the Type field. If the MBR entries are not valid or if no usable partition is found, the ROM code returns to the booting procedure with FAIL.

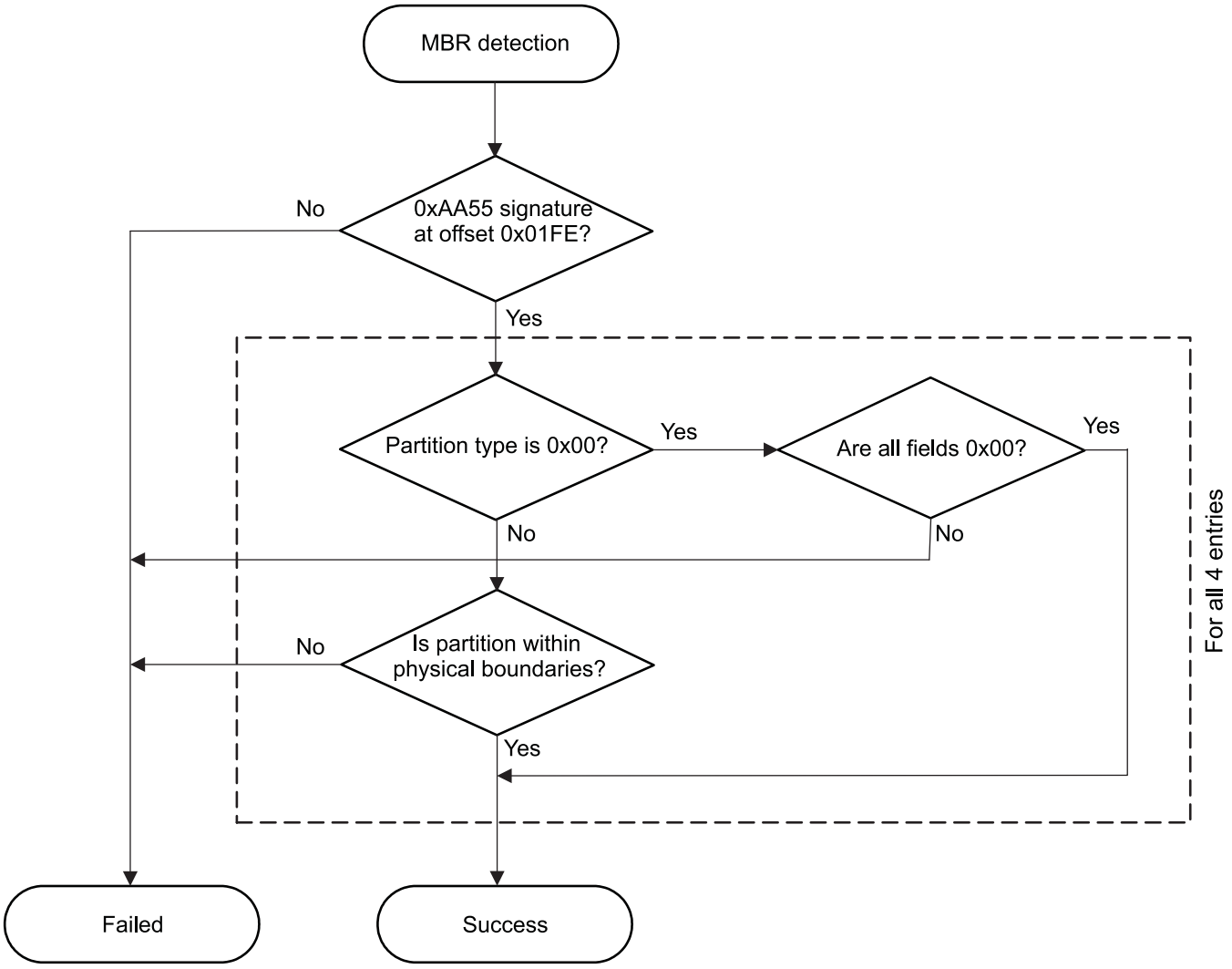
The extended partitions are not checked; the booting file must reside in a primary partition. Each partition entry is checked:

(a) If its type is set to 00h, all fields in the entry must be 00h.

(b) The partition is checked to be within physical boundaries (that is, the partition is inside and it fits the total physical sectors).

See Figure 24-25 for more information about MBR detection.

Figure 24-25. MBR Detection Procedure

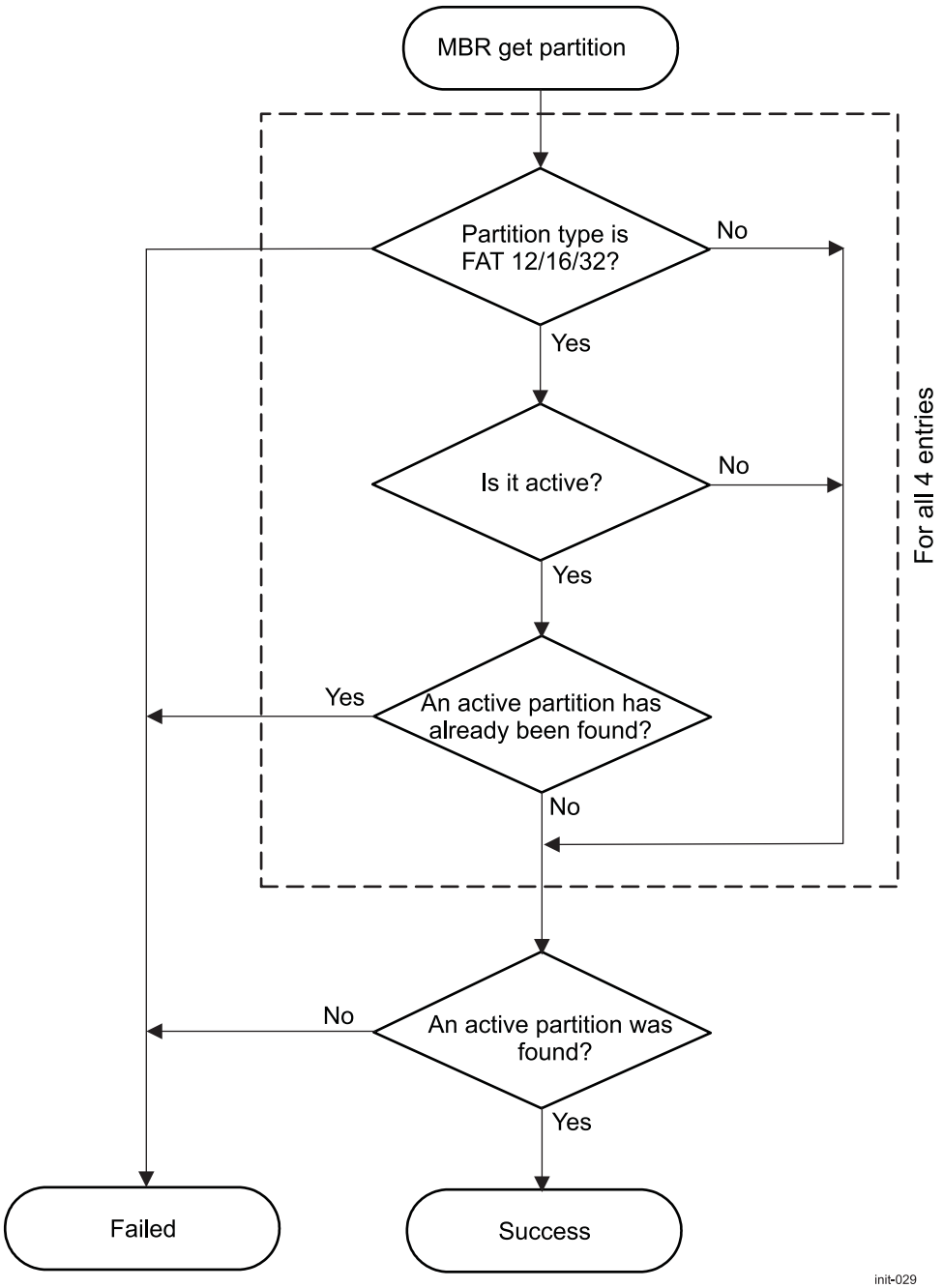


2. Get the MBR partition.
   Once identified, the ROM code gets the partition using the procedure described in Figure 24-25. The partition type is checked to be FAT12/16 or FAT32. Its state must be 00h (inactive) or 80h (active.) The ROM code returns with FAIL if no active primary FAT12/16/32 is found, or if there is more than one active partition, the test fails. If an active partition is found, its first sector is read and used later. If no MBR is present (in case of a floppy-like system), the first sector of the device is read and used later. The read sector is checked to be a valid FAT12/16 or FAT32 partition. If this fails, if another partition type is used (for instance, Linux® FS) or if the partition is not valid, the ROM code returns with FAIL. The FAT file system consists of several parts:
   - Boot sector, which holds the BIOS parameter block (BPB). Not all are used by the ROM code.
   - FAT, which describes the use of each cluster of the partition

- Data area, which holds the files, directories, and root directory (for FAT12/16, the root directory has a specific fixed location).

To check whether a sector holds a valid FAT12/16/32 partition, many fields of the boot sector (used by all FAT types) that must have specific values are checked. Figure 24-26 shows more information about getting the MBR partition.

Figure 24-26. Get MBR Partition



24.4.7.8 SPI Flash

ROM code supports SPI flashes that have the following characteristics:

- Only 24 bit addressable flashes are supported
- Only flash sizes between 128KBytes and 16 MBytes are supported

- Only Mode 3 flashes are supported (Phase = 1, Polarity = 1)
- The SPI flash must be connected to CS0
- Only 4 pin mode is supported (spi_clk, spi_cs0, spi_somi, spi_simo)
- SPI clock frequency is 12 MHz

SPI Sector Read Procedure

The ROM code searches the first four sectors of the SPI flash looking for a valid image. If the first two words of any sector is neither 0x0000 0000 or 0xFFFF FFFF, then ROM code assumes a valid image is present.

Once the ROM code detects a valid image, it reads the size of the image from the header , copies the image to the destination address mentioned in the header and executes it.

The ROM code assumes one sector is 512 bytes. To read a particular address from the SPI flash, the ROM code issues a READ command (0x03) followed by the 24 bit address of the byte to be read from the flash. The ROM code then reads the desired number of bytes, 4 bytes at a time.

The chip select is controlled by the ROM code, and is asserted before sending the read command. It is de-asserted only once the required number of bytes are read.

24.4.8 Image Format

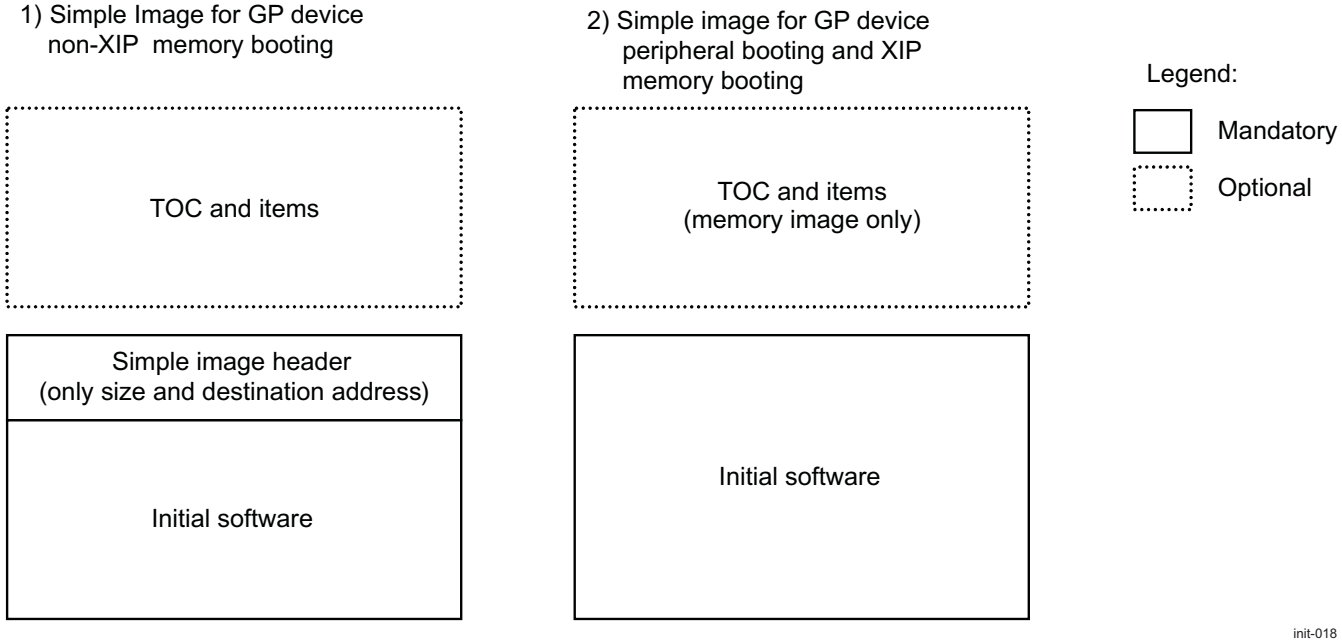
24.4.8.1 Overview

The image is the SW which is loaded into the memory and executed.

An overview of the image formats is shown in Figure 24-27. There are two image types for GP devices:

1. GP non-XIP memory booting:
   This image type is used for memories that require shadowing. The image must begin with a simple header that contains information on size to copy and destination. This format is detailed in Section 24.4.8.3, *Image Format for GP Devices*.
2. GP XIP memory booting and peripheral booting:
   GP image on XIP memory contains only code. The GP peripheral booting image contains only code.

Figure 24-27. Image Format



24.4.8.2 Image Header

The beginning of the image is a table of contents (TOC) pointing to each item. This is described in Figure 24-28. Each TOC item is a simple structure described in Table 24-42. The complete image header (TOC and items) should fit in a 512-byte sector.

The ROM code reads the first TOC item and if it contains a known string (MLO) then the TOC is identified and searched until a 0xFFFF FFFF offset is found.

Figure 24-28. Image Header Format

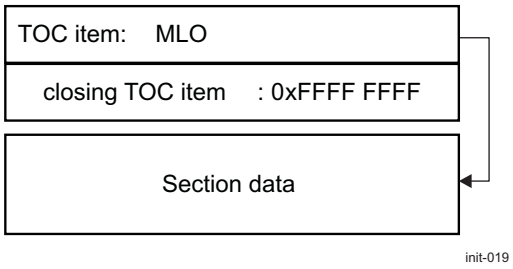


Table 24-42. TOC Item

Offset	Field	Size [bytes]	Description
0x0000	Start	4	Offset from the start address of TOC to the actual address of a section
0x0004	Size	4	Size of a section
0x0008	reserved	4	Unused
0x000C	reserved	4	Unused
0x0010	reserved	4	Unused
0x0014	Filename	12	12-character long name of a section, including the zero (\0) terminator.

24.4.8.3 Image Format for GP Devices

For a GP device, no security is enabled and, therefore, neither keys nor certificates are required. The image is simple and must contain a small header having the size of the SW to load and the destination address of where to store it when a device is other than XIP. The XIP device image is even simpler and starts with executable code. The image format is described in Table 24-43. The header is used only for memory booting. The peripheral booting image does not have any header and starts directly with executable code.

Table 24-43. GP Device SW Image

Filed	Non-XIP Device	XIP Device	Size (Bytes)	Description
	Offset	Offset		
Size	0x0000	-	4	Size of the image
Destination	0x0004	-	4	Address where to store the image
Image	0x0008	0x0000	x	The Image

24.4.8.4 Image Execution

The image is executed when the ROM code performs a branch instruction to the first executable instruction inside the Initial SW. The execution address is the first word after the image header. After the branch, the ARM runs in public ARM supervisor mode. The R0 register points to the booting parameter structure that contains some information about the booting process. Table 24-44 details the booting parameters structure.

Table 24-44. Booting Parameters Structure

Offset	Field	Size [Bytes]	Description
0x00	Booting message	4	Last received Booting Message.
0x04	Current booting device	1	Code of device used for booting: 0x01: XIP memory 0x02: NAND 0x03: OneNAND 0x04: DOC 0x05: MMC/SD2 0x06: MMC/SD1 0x07: XIP memory with wait monitoring 0x10: UART 0x11: HS USB 0x12: EMAC 0x13: EMAC
0x05	Reserved	1	Reserved
0x06	Reset reason	1	Current reset reason bit mask (bit = 1, event present): [0]: Power-on reset [1]: Global SW reset [3]: Violation reset [4]: MPU watchdog reset [5]: Watchdog reset [6]: External warm reset Other bits: Reserved
0x07	Reserved	1	Reserved
0x08	Device descriptor	4	Pointer to the device descriptor structure. This pointer is required when current booting device driver functions are called.

24.4.9 Tracing

Tracing in the public ROM code consists in a 64-bit vector in which each bit corresponds to a certain point of the ROM code execution flow. The tracing vector is divided into two 32-bit words. Table 24-45 lists the location and organization of the tracing data in RAM. Tracing data is initialized at the beginning of the start-up phase.

There are two sets of tracing data: the first set is the current trace information, the second set holds tracing collected during the first ROM code run following a cold reset. This cold reset tracing is copied into its location during the tracing initialization of the second ROM code run after the cold reset run. These data can be used for debugging purposes.

Table 24-45. Tracing Vector

Bit No.	Group	Meaning
0	General	Reset
1	General	ROM code C main
2	General	ROM code runs after the cold reset
3	Boot	Booting started
4	Memory boot	Memory booting started
5	Boot	No more device to check
6	Peripheral boot	Peripheral booting started
7	Boot	Booting message change device
8	Boot	Booting message skip per. booting
9	Reserved	Reserved
10	Reserved	Reserved
11	Memory boot	Image header correct
12	Peripheral boot	Device initialized
13	Peripheral boot	ASIC ID sent
14	Peripheral boot	Booting message received
15	Peripheral boot	Image received
16	Peripheral boot	Peripheral booting failed
17	Peripheral boot	UART
18	Peripheral boot	USB
19	Peripheral boot	EMAC
20	Reserved	Reserved
21	Peripheral boot	NULL device
22	Execute	Image executed
23	Reserved	Reserved for non-GP devices
24	Reserved	Reserved for non-GP devices
25	Reserved	Reserved for non-GP devices
26	Reserved	Reserved for non-GP devices
27	Reserved	Reserved for non-GP devices
28	Reserved	Reserved for non-GP devices
29	Boot	Software booting configuration section 1 found
30	General	Software booting configuration clocking section found
31	Reserved	Reserved
32	Memory boot	Null device
33	Memory boot	XIP
34	Memory boot	NAND
35	Memory boot	OneNAND
36	Memory boot	DOC
37	Memory boot	MMC/SD2
38	Memory boot	MMC/SD1
39	Memory boot	XIP memory with wait monitoring
40	Memory boot	SPI

24.5 Debug Configuration

24.5.1 Overview

This section provides information for using a debugger on the public ARM Cortex™-A8 in the processor. The debug capability is accessible through the JTAG interface with a limited number of pins.

24.5.2 JTAG Port Signal Description

The target debug interface uses the five standard IEEE 1149.1 (JTAG) signals (nTRST, TCK, TMS, TDI, and TDO), a return clock (RTCK), and the two instrumentations pins (EMU0, EMU1). For more information, see Table 24-46.

Table 24-46. Debug POR Signals

Pin	Type[1]	Name	Description
nTRST	I	Test logic reset	When asserted (active low), causes all test and debug logic in the device to be reset with the IEEE 1149.1 interface
TCK	I	Test clock	This is the test clock used to drive an IEEE 1149.1 test access port (TAP) state-machine and logic. Depending on the emulator attached to the processor, this is a free-running clock or a gated clock, depending on RTCK monitoring.
RTCK	O	Returned test clock	Synchronized TCK. Depending on the emulator attached to the processor, the JTAG signals are clocked from RTCK or RTCK is monitored by the emulator to gate TCK.
TMS	I	Test mode select	Directs the next state of the IEEE 1149.1 TAP state-machine
TDI	I	Test data input	Scan data input to the device.
TDO	O	Test data output	Scan data output of the device.
EMU0	I/O	Emulation 0	Channel 0 trigger: Boot mode
EMU1	I/O	Emulation 1	Channel 1 trigger: Boot mode

(1) I = Input, O = Output

24.5.3 Initial Scan Chain Configuration

The general-purpose ports that can provide access to many test support functions built into the device (including the test logic defined by the IEEE 1149.1 standard) are the TAP.

The first level of the debug interface that sees the scan controller is the TAP router module.

The debugger can configure the TAP router to serially link to 16 TAP controllers or to individually scan one TAP controller without disrupting the instruction register (IR) state of the other TAPs. The initial scan chain configuration of the device is determined from the level of the EMU0 and EMU1 pins on the release of POR. At POR, EMU0 and EMU1 are automatically configured as inputs. The EMU0 and EMU1 pins should be pulled high at POR to configure the initial scan chain of the device to TAP router-only mode. In the TAP router-only configuration, no secondary TAPs are selected. The TAP router is the only TAP between the device-level TDI and TDO.

The router TAP has an IR length of 6 bits. This is the recommended boot mode. The third-party debugger must assume that the TAP router is the only TAP between TDI and TDO at boot.

24.5.4 Debugger Address Space

The CoreSight components are interfaced with the TAP router through the DAP. As recommended by the CoreSight architecture, the DAP is directly interfaced to the device bus. The debugger can directly access the entire memory space without requiring the processor to enter debug state and be programmed with a load or store instruction. Table 24-47 lists the modules that are mapped to the DAP address, and the address space detail.

Table 24-47. Debugger Address Space

Start Addr (Hex)	End Addr (Hex)	Size	Description
0xD401 0000	0xD401 0FFF	4KB	ARM embedded trace macrocell (ETM) module
0xD401 1000	0xD401 1FFF	4KB	Cortex-A8 module
0xD401 9000	0xD401 9FFF	4KB	Trace port interface unit (TPIU)
0xD401 B000	0xD401 BFFF	4KB	ARM embedded trace buffer (ETB) module

The DBGEM signal on the Cortex-A8 is driven by setting bit 13 at address 0xD401 D030 in the DAP-APB address space.

24.6 Revision History

Table 24-48 lists the changes made since the previous version of this document.

Table 24-48. Document Revision History

Reference	Additions/Modifications/Deletions
Global	Removed SW Booting Configuration.
Section 24.2.3	Changed sys_boot[8:0] to sys_boot[5:0] in 2nd paragraph 1st sentence.
Table 24-3	Added new boot configuration to boot sequence.
Table 24-4	Added new boot configuration to boot sequence.
Table 24-5	Added new boot configuration to boot sequence.
Section 24.4.1.2	Changed 2nd bullet.
Section 24.4.3	Removed last 3 sentences from step 3 and deleted last paragraph.
Figure 24-8	Removed the check for configuration header in step 5 from figure.
Section 24.4.4.2	Removed software configuration information including: 2nd sentence of 4th paragraph, 5th paragraph, and last bulleted list.
Section 24.4.4.3	Changed 1st sentence and removed bulleted list.
Figure 24-9	Removed software configuration from the flow chart figure.
Section 24.4.7.1	Added last bullet and removed 2nd and 3rd sentence from 3rd paragraph.
Figure 24-16	Removed check for Configuration Header (CH) from figure.
Section 24.4.7.2	Removed 2nd paragraph.
Figure 24-17	Removed Configuration Header (CH) path from figure.
Section 24.4.7.3	Removed Configuration Header (CH) information from 3rd paragraph and from ordered list.
Section 24.4.7.4	Removed 2nd and 3rd sentences from 2nd paragraph.
Section 24.4.7.5	Removed 3rd and 4th sentences from 3rd paragraph.
Section 24.4.7.6	Updated 2nd and 3rd bullets and 3rd paragraph.
Section 24.4.7.6.1	Changed 1st sentence, 1st paragraph.
Section 24.4.7.6.2	Updated RAW mode bullet.
Section 24.4.7.8	Added subsection.
Figure 24-27	Removed Configuration header from figure.
Section 24.4.8.2	Removed 1st paragraph and bulleted list.
Figure 24-28	Updated figure.
	Removed CHSETTINGS subsection.
	Removed CHRAM subsection.
	Removed CHFLASH subsection.
	Removed CHMMCSD subsection.
Table 24-44	Changed Offset 0x07 to reserved.
Table 24-45	Changed bit 10 to reserved, bit 19 meaning to EMAC, and added bit 40.

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		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265